

Wolaita So University

School of Informatics

Department of Information Technology

Internet Programming Module for Exit Examination

Prepared By:

Alemayehu Dereje (MSc. In Information Technology)

Email:

alemayehu.dereje@wsu.edu.et

alem20@gmail.com

Reviewed by:

Nuhamin Nigussie (MSc.)

Biruk Sidamo(MSc.)

February 2023/2015 E.C

Wolaita, Ethiopia

Contents

Objectives of the module 3

Chapter 1 Internet Technologies and Protocols 4

1.1. What is Internet? 4

1.2. Advantage of Internet 4

1.3. Disadvantage of Internet 4

1.4. Terminology 4

1.5. Web development tools 6

1.6. Client-server architecture 7

1.7. How the Web works? 7

1.8. Website validation 8

Chapter 2 HTML 10

2.1. What is Markup language? 10

2.2. What is Hyper Text Markup Language (HTML)? 10

2.3. The structure of an HTML element 10

2.4. The minimal structure of HTML documents: 10

2.5. HTML Element 11

2.6. Heading Tags 11

2.7. HTML Links 13

2.8. Conditional Comments 15

2.9. HTML Tables 15

2.10. HTML Forms 17

Chapter 3 Styling HTML with CSS 26

3.1. What is CSS? 26

3.2. Why Use CSS? 26

CSS Syntax and Selectors CSS Syntax 26

3.3. CSS Selectors 26

3.4. CSS Comments 28

3.5. Inline Styles 29

3.6. Internal Style Sheet 29

3.7. External Style Sheet 29

Chapter 4 JavaScript 31

4.1. Introduction 31

4.2. Benefits of JavaScript 31

4.3. JavaScript Rules 31

4.4. Data types and primitives 33
4.5. JavaScript Popup Boxes 33
4.6. Operators in JavaScript 34
4.7. Functions in JavaScript 35
4.8. Objects in JavaScript 36
4.9. Regular Expression: 37
4.10. Cookies in JavaScript What are cookies? 38
4.11. Using JavaScript on HTML forms 39
4.12. What is the DOM? 39
Chapter 5 Server-side programming 42
5.1. Introduction to server-side scripting 42
5.2. Differences between Client-side and Server-side Scripting 42
5.3. Client-Side Scripting 42
5.4. Server-Side Scripting 42
5.5. What is PHP? 43
5.6. Use Basic Syntax 43
5.7. Sending Data to the Web Browser 44
Example using Print <?php 45
5.8. Comments in PHP 45
5.9. Utilize Variables 46
5.10. Rules for PHP variables: 47
5.11. PHP has three different variable scopes: 47
5.11. PHP the global Keyword 48
5.12. PHP the static Keyword 49
5.14. Operators in PHP 51
Laboratory Exercises and Projects 56
Question 1 56
Question 2 An html code that creates a link and insert image into webpage 56
Question 3 57
Question 4 Creating Table 57
Question 5 Bullets and Numbering 58
Question 6 Forms and html 59
Question 7 Inline and internal style 61
Question 8 Internal stylesheet 61
Question 9 Multiple stylesheet 62
Question 10 External Stylesheet 62

Objectives of the module

The main objective of this module is to prepare information technology students for national exit examination.

The following are the specific objectives

- Acquire the skills of developing static and dynamic websites, in creating Server/client-side scripts for commercial and scientific programs
- Collaborate in teams to accomplish a common goal by integrating personal initiatives
- Communicate effectively and efficiently with clients, users and peers both verbally and in writing, using appropriate terminology.
- Understand and explain the quantitative dimensions of a problem.

Chapter 1 Internet Technologies and Protocols

1.1. What is Internet?

- It is a world-wide global system of interconnected computer networks. A ***computer network*** is the interconnection of many individual computers to exchange message. ***Network*** is a group of two or more computer connected together.
- It uses the standard Internet Protocol ***TCP/IP***. Every computer in internet is identified by a unique ***IP address***. IP Address is a unique set of numbers which identifies a computer location. Domain Name server (DNS) is used to give *name to the IP Address* so that user can locate a computer by a name.
  - *For example, a DNS server will resolve a name http://www.google.com to a particular IP address to uniquely identify the computer on which this website is hosted.*
- Internet is accessible to every user all over the world. So, Internet is a network of networks.

1.2. Advantage of Internet

- Information sharing
- Communication i.e. social networking
- Sharing of resource

1.3. Disadvantage of Internet

- Threat to personal information
- Virus attack
- Spamming
- Cyber crime

1.4. Terminology

 Learning about the Internet can be a bit confusing at first, but it becomes a lot simpler if you can become familiar with some of the terminology used when talking about the Internet. Here is a list of common words and phrases that you might hear.

A. ***On-line***

- You may sometimes hear people talk about "***being on-line***". This is just another way of saying that ***they are using the Internet***.

B. ***World-Wide-Web*** **(WWW)**

- Tim Berners-Lee, a physicist in Switzerland, invented the World Wide Web in 1992 as a ***way to organize and access information on the Internet***.

C. ***Web browser***

- ☞ A web browser is a program that runs on users' computers and allows them to *view and interact with the web pages on the World Wide Web*. The most common web browsers are called ***Internet Explorer and Google Chrome***.

D. ***Hypertext***

- ➲ Hypertext is a text document that contains links to other text document.
- ➲ It allows a user to ***move from one web page to another*** by using a mouse to click on special ***hypertext links***.

 For example, a user viewing web pages that describe airplanes might encounter a link to jet engines from one of those pages. By clicking on that link, the user automatically jumps to a page that describes jet engines. E. ***Webpage***

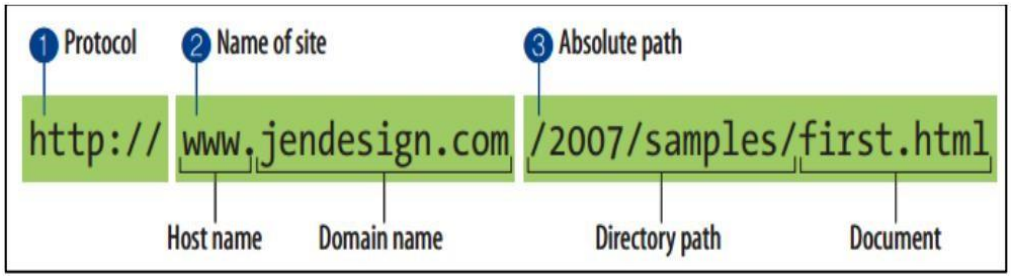
- ➲ Webpage is a single document that may contain text, graphics, and icon.
- ➲ They are created using HTML.

F. ***Web site***

- ➲ It is a group of well-structured and interlinked webpages.

G. ***Uniform Resource Locator* (URL)**

- ➲ To visit a Web site, users type the URL, which is the site's address, into the web browser. An example of a URL is ***www.yahoo.com*.**
- ➲ A complete URL is generally made up of three components: the ***protocol***, the ***site name***, and the ***absolute path*** to the document or resource as shown in the figure below:



H. ***Web server***

- ➲ A *web server* is a ***computer that stores a web site***, and is responsible for checking requests for viewing that web site.
- ➲ Client computers ***send requests*** for particular URLs to the web server, which then finds the appropriate web page, and sends it back to the client computer.
- ➲ A web server on the Internet must have a ***permanent Internet connection***, so that whenever a client computer requests a URL, the web server can respond straight away.

I. ***Internet Service Provider*** **(ISP)**

 A company that provides Internet connections to customers.

J. ***Protocol***

 It is a set of rules that govern the communication.

K. ***Hypertext Transfer Protocol (HTTP/HTTPS)*** It is a communications protocol.

- ➲ It defines mechanism for communication between ***browser and the web server***. It is also called ***request and response protocol*** because the communication between browser and server takes place in request and response pairs. Simply, it is the means by which ***computers on the WWW communicate***.
- ➲ HTTPS is the secure version of HTTP. It is used on web sites where ***sensitive information*** such as bank details is exchanged.

L. ***Hypertext Markup Language (HTML)***

- ➲ It is the language used to write web pages on the WWW.

M. ***Extensible Markup Language (XML)***

- ➲ It is an alternative language for writing web pages. Whereas HTML pages describe the format of the data's presentation, pages written in XML describe only how the data is structured.
- ➲ XML provides a standard format for the movement of data in and between applications.
- ➲ The data in an XML file usually requires some other application to interpret the data and display it in a useful format.

N. ***World Wide Web Consortium (W3C)***

- ➲ It is a group of experts who meet regularly to develop common protocols for the evolution of the WWW.
- ➲ The W3C agrees on standards for HTML, XML and other web technologies, and for how web browsers should interpret them.

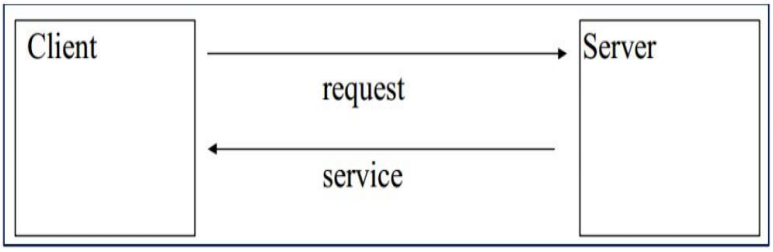
1.5. Web development tools

 A number of tools exist for use by web authors (i.e. people who write web sites).

 HTML/XML: HTML and XML are the two main languages used for writing web pages. Web authors can use a simple text editor such as ***Notepad*** to enter the *HTML/XML* commands. The final page can then be viewed using a web browser.

1.6. Client-server architecture

The data processing is split into distinct parts. A part is either requester (client) or provider (server). The client sends during the data processing one or more requests to the servers to perform specified tasks. The server part provides services for the clients.



1.7. How the Web works?

 What happens when a browser requests an HTML document over the Internet?

1. *The user types a URL into the Web browser to identify which Web page they would like to view.*
2. *The browser parses the URL and requests a DNS server using broadcast IP. It then sends the URL to the DNS to resolve the IP address. In other words, it converts appdev.com to 206.191.222.239.*
3. *The browser then uses the IP address to send the HTTP packet to the browser's ISP connection, which passes it to the next server, routing it from server to server until it reaches the destination Web server.*

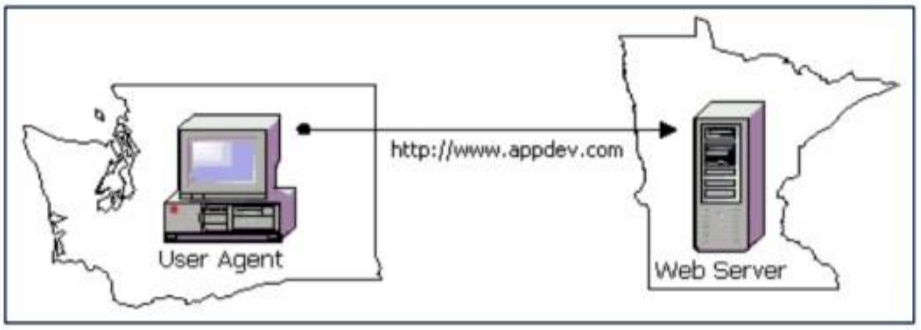


Figure 1: A request is made to the server.

4. *The Web server locates the request page either on its hard drive or cached in memory.*
5. *The Web server sends its contents back to the requested browser.*

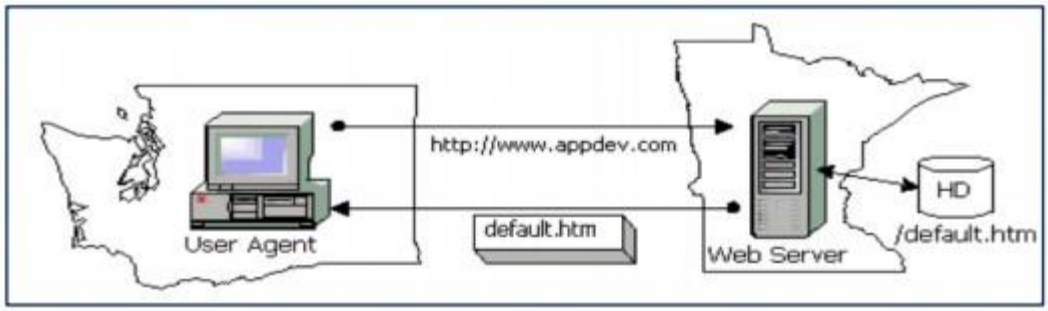


Figure 2: -The page is located and returned to the client machine.

6. *The browser interprets the HTML formatting instructions and displays the content to the user.*

1.8. Website validation

1. **Authorship:** Who put up the site?
   - ☞ The name of the individual or organization creating the site should be clearly stated.
   - ☞ The site should list the credentials of the author, whether it is an individual or an organization.
   - ☞ The site should provide a way for users to contact the author and to make comments or ask questions.
2. **Purpose:** Every site has a reason for being on the web
   - ☞ A site's purpose should be clear and its content should reflect its purpose, be it to inform, entertain, persuade, educate or sell.
   - ☞ Bias (if any) should be clearly stated through a mission statement or "about us" section or elsewhere on the site.
3. **Content & Currency:** Is the information authoritative and up to date?
   - ☞ The information should be accurate and the site should be updated regularly, especially if the topic is time-sensitive.
   - ☞ The site's content should be easy to read and easy to understand by its intended audience.
   - ☞ The site should offer enough information to make it worth visiting.
4. **Technical Aspects**
   - ☞ A search function should be provided for sites with large amounts of information.
   - ☞ You should not have to pay to view the information on the site.
   - ☞ Spelling and grammar should always be correct.
   - ☞ Links to more information should be provided.
   - ☞ Graphics on the site should be relevant and appropriate to the content.

- ☞ Advertising should be limited.

Putting it all together: If the website you found provides:

- ☞ Author name, acceptable author credentials and a way to contact the author.
- ☞ A clear statement of purpose or mission.
- ☞ accurate information (as measured by the citations for information on the site OR by what you already know about the topic OR by comparing it to information from an authoritative source)
- ☞ up-to-date information

Chapter 2

HTML

2.1. What is Markup language?

- A markup language is a programming language that is used to make text more ***interactive and dynamic***. It can turn a text into images, tables, links etc.

2.2. What is Hyper Text Markup Language (HTML)?

- *It is the standard markup language for creating Web pages.*
- *It is a language for describing web pages.*
- *HTML documents contain **HTML tags and plain text***
- *HTML documents are also called **web pages***
- *It consists of a series of elements*
- *HTML elements tell the browser how to display the content*
- *HTML elements are represented by tags*
- *HTML tags label pieces of content such as heading, paragraph, table, and so on*
- *Browsers do not display the HTML tags, but use them to render the content of the page*

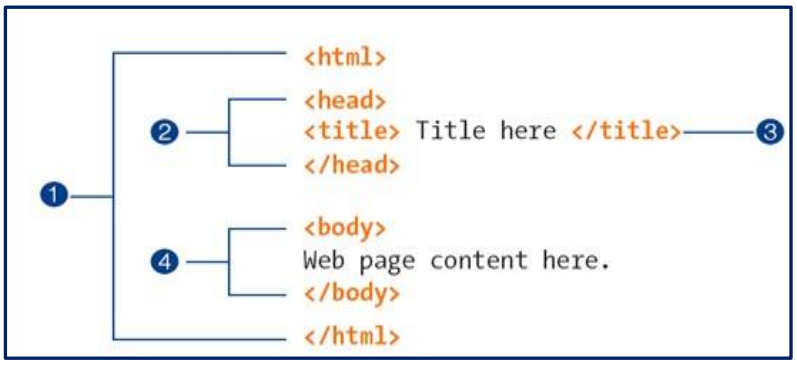
2.3. What is HTML Tag?

- HTML tags are element names surrounded by angle brackets:

  <tagname>** content goes here...**</tagname>

- HTML tags normally come in pairs like and
- The first tag in a pair is the ***start tag***, the second tag is the ***end tag***
- The end tag is written like the start tag, but with a ***forward slash inserted before the tag name***

2.3. The structure of an HTML element



2.4. The minimal structure of HTML documents:

1. Identifies the document as written in HTML.

2. The head provides information about the document.
3. A descriptive title is required.
4. The body contains the content that displays in the browser.

2.5. HTML Element

- The **<!DOCTYPE html>** declaration defines this document to be ***HTML version 5***
- The **<html>** element is the ***root element of an HTML page***
- The **<head>** element contains ***meta information about the document***
- The **<title>** element specifies a ***title for the document***
- The **<body>** element contains the ***visible page content***
- The **<h1>** element defines a ***large heading***
- The ** ** element defines a ***paragraph***

2.6. Heading Tags

- Any document starts with a heading. You can use different sizes for your headings.
- HTML also has six levels of headings, which use the elements <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>. While displaying any heading, *browser adds one line before and one line after that heading.*

Example

```
<!DOCTYPE html>
<html>
<head> <title>Heading Example</title> </head>
<body>
<h1>this is heading 1</h1>
<h2>this is heading 2</h2>
<h3>this is heading 3</h3>
<h4>this is heading 4</h4>
<h5>this is heading 5</h5>
<h6>this is heading 6</h6>
</body>
</html>
```

Paragraph Tag

- The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example:

```
<html>
<head> <title>Paragraph Example</title> </head>
<body>
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
```

</html>

Line Break Tag

 Whenever you use the ***
 element***, anything following it starts from the next line. This tag is an example of an ***empty element***, where you do not need opening and closing tags, as there is nothing to go in between them. The
 tag has a space between the characters br and the forward slash. If you omit this space, older browsers will have trouble rendering the line break. **Example** *<html>*

<head> <title>Line Break Example</title> </head>

<body>

* Hello
*

*You delivered your assignment on time.
*

*Thanks
*

*Mahnaz *

</body>

</html>

Centering Content

 You can use * tag* to put any content in the center of the page or any table cell.

<html>

<head>

<title>Centring Content Example</title> </head>

*<body> This text is not in the center. *

* This text is in the center. *

</body> </html>

Horizontal Lines

- Horizontal lines are used to visually ***break-up sections of a document***.
- The ***<hr> tag*** creates a line from the current position in the document to the right margin and breaks the line accordingly. For example, you may want to give a line between two paragraphs as in the given example below:

<html>

<head> <title>Horizontal Line Example</title> </head>

<body>

* This is paragraph one and should be on top *

```
<hr />
<p>This is paragraph two and should be at bottom</p>
</body>
</html>
```

- Again **<hr /> tag** is an example of the ***empty element***, where you do not need opening and closing tags, as there is nothing to go in between them. The **<hr /> element** has a space between the characters ***hr and the forward slash***. If you omit this space, older browsers will have trouble rendering the horizontal line. *Nonbreaking Spaces*
- In cases, where you do not want the client browser to break text, you should use a nonbreaking space entity *** ***; instead of a normal space.
- For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code:

```
<html>
<head> <title>Nonbreaking Spaces Example</title> </head>
<body>
<p>An example of this technique appears in the movie
"12 Angry Men."</p>
</body> </html>
```

2.7. HTML Links

- HTML links are defined with the **<a> tag**: *link text*. The link's destination is specified in the **href** attribute. **Attributes** are used to provide **additional information about HTML elements**.

Example: -Visit our HTML tutorial

HTML image

- HTML images are defined with the **<img>** tag.
- The source file (src), alternative text (alt), width, and height are provided as attributes.

Example: -<img src="smiley.gif" alt="Smiley face" height="42" width="42"> *HTML Lists*

- HTML lists are defined with the * (unordered/bullet list)* or the * (ordered/numbered list) tag*, followed by * tags* (list items): ***Example***

```
<html>
<head> <title>HTML list Tag</title>  </head>
```

```
<body>
 <h1>this is first group</h1>
 <p>Following is a list of vegetables</p>  <ol>
        <li>Beetroot</li>
        <li>Ginger</li>
        <li>Potato</li>
        <li>Radish</li>
</ol>
<h2>this is second group</h2>
<p >Following is a list of fruits</p>
<ul>
         <li>Apple</li>      <li>Banana</li>
        <li>Mango</li>
        <li>Strawberry</li>
 </ul>
</body> </html> HTML
```

Comments

- Comment is a piece of code which is ignored by any web browser. It is a good practice to add comments into your HTML code, especially in complex documents, to indicate sections of a document, and any other notes to anyone looking at the code. Comments help you and others understand your code and increases code readability.
- HTML comments are placed in between ***<!-- ... --> tags***. So, any content placed with-in ***<!-- ... -->*** tags will be treated as comment and will be completely ignored by the browser. In HTML there are three types of comments those are described as follows:

1. *Single comment* ***Example***

```
<!DOCTYPE html>
<html>
 <head>  <!-- Document Header Starts -->  <title>This is document title</title>
 </head> <!-- Document Header Ends -->
 <body>
   <p>Document content goes here.....</p>
 </body> </html>
```

2. *Multiline Comments*

- So far we have seen single line comments, but HTML supports multi-line comments as well. You can comment multiple lines by the special beginning tag ***<!-- and ending tag ->*** placed before the first line and end of the last line as shown in the given example below. ***Example***

```
<html>
```

```
<head> <title>Multiline Comments</title> </head>
<body>
  <!--
    This is a multiline comment and it can span
through as many as lines you like.
  -->
<p>Document content goes here.....</p>
</body> </html>
```

2.8. Conditional Comments

 Conditional comments only work in Internet Explorer (IE) on Windows but they are ignored by other browsers. They are supported from Explorer 5 onwards, and you can use them to give conditional instructions to different versions of IE. ***Example***

```
<html>
<head>  <title>Conditional Comments</title>
  <!--[if IE 6]>
    Special instructions for IE 6 here
  <![endif]-->
</head>
<body>
<p>Document content goes here.....</p>
</body>  </html>
```

2.9. HTML Tables

 HTML Tables is used to preset data in rows and columns format. A Table is the collection of rows and rows is the collection of columns. <tr> stands for table rows. To add a row in a table ***<table row> tags*** are used. <td> or <th> is used to put the column inside the row. Whereas <td> means table data and <th> means table head.

Syntax:

```
<table>
  <tr>
    <td>row1col1</td>
    <td>row1col2</td>
  </tr>
</table>
```

Example

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Email</th>
```

```
</tr>
<tr>
  <td>phptpoint</td>
  <td>phptpoint@gmail.com</td>
</tr>
<tr>
  <td>phptpoint blog</td>
  <td>blog@gmail.com</td>
</tr> </table>
```

- In the above example a table is created have 3 rows and 6 columns where each row contains 2 column. <tr> tag is used to create a row while <td> or <th> is used to create column. <tr> comes in between <table> tag while <td> or <th> comes in between <tr>.

How to merge table column

- When you want to merge 2 or more than 2 column (cell), use ***colspan*** property of <td colspan="2″> or <th colspan=”2″>. *Example*

```
<table border="1">
<tr>
  <th colspan="2"> User Details</th>
</tr>
<tr>
  <th>Name</th>
  <th>Email</th>
</tr>
<tr>
  <td>phptpoint</td>
  <td>phptpoint@gmail.com</td>
</tr>
<tr>
  <td>phptpoint blog</td>
  <td>blog@gmail.com</td>
</tr> </table>
```

How to merge table rows

- When you want to merge 2 or more than 2 rows in a single row, use ***rowspan*** property of

  <td rowspan=”2″> or <th rowspan=”2″>

- ***Example***

```
<table  border="1">
<tr>
  <td rowspan="2">&amp;nbsp;</td>
  <td>&amp;nbsp;</td>
</tr>
<tr>
  <td>&amp;nbsp;</td>
</tr>
```

```
</table>
```

Nested Table

- Nested table means how to use table inside a table. Multiple times you need to use table inside a table. When you want to use a table inside a table write the syntax of table in between your cell i.e either <td> or <th>.
- ***Example***

```
<html>
<body>
<table border="1" bgcolor="gray" width="200" height="200">
<tr>
<th>
<table align="center" border="1" bgcolor="#F8F8F8" width="100"
height="100">
<tr>
<th>Inner Table</th>
</tr>
</table>
</th>
</tr>
</table> </body> </html>
```

Definition and Usage

- The <table> tag defines an HTML table. An HTML table consists of the <table> element and one or more <tr>, <th>, and <td> elements.
- The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell. A more complex HTML table may also include <caption>, <col>, <colgroup>, <thead>, <tfoot>, and <tbody> elements. *HTML Buttons*
- HTML buttons are defined with the **<button> tag**. The <button> tag defines a ***clickable button***. Inside a <button> element you can put content, like text or images. This is the difference between this element and buttons created with the <input> element.
- ***Example: -***Two button elements that act as one submit button and one reset button (in a form):

```
<form action="/action_page.php" method="get">
        First name: <input type="text" name="fname"><br>
        Last name: <input type="text" name="lname"><br>
 <button type="submit" value="Submit">Submit</button>
 <button type="reset" value="Reset">Reset</button> </form>
```

2.10. HTML Forms

- HTML Forms are required, when you want to collect some data from the site visitor.

- For example, during user registration you would like to collect information such as name, email address, credit card, etc.
- A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc.
- The back-end application will perform required processing on the passed data based on defined business logic inside the application.
- There are various form elements available like text fields, text-area fields, drop-down menus, radio buttons, checkboxes, etc.
- The HTML ***<form> tag*** is used to create an HTML form and it has following syntax: *<form action = "Script URL" method = "GET|POST"> form elements like input, textarea etc.* </form>

Form Attributes

 Apart from common attributes, following is a list of the most frequently used form attributes

S.No	***Attribute***	***Description***
1	*Action*	*Backend script ready to process your passed data.*
2	*Method*	*Method to be used to upload data. The most frequently used are GET and POST methods.*
3	*Target*	*Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.*
4	*Enctype*	*You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are:* ➲ ***application/x-www-form-urlencoded*** *− This is the standard method most forms use in simple scenarios.* ➲ ***mutlipart/form-data*** *− This is used when you want to upload binary data in the form of files like image, word file etc.*

HTML Form Controls

 There are different types of form controls that you can use to collect data using HTML form:

1. Text Input Controls

 There are three types of text input used on forms:

A. ***Single-line text input controls***: is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag. Here is a basic example of a single-line text input used to take first name and last name

```
<head> <title>Text Input Control</title>  </head>
<body>
<form >
First name: <input type = "text" name = "first_name" />     <br>
 Last name: <input type = "text" name = "last_name" /
</form>  </body>  </html> <html>
```

Attributes: -*Following is the list of attributes for <input> tag for creating text field.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for text input control it will be set to text.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*This can be used to provide an initial value inside the control.*
4	*Size*	*Allows specifying the width of the text-input control in terms of characters.*
5	*maxlength*	*Allows specifying the maximum number of characters a user can enter into the text box.*

B. ***Password input controls***: This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag. Here is a basic example of a single-line password input used to take user password:

```
<html>
<head>  <title>Password Input Control</title>  </head>
 <body>
<form >
 User ID : <input type = "text" name = "user_id" />      <br>
 Password: <input type = "password" name = "password" />
 </form>  </body> </html>
```

Attributes: -*Following is the list of attributes for <input> tag for creating password field.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for password input control it will be set to password.*

2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*This can be used to provide an initial value inside the control.*
4	*Size*	*Allows specifying the width of the text-input control in terms of characters.*
5	*maxlength*	*Allows specifying the maximum number of characters a user can enter into the text box.*

C. ***Multi-line text input controls***: This is used when the user is required to give details that may be longer than a single sentence. They are created using HTML <textarea> tag. Here is a basic example of a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head> <title>Multiple-Line Input Control</title>  </head>
<body>
 <form>
     Description : <br />
    <textarea rows = "5" cols = "50" name = "description">          Enter
description here...
   </textarea>
 </form> </body> </html>
```

Attributes: -*Following is the list of attributes for <textarea> tag.*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Rows*	*Indicates the number of rows of text area box.*
3	*Cols*	*Indicates the number of columns of text area box*

2. Checkbox Control

 Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox.Here is an example HTML code for a form with two checkboxes:

```
<html>
<head> <title>Checkbox Control</title> </head>
<body>
<form>
    <input type = "checkbox" name = "maths" value = "on"> Maths
    <input type = "checkbox" name = "physics" value = "on"> Physics
</form> </body> </html>
```

Attributes: -*Following is the list of attributes for <checkbox> tag.*

S.No	***Attribute***	***Description***
1	*Type*	*Indicates the type of input control and for checkbox input control it will be set to checkbox.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*The value that will be used if the checkbox is selected.*
4	*Checked*	*Set to checked if you want to select it by default.*

3. Radio Button Control

- *Radio buttons are used when out of many options; just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio. Here is example HTML code for a form with two radio buttons*

```
<!DOCTYPE html>
<html>
<head> <title>Radio Box Control</title> </head>
<body>
  <form>
    <input type = "radio" name = "subject" value = "maths"> Maths
    <input type = "radio" name = "subject" value = "physics"> Physics
  </form>
</body> </html>
```

Attributes: -*Following is the list of attributes for radio button.*

S.No	***Attribute***	***Description***

1	*Type*	*Indicates the type of input control and for checkbox input control it will be set to radio.*
2	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
3	*Value*	*The value that will be used if the radio box is selected.*
4	*Checked*	*Set to checked if you want to select it by default.*

4. Select Box Control

- *A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options. Here is example HTML code for a form with one drop down box*

```
<html>
<head>  <title>Select Box Control</title>  </head>
<body>
    <form>
        <select name = "dropdown">
            <option value = "Maths" selected>Maths</option>
            <option value = "Physics">Physics</option>  </select>
    </form>
</body> </html>
```

Attributes: -*Following is the list of important attributes of* ***<select>*** *tag*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Size*	*This can be used to present a scrolling list box.*
3	*Multiple*	*If set to "multiple" then allows a user to select multiple items from the menu.*

- *Following is the list of important attributes of* ***<option>*** *tag*

S.No	***Attribute***	***Description***
1	*Value*	*The value that will be used if an option in the select box box is selected.*

2	*Selected*	*Specifies that this option should be the initially selected value when the page loads.*
3	*Label*	*An alternative way of labeling options*

5. File Upload Box

➲ *If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to file. Here is example HTML code for a form with one file upload box*

```
<html>
<head><title>File Upload Box</title></head>
<body>
  <form>
    <input type = "file" name = "file upload" accept = "image/*" />
</form> </body> </html>
```

Attributes: -*Following is the list of important attributes of file upload box*

S.No	***Attribute***	***Description***
1	*Name*	*Used to give a name to the control which is sent to the server to be recognized and get the value.*
2	*Accept*	*Specifies the types of files that the server accepts.*

6. Button Controls

➲ *There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to button. The type attribute can take the following values*

S.No	***Attribute***	***Description***
1	*Submit*	*This creates a button that automatically submits a form.*
2	*Reset*	*This creates a button that automatically resets form controls to their initial values.*
3	*Button*	*This creates a button that is used to trigger a client-side script when the user clicks that button.*
4	*Image*	*This creates a clickable button but we can use an image as background of the button.*

Here is example HTML code for a form with three types of buttons

```
<!DOCTYPE html>
<html>
<head> <title>File Upload Box</title> </head>
<body>
<form>
    <input type = "submit" name = "submit" value = "Submit" />
    <input type = "reset" name = "reset"  value = "Reset" />
    <input type = "button" name = "ok" value = "OK" />
    <input type = "image" name = "imagebutton" src = "/html/images/logo.png" />
 </form>
</body>
</html>
```

7. Hidden Form Controls

- Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page. *Here is example HTML code to show the usage of hidden control*

```
<!DOCTYPE html>
<html>
  <head> <title>File Upload Box</title> </head>
 <body>
    <form>
      <p>This is page 10</p>
      <input type = "hidden" name = "pagename" value = "10" />
      <input type = "submit" name = "submit" value = "Submit" />
      <input type = "reset" name = "reset" value = "Reset" />
    </form>
 </body>
</html>
```

Chapter 3 Styling HTML with CSS

CSS was introduced together with HTML 4, to provide a better way to style HTML elements. Styles were added to HTML 4.0 to solve a problem

3.1. What is CSS?

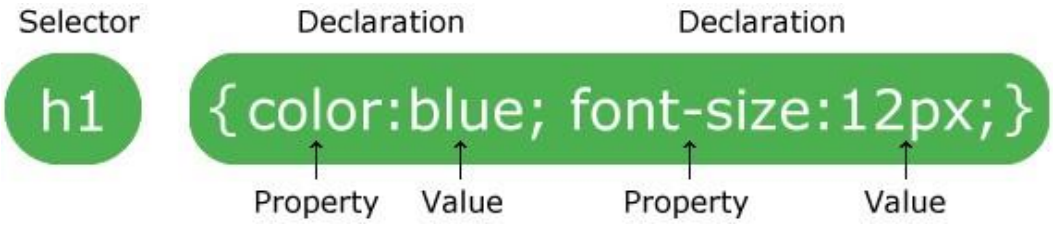
- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- CSS is used to control the style of a web document in a simple and easy way.

3.2. Why Use CSS?

- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Syntax and Selectors CSS Syntax

- A CSS rule-set consists of a selector and a declaration block:



- The **selector** points to the HTML element you want to style.
- The **declaration block** contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example

In this example all <p> elements will be center-aligned, with a red text color: *p {*

color: red;//declaration text-align: center;

}

3.3. CSS Selectors

- CSS selectors are used to "find" (or select) HTML elements based on their element ***name, id, class, attribute***, and more.

The element Selector

- The element selector selects elements based on the element name.

Example

- You can select all <p> elements on a page like this (here, all <p> elements will be centeraligned, with a red text color):

```
p {  text-align: center;
color: red;
}
```

The id Selector

- The id selector uses the ***id attribute*** of an HTML element to select a specific element.
- The id of an element should be unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

The style rule below will be applied to the HTML element with id="para1":

```
#para1 {  text-align:
center;  color: red;
}
```

Note: An id name cannot start with a number!

The class Selector

- The class selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the name of the class.

Example

- In this example all HTML elements with class="center" will be red and center-aligned: `.center { text-align: center; color: red; }`
- You can also specify that only specific HTML elements should be affected by a class. Example
- In this example only <p> elements with class="center" will be center-aligned:

```
p.center {  text-align:
center;  color: red;
}
```

- HTML elements can also refer to more than one class.
- Example
- In this example the <p> element will be styled according to class="center" and to class="large":

- This paragraph refers to two classes. **Note:** A class name cannot start with a number!

Grouping Selectors

- If you have elements with the same style definitions, like this:

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
color: red;
}

p {
  text-align: center;
color: red;
}
```

- It will be better to group the selectors, to minimize the code.
- To group selectors, separate each selector with a comma.
- Example
- In this example we have grouped the selectors from the code above:
- h1, h2, p { text-align: center; color: red;

  }

3.4. CSS Comments

- Comments are used to explain the code, and may help when you edit the source code at a later date.
- Comments are ignored by browsers.

Example

- A CSS comment starts with /* and ends with */. Comments can also span multiple lines:
- p {

  color: red;

  /* This is a single-line comment */ text-align: center;
  }

CSS can be added to HTML in the following ways:

- Inline - using the style **attribute** in HTML elements
- Internal - using the <style> **element** in the <head> section
- External - using an external CSS **file**
- The preferred way to add CSS to HTML is to put CSS syntax in separate CSS files.

3.5. Inline Styles

- An inline style can be used if a ***unique style is to be applied to one single occurrence of an element***.
- To use inline styles, use the style attribute in the relevant tag. The style attribute can contain any CSS property.
- The example below shows how to change the text color and the left margin of a paragraph:

** This is a paragraph. **

HTML Style Example - Background Color

- The background-color property defines the background color for an element: **Example**

```
<!DOCTYPE html>
<html>
<body style="background-color:yellow;">
<h2 style="background-color:red;">This is a heading</h2>
<p style="background-color:green;"> This is a paragraph.</p>
</body>
</html>
```

3.6. Internal Style Sheet

- An internal style sheet can be used if ***one single document has a unique style***. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

```
<head> <style> body {background-
        color:yellow;} p {color:blue;}
        </style>
</head>
```

3.7. External Style Sheet

- An external style sheet is ideal when the style is applied to many pages.
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files
- With an external style sheet, you can change the look of an entire Web site by changing one file.
- Each page must link to the style sheet using the ***<link> tag***. The <link> tag goes inside the <head> section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

</head>

Example mystle.css body { background-color: lightblue;
} h1 { color: navy;
margin-left: 20px;
}

Chapter 4 JavaScript

4.1. Introduction

- JavaScript is a popular web-page scripting language, and is supported by almost every browser.
- It adds interactivity to web technology pages.
- JavaScript is usually embedded directly into HTML pages. *Features of JavaScript*
- It is lightweight, whose implementations allow client-side script to interact with the user and make dynamic pages.
- It is interpreted programming language with object-oriented capabilities.
- Open and cross-platform (machine independent).
- JavaScript is a case-sensitive language. So the identifiers ***Time and TIME*** will convey different meanings in JavaScript.

So, JavaScript is a ***client-side scripting language*** that runs entirely inside the web browser.

4.2. Benefits of JavaScript

Following are some of the benefits that JavaScript language possesses to make the website dynamic.

- It is widely supported in browser.
- It gives easy access to document object and can manipulate most of them.
- It can give interesting animations with many multimedia data types.
- It is secure language because it can detect the visitor's browser.
- It can react to events, read and write HTML elements
- It can be used to validate data

4.3. JavaScript Rules

- JavaScript program contains variables, objects and functions.
- Each line is terminated by a semicolon. Blocks of code must be surrounded by curly brackets.
- Functions have parameters which are passed inside parenthesis.
- Variables are declared using the keyword **var**.
- Script does not require ***main function and exit condition***. *Language Format*
- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the <script> tags, containing your JavaScript, anywhere within you web page, but it is normally *recommended that you should keep it within the <head> tags*.

- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

So, syntax will look as follows:

<script language="javascript" type="text/javascript"> JavaScript code

</script>

The script tag takes two important attributes:

1. ***Language***: This attribute specifies what scripting language you are using. Typically, its value will be **JavaScript**. Although recent versions of HTML have phased out the use of this attribute.
2. ***Type***: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "**text/javascript**".

Take a look at the following code.

```
<html>
<body>
<script language="javascript" type="text/javascript"> document.write
("Hello World!")
</script>
</body>
</html>
```

This code will produce the following result: Hello World!

Description:

- The <script type="text/javascript"> tag tells the browser that whatever comes between that tag and the coming </script> tag is **script**.
- The type="text/javascript" tells it that it is JavaScript.
- The **document.write()** is the JavaScript standard for writing text to the browser window.
- The **'document'** clause refers to the HTML webpage (termed a document) as a whole; what follows ('.write()') is a command for the document object to carry out. *JavaScript variables*
- JavaScript is used for the manipulation of data. The pieces of data can be stored either in variables or arrays.
- JavaScript Variables are declared using the "**var**" keyword. ***Syntax: var num;***
- JavaScript variables can be declared with initial value. ***Syntax: var num=1;***
- Multiple JavaScript variables can even be declared at the same time.
- ***Syntax: var num=1; var name= "Bikila";***

- ➲ Every JavaScript variables ends with semicolon (;).
- ➲ Variables in JavaScript are weakly typed, meaning that the types of individual variables are not determined by the programmer. Unlike many languages, JavaScript only provides a generic "var" rather than separate types for integers, floating point numbers, characters, and strings.

Example: var x = 0; x is a number

4.4. Data types and primitives

 There are six data types in JavaScript:

1. Numbers: – Integer or floating point numbers
2. Booleans: – Either true/false or a number (0 being false) can be used for boolean values
3. Strings: – Sequence of characters enclosed in a set of single or double quotes
4. Objects: – Entities that typically represents elements of a HTML page
5. Null: – No value assigned which is different from 0
6. Undefined: – Is a special value assigned to an identifier after it has been declared but before a value has been assigned to it

4.5. JavaScript Popup Boxes

 JavaScript has three kinds of popup boxes:

1. **Alert Box**: - is used if you want to make sure information comes through to the user. Syntax: **alert("sometext");**
2. **Confirm Box**: - is used if you want the user to ***verify or accept*** something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

   If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

   - Syntax: **confirm("sometext");**

```
if (confirm("Press a button!")) { txt = "You pressed OK!";
} else { txt = "You pressed Cancel!";
}
```

3. **Prompt Box**: - is used if you want the *user to input a value before entering a page*. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

- Syntax: **prompt("sometext","defaultvalue");** *var person = prompt("Please enter your name", "Nahil"); if (person == null || person == "") { txt = "User cancelled the prompt."; } else { txt = "Hello " + person + "! How are you today?"; }*

4.6. Operators in JavaScript

1. Arithmetic Operators

Operator	Description	Example	Result
+	Addition	var x = 2; var y = 2; var sum=x + y;	4
-	Subtraction	var=x = 5; var y = 2; var diff=x – y;	3
*	Multiplication	var x = 5; var y = 4; var prod=x * y;	20
/	Division	var x= 15; var y= 5; var quotient=x/y;	3
%	Modulus	var x= 15; var y= 5; var mod=x%y;	0
++	Increment	var x = 5; x++;	6
--	Decrement	var x = 5; x--;	4

2. Logical Operators

Operator	Description	Example
&&	And	var x = 6; var y = 3; (x < 10 && y > 1) returns true
||	Or	var x = 6; var y = 3; (x == 5 || y == 5) returns false
!	Not	var x = 6; var y = 3; !(x == y) returns true

3. Assignment Operators: used to assign value to variable

Operator	Example	Is The Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

4. Comparison Operators

Operator	Description	Example
==	is equal to	5 == 8 returns false
===	is equal to (checks for both value and type)	var x = 5; var y = "5"; x == y returns true x === y returns false
!=	is not equal	5 != 8 returns true
>	is greater than	5 > 8 returns false
<	is less than	5 < 8 returns true
>=	is greater than or equal to	5 >= 8 returns false
<=	is less than or equal to	5 <= 8 returns true

5. **String Operator: -** is used to concatenate two strings together. Example: var x = "Hello " + "world!"

4.7. Functions in JavaScript

- A function consists of the "function" keyword followed by the name of the function, a set of open and close parentheses enclosing an optional parameter list and a body enclosed in a set of curly braces.

Syntax: function functionName(parameterList) {

// body

}

- Function parameters are separated by commas in the function declaration.
- A function uses the return keyword to return a value from a function.
- JavaScript code found in a function is not executed until the function is called.

Assigns function to event: - many elements of DOM support events. These events are normally the result of some user actions.

Event	Meaning
onload	Occurs when a window or frame has loaded
onunload	Occurs when a document is removed from a window or frame
onclick	The mouse is clicked on an element
ondblclick	The double click event
onmousedown	Mouse down event
onmouseup	Mouse up event
onmouseover	Mouse moves onto an element
onmousemove	Mouse moves over an element
onmouseout	Mouse leaves an element
onfocus	Element receives focus
onblur	Element loses focus
onkeypress	Key press event
onkeydown	Key is pressed down
onkeyup	Key is released
onsubmit	Submit button is pressed
onreset	Form reset event occurs
onselect	Some text in an element is selected
onchange	Element loses focus and its value changes

Example:

```
<html>
<body>
<script type="text/javascript">
function addition(){ var x=7;  var
y=10; var sum=x+y;
alert(sum);} </script>
<form>
<input type="button" onclick="addition()" value="show">
</form> </body></html>
```

4.8. Objects *in JavaScript*

- JavaScript is not a pure object-oriented programming language, but uses the concept of objects. The ***new*** keyword used here is to create an object, it allocates memory and storage.
- Objects can have functions and variables. To differentiate between global variables and those which are part of an object but may have the same name,

- JavaScript uses this keyword. When referring to a property of an object, whether a method or a variable, a dot is placed between the object name and the property.

4.9. Regular Expression:

- A script language may take name data from a user and have to search through the string one character at a time. The usual approach in scripting language is to create a pattern called a regular expression which describes a set of characters that may be present in a string. The regular expression can be shown below.

var pattern = new RegExp("target"); var
string = "can you find the target";
pattern.test(string);

Regular expression is a javascript object. Dynamic patterns are created using the keyword *new*.

regex = new RegExp("feroz | amer");

Example 5: JavaScript code to implement RegExp

```
<html>
<head> <title> registration form</title>
<script type="text/javascript"> function v()
{
var a=document.f.un.value; var
b=document.f.pw.value; var r1=new
RegExp("^[a-zA-Z]{6,}$"); var
r2=new RegExp("^[a-zA-Z0-
9]{6,}$");
if(r1.test(a))
{ if(r2.test(b))
{  window.alert("ur successfully login");
} else window.alert("enter password atleast 6 characters
");
} else { window.alert("enter uname atleast 6 characters and should contain only
alphabets"); }
}
</script> </head>
<body bgcolor="#5555" text="white"onSubmit="return v()">
```

```
<form name="f">
<center>
<table border="0"><br><br><br><br><br><br>
<tr>
<td align="center"> username:</td>
<td align="center"><input type="text" value="" name="un"/></td>
</tr>
<tr>
<td align="center">password:</td>
<td align="center"><input type="password" value="" name="pw"/></td></tr>
<tr> <td align="center" colspan="2"><input type="submit" value="Submit">
<input type="reset" value="reset"/></td></tr>
</table></center></form> </body> </html>
```

4.10. Cookies in JavaScript What are cookies?

- Cookies are data, stored in small text files, on our computer.
- Cookies were invented to remember information about the user.
- JavaScript can create, read, and delete cookies with the **document.cookie** property.
  - With JavaScript, a cookie can be created like: *document.cookie = "username=Ayantu";*
  - We can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed. *document.cookie = "username=Ayantu; expires=Thu, 18 Dec 2019 12:00:00 UTC";*
  - With JavaScript, we can read cookies like:

```
<script>
{
 document.write(document.cookie);
}
</script>
```

  - Deleting a cookie is very simple. Just set the "expires" parameter to a passed date. *document.cookie = "username=Ayantu; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/" ;*
  - **Session:** - A session uniquely identifies and provides continuity for a user's website usage to a better degree than an IP address or cookie could (essentially, by using both together)
  - **Session Cookie:** - A cookie that is deleted when the web browser is closed.

4.11. Using JavaScript on HTML forms

- JavaScript makes HTML pages more dynamic and interactive.
- JavaScript can be used to validate HTML forms.
- Validation can be defined by many different methods, and deployed in many different ways.
  - ***Server side validation*** *is performed by a web server, after input has been sent to the server.*
  - ***Client side validation*** *is performed by a web browser, before input is sent to a web server.*
- JavaScript is used for client side form validation.
- If a form field (fieldname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

```
function validateForm() { var x =
document.forms["myForm"]["fieldname"].value; if (x == "")
{ alert("Name must be filled out"); return false;
}
}
```

The function can be called when the form is submitted.

```
<form name="myForm" action="/action_page.php" onsubmit="return
validateForm()" method="post"> Name: <input type="text"
name="fname">
<input type="submit" value="Submit">
</form>
```

4.12. What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document." *What is the HTML DOM?*

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- *The HTML elements as objects*
- *The properties of all HTML elements*
- *The methods to access all HTML elements*
- *The events for all HTML elements*

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

1. JavaScript HTML DOM Window Object

- The window object represents an open window in a browser.
- If a document contain frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.
- Note: There is no public standard that applies to the Window object, but all major browsers support it.
- Window object ***properties and methods***:
  1. *self :- it returns the current window*
  2. *status:- it sets or returns the text in the status bar of a window*
  3. *top:- it returns the topmost browser window*

2. JavaScript HTML DOM Document:

- The HTML DOM document object is the owner of all other objects in our web page.
- The document object represents our web page. When we want to access any element in an HTML page, we always start with accessing the document object.
- Below are some examples of how we can use the document object to access and manipulate HTML.
  - DOM Document methods for ***finding HTML elements***
    - ✓ *document.getElementById(id):- Find an element by element id.*
    - ✓ *document.getElementsByTagName(name):- Find elements by tag name*
    - ✓ *document.getElementsByClassName(name):- Find elements by class name*
  - DOM Document methods for ***changing HTML elements***
    - ✓ *element.innerHTML = new html content:- Change the inner HTML of an element*
    - ✓ *element.attribute = new value: - Change the attribute value of an HTML element.*
    - ✓ *element.setAttribute(attribute, value):- Change the attribute value of an HTML element*
    - ✓ *element.style.property = new style:- Change the style of an HTML element* □ DOM Document methods for ***Adding and Deleting Elements***
    - ✓ *document.createElement(element):- Create an HTML element*
    - ✓ *document.removeChild(element):- Remove an HTML element*
    - ✓ *document.appendChild(element):- Add an HTML element*
    - ✓ *document.replaceChild(element):- Replace an HTML element*

- ✓ *document.write(text):- Write into the HTML output stream* *JavaScript HTML DOM Elements*

➲ Used to find and access HTML elements in an HTML page. There are a couple of ways to do this:

- ☞ *Finding HTML elements by id*
- ☞ *Finding HTML elements by tag name*
- ☞ *Finding HTML elements by class name*
- ☞ *Finding HTML elements by CSS selectors*
- ☞ *Finding HTML elements by HTML object collections*

➲ The easiest way to find an HTML element in the DOM, is by using the element id.

o example to find the element with id="intro":

- *var myElement = document.getElementById("intro");* o example to find all <p> elements:
- *var x = document.getElementsByTagName("p");*

➲ *To find all HTML elements with the same class name, use getElementsByClassName().*

➲ This example returns a list of all elements with class="intro".

o *var x = document.getElementsByClassName("intro");*

➲ Finding elements by class name does not work in Internet Explorer 8 and earlier versions.

Form methods and properties

➲ Document.forms:- returns an array of HTML FormElement objects listing each of the forms on the page. We can then use any of the following syntaxes to get an individual form:

- ☞ *document.forms[index]:- used to returns the form at the specified index into the array of forms.*
- ☞ *document.forms[id]:- used to returns the form whose ID is id.*
- ☞ *document.forms[name]:- used to returns the form whose name attribute's value is name.*

Chapter 5 Server-side programming

5.1. Introduction to server-side scripting

- Server-side scripting is a method of designing websites so that the process or user request is run on the originating server.
- It is a technique used in **web development** which involves employing scripts on a **web server** which produce a response for each user's request to the website.
- It provides an interface to the user and is used to limit access to proprietary data and help keep control of the script source code.
- Examples: Java, JavaScript using Server-side JavaScript (SSJS), Perl, PHP.
- All of the code is executed on the server before the data is passed to the user's browser. In the case of PHP this means that no PHP code ever reaches the user, it is instead executed and only the information it outputs is sent to the web browser.
- Server-side processing is used to interact with permanent storage like databases or files.
- Server-side processing happens when a page is first requested and when pages are posted back to the server.
- Examples of server-side processing are:
  - ***User validation*****,**
  - ***Saving and retrieving data*****, and** □ ***Navigating to other pages***.

5.2. Differences between Client-side and Server-side Scripting

- The **client** refers to a web browser running on the user's local machine.
- The **server** is the web server software (e.g. Apache) that runs on server hardware.

5.3. Client-Side Scripting

- It is used to run scripts which are usually a browser.
- The processing takes place on the **end users computer**.
- The **source code** is transferred from the web server to the user's computer over the **internet** and run directly in the browser.
- The scripting language needs to be **enabled on the client computer**.

5.4. Server-Side Scripting

- It is used to runs a scripting language in a web server.

- A user's request is fulfilled by running a script directly on the web server to **generate dynamic HTML pages**. This HTML is then sent to the client browser.
- It is usually used to provide **interactive web sites** that interface to databases or other data stores on the server.
- Its advantage is the ability **to highly customize the response** based on the user's requirements, access rights, or queries into data stores.

5.5. What is PHP?

- It is an acronym for "**PHP Hypertext Preprocessor**"
- It is used for creating dynamic and interactive websites.
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension "**.php**"
- It can create, open, read, write, delete, and close files on the server
- It can collect data from form, and it can send and receive cookies
- It can add, delete, modify data in your database and supports a wide range of databases
- It can restrict users to access some pages on your website and it can encrypt data
- It runs on various platforms (Windows, Linux, UNIX, Mac OS X, etc.)
- It is compatible with almost all servers used today (Apache, IIS, etc.)
- It is free. Download it from the official PHP resource: www.php.net
- It is easy to learn and runs efficiently on the server side *To start using PHP, you can:*

- Install a web server on your own PC
- Install PHP
- Install a database, such as MySQL
- Or you can have and install service that comprises all things together such as WAMP, XAMPP.

5.6. Use Basic Syntax

There are three main differences between a standard **HTML document** and a **PHP document.**

- **First**, PHP scripts should be saved with the .php file extension (for example, index.php). Just as a file's extension on your computer tells the operating system in what application to open the file, a Web page's extension tells the server **how to process the file**.
- **Second**, you place PHP code within <?php and ?> tags, normally within the context of some HTML.

Example

```
<html><body><h1>This is HTML.</h1>
<?php PHP code! ?>
<p>More HTML</p>
</body></html>
```

PHP tags indicate the parts of the page to be run through the PHP processor on the server.

- **Third**, PHP scripts must be run on a **PHP-enabled Web server** (whereas HTML pages can be viewed on any computer, directly in a browser). This means that PHP scripts must always be run through a **URL**.

Note

- A PHP script can be placed anywhere in the document.
- PHP statements are terminated by **semicolon** (;).
- The closing tag of a block of PHP code also automatically implies a semicolon (so you do not have to have a semicolon terminating the last line of a PHP block).

Example

```
<!DOCTYPE html>
<html><body><h1>My first PHP page</h1>
<?php echo "Hello
World!";
?>
</body></html>
```

5.7. Sending Data to the Web Browser

1. You'll use PHP most frequently to send information to the browser in the form of **plain text and HTML tags**. To do so, there are two basic ways to get output: **echo and print**.

There are some differences between echo and print:

1. **Echo: -** can output one or more strings
   - It is a language construct, and used with or without parentheses: echo or echo().
   - It is faster compared to print as echo **does not return any value**.

2. **Print:** - can only output one string, and returns always 1
   - ☞ It is can be used with or without parentheses: print or print().
   - ☞ Just type the word print, followed by what you want to display: a simple message, the value of a variable, the result of a calculation, and so forth.
   - ☞ To be clear, **print doesn't actually print anything; it just outputs data**. When a PHP script is run through a Web browser, that PHP output is received by the browser itself.

Example using Echo

```
<?php
echo "<h2>PHP is fun!</h2>"; echo
"Hello world!<br>"; echo "I'm about to
learn PHP!<br>";
echo "This", " string", " was", " made", " with multiple parameters.";
?>
```

Example using Print

```
<?php
print "<h2>PHP is fun!</h2>";
print "Hello world!<br>"; print "I'm
about to learn PHP!";
?>
```

5.8. Comments in PHP

- ➲ Comments are integral to programming, not because they do anything but because they help you remember why you did something.
- ➲ The computer ignores these comments when it processes the script. Furthermore, PHP comments are never sent to the Web browser and therefore remain your secret.

Comments are useful for:

- ☞ **To let others understand what you are doing** - Comments let other programmers understand what you were doing in each step.
- ☞ **To remind yourself what you did** - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did.
  Comments can remind you of what you were thinking when you wrote the code.

PHP supports three ways of commenting:

1. // This is a single line comment
2. # This is also a single line comment
3. /* This is a multiple lines comment block that spans over more than one line */ *PHP Case Sensitivity*

- ☞ In PHP, all user-defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are NOT case-sensitive.
- ☞ However; in PHP, all **variables are case-sensitive**.
- ☞ In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

Example

```
<!DOCTYPE html>
<html><body>
<?php
$color="red"; echo "My car is " . $color .
"<br>"; echo "My house is " . $COLOR .
"<br>"; echo "My boat is " . $coLOR .
"<br>";
?>
</body></html
```

5.9. Utilize Variables

- ➲ A variable is a **container for data**.
- ➲ Once data has been stored in a variable that data can be altered, printed to the Web browser, saved to a database, emailed, and so forth.
- ➲ Variables in PHP are, by their nature, **flexible**: You can put data into a variable, retrieve that data from it (without affecting the value of the variable), put new data in, and continue this cycle as long as necessary.
- ➲ Variables in PHP are **largely temporary**: - they only have a value for the duration of the script's execution on the server. Once the execution passes the final closing PHP tag, those variables cease to exist.

Example

```
<?php
$x=5; $y=6;$z=$x+$y; echo $z;
?>
```

Note

- A variable can have **a short name** (like x and y) or a more **descriptive name** (age, carname, total_volume).

5.10. Rules for PHP variables:

- A variable starts with the **$ sign**, followed by the name of the variable
- A variable name must start with a **letter or the underscore character**
- A variable name cannot start with a **number**
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, _)
- Variable names are **case sensitive** ($y and $Y are two different variables) PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
<?php
$txt="Hello world!";
$x=5;
$y=10.5;
?>
```

After the execution of the statements above, the variable **txt** will hold the value Hello world!, the variable **x** will hold the value 5, and the variable **y** will hold the value 10.5.

Note

- When you assign a text value to a variable, put quotes around the value.
- PHP is a Loosely Typed Language
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it. *PHP Variables Scope*
- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/ used.

5.11. PHP has three different variable scopes:

1. local
2. global

3. static

Local and Global Scope

- A variable **declared outside a function** has a GLOBAL SCOPE and can **only be accessed outside a function**.
- A variable **declared within a function** has a LOCAL SCOPE and can **only be accessed within that function**.

The following example tests variables with local and global scope: **Example**

```
<?php
$x=5; // global scope function myTest() { $y=10; //
local scope echo "<p>Test variables inside the
function:</p>"; echo "Variable x is: $x"; echo
"<br>"; echo "Variable y is: $y"; }
myTest();
echo "<p>Test variables outside the function:</p>";
echo "Variable x is: $x"; echo "<br>";
echo "Variable y is: $y";
?>
```

- In the example above there are two variables $x and $y and a function myTest().
- **$x** is a **global variable** since it is declared outside the function and **$y** is **a local variable** since it is created inside the function.
- When we output the values of the two variables **inside** the myTest() function, it prints the value of $y as it is the locally declared, but cannot print the value of $x since it is created outside the function.
- Then, when we output the values of the two variables **outside** the myTest() function, it prints the value of $x, but cannot print the value of $y since it is a local variable and it is created inside the myTest() function.
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

5.11. PHP the global Keyword

- The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function): **Example** *<?php*

```
$x=5;$y=10; function
myTest() { global $x,$y;
$y=$x+$y; } myTest();
echo $y; // outputs 15
?>
```

- PHP also stores all global variables in an array called $GLOBALS[index].
- The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this: **Example**

```
<?php
$x=5; $y=10; function
myTest() {
$GLOBALS['y']=$GLOBA
LS['x']+$GLOBALS['y']; }
myTest();
echo $y; // outputs 15 ?>
```

5.12. PHP the static Keyword

- Normally, when a function is completed or executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the **static** keyword when you first declare the variable:

Example

```
<?php
function myTest() {
static $x=0; echo $x;
$x++; } myTest();
myTest(); myTest();
?>
```

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called. The variable is still local to the function.

Manipulate Strings

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes: **Example**

```
<?php
$x = "Hello world!";
echo $x; echo "<br>";
$x = 'Hello world!';
echo $x; ?>
```

Concatenating Strings

- It refers to the appending of one item onto another.
- The period (.) is the operator for performing this action, and it's used like so:

```
$s1 = 'Hello, ';
$s2 = 'world!';
$greeting = $s1 . $s2;
```

- The end result of this concatenation is that the $greeting variable has a value of Hello, world!. Because of the way PHP deals with variables, the same effect could be accomplished using $***greeting = "$s1$s2";***.
- This code works because PHP replaces variables within double quotation marks with their value. However, the formal method of using the period to concatenate strings is more commonly used and is recommended (it will be more obvious what's occurring in your code).

5.13. Adjusting String Case

A handful of PHP functions are used to change the case of a string's letters:

- . ucfirst() capitalizes the first letter of the string.
- . ucwords() capitalizes the first letter of words in a string.
- . strtoupper() makes an entire string uppercase.
- . strtolower() makes an entire string lowercase.

Manipulate Numbers

- An integer is a number without decimals.

Rules for integers:

- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based – prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example we will test different numbers. The PHP var_dump() function returns the data type and value of variables: **Example** *<?php*

```
$x = 5985;
var_dump($x); echo
"<br>";
$x = -345; // negative number
var_dump($x); echo "<br>";
$x = 0x8C; // hexadecimal number
var_dump($x); echo "<br>";
$x = 047; // octal number var_dump($x);
?>
```

PHP Floating Point Numbers

- A floating point number is a number with a decimal point or a number in exponential form.
- In the following example we will test different numbers. The PHP var_dump() function returns the data type and value of variables: **Example** *<?php*

```
$x = 10.365;
var_dump($x); echo
"<br>"; $x = 2.4e3;
var_dump($x); echo
"<br>"; $x = 8E-5;
var_dump($x);
?>
```

PHP Booleans

- Booleans can be either TRUE or FALSE. Booleans are often used in conditional testing.

```
$x=true;
$y=false;
```

5.14. Operators in PHP

1. Arithmetic Operators

There are following arithmetic operators supported by PHP language.

- *Addition (+), Subtraction (-), Multiplication (*), Division (/)*

Assume variable **A** holds **10** and variable **B** holds **20** then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A –B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus operator and remainder of after an integer division	B% A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decrease integer value by one	A—will give 9

2. Comparison Operators

There are following comparison operators supported by PHP language. Assume variable A holds 10 and variable B holds 20 the**n**

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A! = B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

3. Logical Operators

There are following logical operators supported by PHP language. Assume variable A holds 10 and variable B holds 20 then

Operator	Description	Example
And	Called Logical AND operator. If both the operands are true then condition becomes true	(A and B) is true.
Or	Called Logical OR Operator. If any of the two operands are non-zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are nonzero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are	(A || B) is true.
	non-zero then condition becomes true.	
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!!(A && B) is false.

4. Assignment Operators

There are following assignment operators supported by PHP language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

5. Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
?:	Conditional Expression	If condition is true? Then value x: otherwise value y

All the operators we have discussed above can be categorized into following categories

1. **Unary prefix operators: -** which precede a single operand.
2. **Binary operators: -** which take two operands and perform a variety of arithmetic and logical operations.
3. **The conditional operator** (a ternary operator): - which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
4. **Assignment operators: -** which assign a value to a variable.

What are constants?

- Constants are like variables except that once they are defined they **cannot be changed** or **undefined**.
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. A constant is **case-sensitive by default**. By convention, constant identifiers are **always uppercase**.

Note: Unlike variables, constants are automatically **global** across the entire script. **Set a PHP Constant**

- To set a constant, use the **define()** function - it takes three parameters:
  - The first parameter defines the **name of the constant**,
  - The second parameter defines the **value of the constant**, and
  - The optional third parameter specifies **whether the constant name should be case-insensitive**. **Default is false**.

The example below creates a **case-sensitive constant**, with the value of "Welcome to Ambo University!": **Example**

```
<?php
define("GREETING", "Welcome to Ambo University!"); echo
GREETING;
?>
```

The example below creates a **case-insensitive constant**, with the value of "Welcome to Ambo University!":

Example

```
<?php
define("GREETING", "Welcome to Ambo University!", true); echo
greeting;
?>
```

- To define a constant you have to use **define**() function and to retrieve the value of a constant, you have to simply specifying its name.
- You can also use the function **constant**() to read a constant's value if you wish to obtain the constant's name dynamically.

constant() function

- As indicated by the name, this function will return the value of the constant.
- This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. it is stored in a variable or returned by a function.

constant() example

```
<?php
define("MINSIZE", 50); echo MINSIZE; echo
constant("MINSIZE"); // same thing as the previous line ?>
```

- Only scalar data (boolean, integer, float and string) can be contained in constants.

Differences between constants and variables are

- There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the **define()** function.
- Constants may be defined and accessed anywhere ***without regard to variable scoping rules***.
- Once the Constants have been set, may not be redefined or undefined. **Valid and invalid constant names**

```
// Valid constant names define("ONE",
"first thing"); define("TWO2", "second
thing"); define("THREE_3", "third
thing"); // Invalid constant names
define("2TWO", "second thing");
define("__THREE__", "third value");
```

Laboratory Exercises and Projects

Prepared by Alemayehu Dereje (alem20@gmail.com)

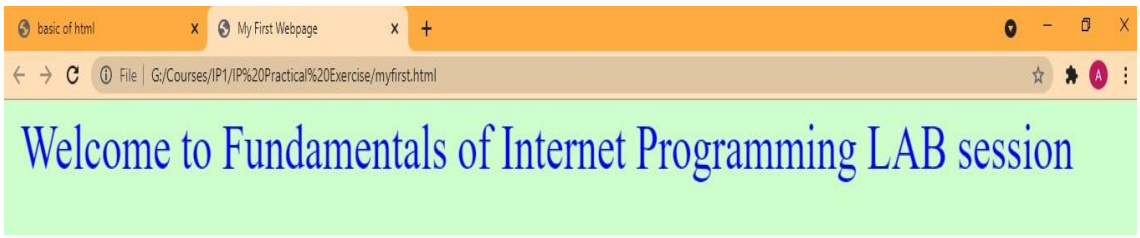
Question 1 **Write an HTML webpage that displays "*Welcome to Fundamentals of Internet Programming LAB session*".**

Note:

Ttitle caption:- My First Webpage Body:- Welcome to Fundamentals of Internet Programming LAB session Bgcolor:- "ccffcc" Text font:- size=22,color=blue Apply marquee style to the text File name:- myfirstweb.html

```
<html>
<hread>
<title> My First Webpage</title>
</head>
<body bgcolor="ccffcc">
<font size=18 color=blue><marquee behavior="alternate">Welcome to Fundamentals of Internet Programming LAB Session</marquee></font>
</body>
</html>
```

Output:



Question 2 An html code that creates a link and insert image into webpage

```
<html>
<hread>
<title> Links and Inserting image</title>
</head>
<body bgcolor="cyan">
<h1> Creating an link</h1>
```

 Click here

<img src="images/Manchester United.gif" alt="My Photo" title="Real Name of the image" width=10% height=15%> </body>

</html>

Output:



Question 3

Headings in HTML

<html>

<head><title> Headings in HTML</title></head>

<body bgcolor="pink">

<h1> Heading 1 </h1>

<h2> Heading 2 </h2>

<h3> Heading 3 </h3>

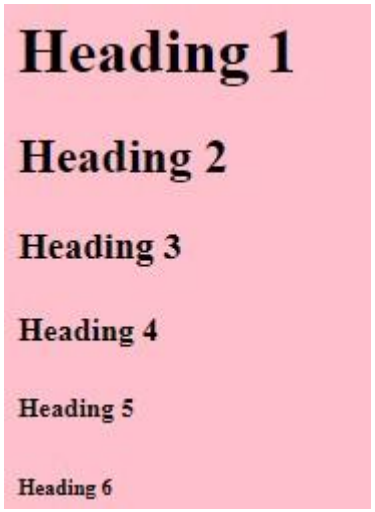
<h4> Heading 4 </h4>

<h5> Heading 5 </h5> <h6> Heading 6 </h6>

</body>

</html>

Output:



Question 4 Creating Table

<html>

<head>

<Title> basic of html</title></head>

<body bgcolor="ffccff" vlink="red">

<table border="2" bordercolor="blue" bgcolor="cyan" align=center Frame=”box” cellpadding="15" cellspacing="5" >

<th colspan=3>Student List</th>

OUTPUT

<tr> <caption> Table 1 List of Student</caption>

<tr>

<td>Name</td>

<td> Sex</td>

<td> Age</td></tr>

<tr> <td> Yonatan Alemayehu</td>

<td> Male</td>

<td> 3</td>

</table>

</body>

</html>



Question 5 Bullets and Numbering

<html>

<head><title>Bullets and Numbering</title></head>

<body>

<ol Type="a" Start="3" >

Introduction

Basic HTML

Databases

<table>

<tr><td>

OL in UpperCase(alphabet) OUTPUT

```
< ol Type="A" Start="3"        >
< li>Introduction</li     >
< li>Basic HTML</li        >
< li>Databases</li     >
< /ol></td >
< td>
OL in Roman Number
< ol Type="I" Start="3"        >
< li>Introduction</li     >
< li>Basic HTML</li        >
< li>Databases</li     >
< /ol></td><td   >
OL in Arabic Number
< ol Type="1" Start="3"        >
<li>Introduction</li>
<li>Basic HTML</li>
<li>Databases</li>
</ol></td>
</table>
Unordered list(circle) </ul>
<Ul Type="circle"> Unordered list(square)
<li>Introduction</li> <Ul Type="square">
<li>Basic HTML</li> <li>Introduction</li>
<li>Databases</li> <li>Basic HTML</li>
</ul> <li>Databases</li>
Unordered list(disk) </ul>
<Ul Type="disc"> </body>
<li>Introduction</li>
<li>Basic HTML</li> </html>
<li>Databases</li>
```

c. Introduction
d. Basic HTML
e. Databases

OL in UpperCase(alphabet) OL in Roman Number OL in Arabic Number

C. Introduction	III. Introduction	3. Introduction
D. Basic HTML	IV. Basic HTML	4. Basic HTML
E. Databases	V. Databases	5. Databases

Unordered list(circle)

- Introduction
- Basic HTML
- Databases

Unordered list(disk)

- Introduction
- Basic HTML
- Databases

Unordered list(square)

- Introduction
- Basic HTML
- Databases

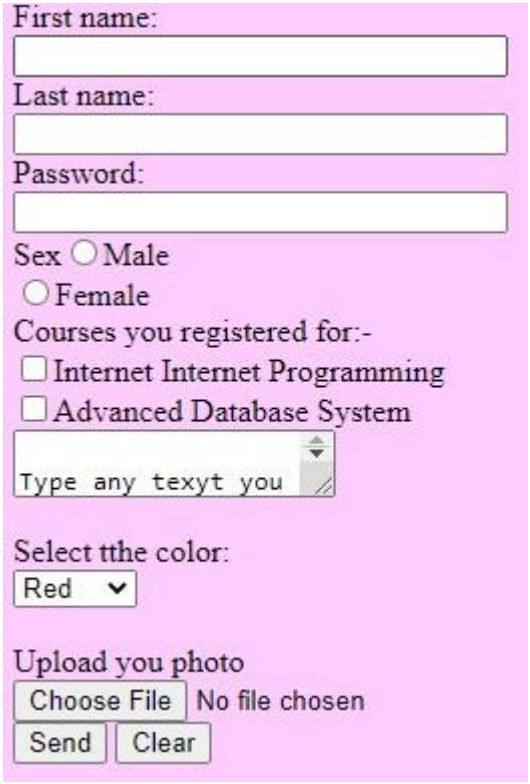
Question 6 Forms and html

```
<html>
<head>
```

```
<title>Using Form</title>
</head>
<body bgcolor=ffccff>
<form name="F1" method="POST" action="">
First name:  <br><input type="text" name="FN" value=""size=30 maxlength=20><br>
Last name:  <br><input type="text" name="lastname" value=""size=30 maxlength=20><br>
Password: <br>  <input type="password" name="password"value=""size=30 maxlength=20><br>
Sex<input type="radio" name="rd" Value="Male">Male<br>
<input type="radio" name="rd" Value="Female">Female<br>
Courses you registered for:-<br>
<input type="checkbox" name="c" Value="IP">Internet Internet Programming<br>
<input type="checkbox" name="c" Value="IP">Advanced Database System<br>
<textarea name="textarea" >

Type any texyt you want
</textarea><br><br>
Select tthe color:<br>
<select name=”color” size =”3”>
<option value=”Red”>Red</option>
<option value=”Cyan”>Cyan</option>
<option value=”Green”>Green</option>
<option value=”Blue”>Blue</option>
        </select><br><br>
        Upload you photo<br>
        <input type="File" name="fu"><br>
<input type="Submit" name="sbt" Value="Send">
<input type="Reset" name="rest" Value="Clear"><br>
</form>
</body>
</html>
```

OUTPUT



Question 7 Inline and internal style

```
<html>
<head><title>Example Inline & Internal Styles</title> <style
type="text/css"> p.floating_para { float:center; width:100%; border:5
dotted red; text-align:center; background-color:silver; margin: 25px;
font-size:32; color:blue; }
</style>
</head>
<body>
<h1 style="text-decoration: underline overline; font-weight: bolder;">
Information Technology Weekend Year II(Inline etyle)
</h1>
<p class=floating_para>This is amazing layout(Internal/embeddeed style) </p>
</body>
</html>
```

OUTPUT



Question 8 Internal stylesheet

```
<html>
<head>
<title> Internal Stylesheets</title>
<style type="text/css"> P{
text-align:center;
color:blue; font-size:28; text-
decoration:overline underline;
}
</style>
</head>
<body>
<p> I'm very happy with internal CSS</p>
```

output



```
</body>
</html>
```

Question 9 Multiple stylesheet

```
<html>
<head>
<title> Internal Stylesheets</title>
<style type="text/css"> P.right{text-
align:right; color:blue; font-
size:32;} P.left{text-align:left;
color:blue; font-size:32;} </style>
</head>
<body>
<p class ="right"> This paragraph is right aligned</p>
<p class="left">This paragraph is left aligned</p>
</body>
</html>
```

OUTPU

This paragraph is right aligned

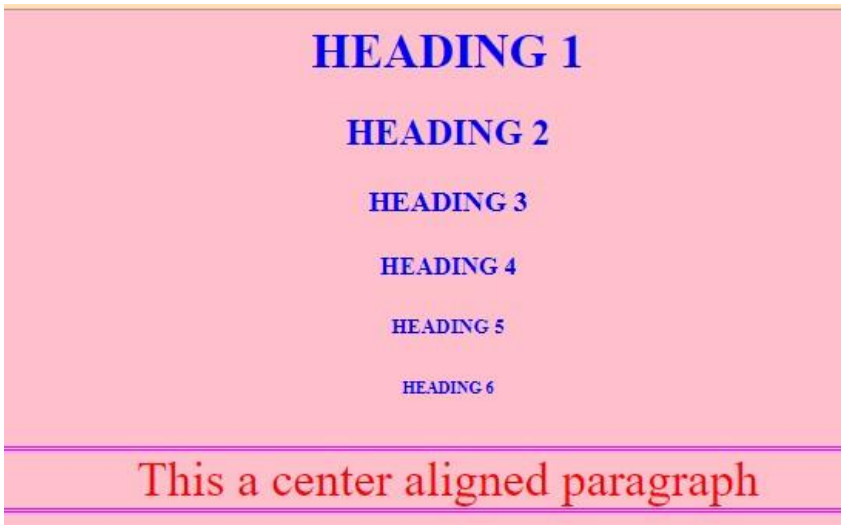
This paragraph is left aligned

Question 10 External Stylesheet

```
<html>
<head>
<title> Style sheets exercises</title>
<link rel="Stylesheet" type="text/CSS" href="externalstyle.css">
</head>
```

OUTPUT

```
<body>
<h1> Heading 1</h1>
<h2> Heading 2</h2>
<h3> Heading 3</h3>
<h4> Heading 4</h4>
<h5> Heading 5</h5>
<h6> Heading 6</h6>
<p>This a  center aligned paragraph</p>
</body>
</html>
```

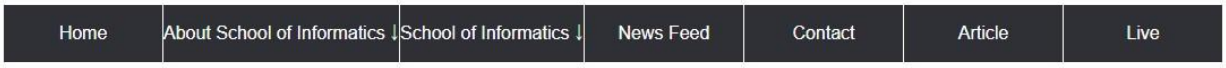


Question 11 Menu

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CSS Only Navigation Menu</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <label for="show-menu" class="show-menu">Show Menu</label>
    <input type="checkbox" id="show-menu" role="button">
        <ul id="menu">
        <li><a href="#">Home</a></li>
        <li>
            <a href="#">About School of Informatics  ↓</a>
            <ul class="hidden">
                <li><a href="#">Who We Are</a></li>
                <li><a href="#">What We Do</a></li>
```

```
        </ul>
    </li>
    <li>
        <a href="#">School of Informatics ↓</a>
        <ul class="hidden">
            <li><a href="4.html">Information Technology</a></li>
            <li><a href="#">Information System</a></li>
            <li><a href="#">Computer Science</a></li>
        </ul>
    </li>
    <li><a href="23.html">News Feed</a></li>
                                                    <li><a
href="#">Contact</a></li>
                                                        <li><a href="#">Article</a></li>
                                                    <li><a href="#">Live</a></li>
  </ul>
</body>
</html>
```

Output





Wolaita So University

School of Informatics

Department of Information Technology

Advanced Internet Programming Module Prepared for Exit Examination

Prepared By:

Mr. Alemayehu Dereje

Email:

Alemayehu.dereje@wsu.edu.et

alem20@gmail.com

Reviewed by: **Nuhamin Nigussie (MSc.) & Biruk Sidamo(MSc.)**

February 2023/2015 E.C

Wolaita, Ethiopia

Contents

Chapter1:- Server Side Scripting Basics in PHP 70
1.1 Overview of Client-Side Script 70
1.2. Introduction to Server-Side Script 70
1.3. PHP Basic Syntax 72
Writing PHP Comments and Output Statements 74
Output Statements: - 75
1.4. Variables and Constants 76
a. PHP Variables 76
 $HTTP_RAW_POST_DATA — Raw POST data 78
Removing Variables 79
b. PHP Constants 81
1.5. PHP Operators (Reading Assignment) 81
The ternary operator 82
1.6. Manipulate Strings 82
Unit 2: PHP Statements and Form Creations 84
2.1. PHP statements: - Use Conditionals/Decision Making, LOOP, Arrays 84
2.1.1. Conditionals/Decision Making Statements 84
 The If...Else Statement 84
 The ElseIf Statement 85
 The Switch Statement 86
While Loop 87
Do…While Loop 87
Php Arrays 89
2.2PHP GET, POST & REQUIST Methods 92
The POST Method 93
The $_REQUEST variable 93
2.3 Creating PHP Forms 94
2.4 Form Validation 96
Chapter 3 Files and Directories in PHP 99
3.1. Reading files/ Directories 99
Creating Directories in PHP 100
3.2. Upload Files 102
3. 3 PHP Cookies and Session 103

Syntax 104
PHP Create/Retrieve a Cookie 104
Example 104
PHP Session 105
Example 105
Example 106
Example 107
3.4 PHP - File Inclusion 107
 The include() Function The require() Function 107
The include() Function 108
The require() function 108
Chapter 4:- PHP Database / PHP with MYSQL 110
Database Queries 110
 Define constants 110
1. Define constants 110
2. Opening Database Connection 111
3. Select database 111
4. Closing database connection 112
Create MySQL Database Using PHP 112
Creating Database Tables 112
Insert Data into MySQL Database 115
Getting Data From MySQL Database 115
Using MYSQL_NUM 116
Deleting Data from MySQL Database 116
Deleting MySQL Database and table Using PHP 117
Updating Data into MySQL Database 117
Background 119
Write a php script that displays “Hello World” 120
I. Some Basics 120
II. Comments 120
III. Variables and Data Types 121
IV. Operators 122
V. Input /Output 122
VI. VI. Control Structures 123
VII. Arrays 125

VIII. Functions 127
XIII. Importing Scripts and HTML Files 133
XIV. Web Form Input 133
XVI. Uploading Files 135

Chapter 1

Server Side Scripting Basics in PHP

1.1 Overview of Client-Side Script

- **Client-side scripting** generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser
- Client-side scripts are often embedded within an HTML or XHTML document (hence known as an "embedded script"), but they may also be contained in a separate file, to which the document (or documents) that use it make reference (hence known as an "external script").
- instructions can be followed without further communication with the server
- By viewing the file that contains the script, users may be able to see its source code. Many web authors learn how to write client-side scripts partly by examining the source code for other authors' scripts.
- The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the user‘s computer over the internet and run directly in the browser.
- The scripting language needs to be enabled on the client computer. Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.
- Includes :-
  - JavaScript, Action Script (used to create animated interactive web applications for Adobe Flash Player using Adobe Flash Pro), VBScript (NOTE: VBScript can also be used as

    Server-side so that processing is done on the server.), Python, html etc

1.2. Introduction to Server-Side Script

Server-side scripting language, which means that the scripts are, executed on the server, the computer where the Web site is located.

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser.

The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. From security point of view, server-side scripts are never visible to the browser as these scripts are executes on the server and emit HTML corresponding to user's input to the page.

In contrast, server-side scripts, written in languages such as PHP, ASP.NET, Java, ColdFusion, Perl, Ruby, Go, Python, and server-side JavaScript, are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts.

Server-side Web scripting is mostly about connecting Web sites to back end servers, such as databases. This enables two-way communication:

- Client to server: Customer-entered information as request.
- Server to client: Web pages can be assembled from back end-server to give output.

Server-side scripting is about "programming" the behavior of the server while client-side scripting is about "programming" the behavior of the browser. Normally, when a browser requests an HTML file, the server returns the file. However, if the file contains a server-side script, the script is executed on the server before the file is returned to the browser as plain HTML.

A server script can do:-

- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the result to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since server code cannot be viewed from a browser

In server side script, since the scripts are executed on the server, the browser that displays the file does not need to support scripting at all. The followings are server-side scripts:

- PHP (*.php)
- Active Server Pages (ASP)
- ANSI C scripts

- Java via JavaServer Pages (*.jsp)
- JavaScript using Server-side JavaScript (*.ssjs)
- Lasso (*.lasso) etc

The main focus here is PHP, which is a server-side scripting language, which can be embedded in HTML or used as a standalone binary, and could be run with open source software like WAMP server.

- ✓ PHP can dynamically create the HTML code that generates the Web page.
- ✓ Web page visitors see the output from scripts, but not the scripts themselves.

1.3. PHP Basic Syntax

PHP stands for PHP: Hypertext Preprocessor. It is a server-side scripting language, which can be embedded in HTML. Over the past few years, PHP and server-side Java have gained momentum, while ASP has lost mindshare.

PHP is a server-side scripting language, which means that the scripts are executed on the server, the computer where the Web site is located. This is different than JavaScript, another popular language for dynamic Web sites. JavaScript is executed by the browser, on the user‘s computer. Thus, JavaScript is a client-side language.

Because PHP scripts execute on the server, PHP can dynamically create the HTML code that generates the Web page, which allows individual users to see customized Web pages. Web page visitors see the output from scripts, but not the scripts themselves.

PHP is particularly strong in its ability to interact with databases. PHP handles connecting to the database and communicating with it, so we don‘t need to know the technical details for connecting to a database or for exchanging messages with it, it is enough telling PHP the name of the database and where it is, and PHP handles the details. It connects to the database, passes our instructions to the database, and returns the database response to us.

Major databases currently supported by PHP include the following:

- ➢ dBASE ,Informix ,Ingres, Microsoft SQL Server ,mSQL ,MySQL ,Oracle ,PostgreSQL,Sybase and etc.

Hence, PHP scripts in the Web site can store data in and retrieve data from any supported database. PHP also can interact with supported databases outside a Web environment. Database use is one of PHP‘s best features.

Use of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- Using php, it is possible to add, delete, and modify elements within our database.
- It helps to assign sessions and cookies for privacy.
- Using PHP, we can restrict users to access some pages of the website.
- It can encrypt data and so mores

The syntax of PHP is: - There are four ways to write php syntax: -

1. **Canonical PHP tags:**
   - The most universally effective PHP tag style is:
     <?php statements written here..?>
   - This is the common one
2. **Short-open (SGML-style) tags**: Short or short-open tags look like this: <? Statements written here…?>
   - Short tags are, as one might expect, the shortest option
   - We must do one of two things to enable PHP to recognize the tags:
     - Choose the --enable-short-tags configuration option when building PHP.
     - Set the short_open_tag setting in php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.
3. **ASP-style tags**: ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASP-style tags look like this: <%.. .%>
   - To use ASP-style tags, we should set the configuration option in your php.ini file.
4. **HTML script tags:** HTML script tags look like this:

<script language=‖php‖>.. . </script>

To work with server script, for example with PHP, we need to install a web server, programming itself, and database server. For PHP, we install PHP as a programming. There are many web servers to choose for PHP uses; the most commonly used web server is called Apache which we can download from the internet freely. Similarly, for database, there are many options to use; the most popular database server for web pages is **MySQL** which again can be downloaded freely from internet.

PHP is also available freely on the internet. In order to develop and run PHP Web pages three vital components need to be installed on computer system.

- **Web Server** - PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- **Database** - PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
- **PHP Parser** - In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

The official PHP website (PHP.net) has installation instructions for
PHP: http://php.net/manual/en/install.php

For the apache code to execute properly, it should be saved in web directory. The web directory depends on what web server is used. For example, for xampp web server, our web directory can be install in C:\xampp\htdocs‖. Hence, we should save PHP files in this folder, www.

For XAMPP server, we save the php code in C:\xampp\htdocs\your file here.

Writing PHP Comments and Output Statements

A comment is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP:

- ✓ **Single-line comments**: They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

  <?php

  #This is a single line comment and

// This also a single line comments too. Each style comments only print "An example with single line comments"; ?>

- ✓ **Multi-lines comments**: They are generally used to provide pseudo code algorithms and more detailed explanations when necessary. Use /* and */ Here are the example of multi lines comments.

```
<?php

/* This is a comment with multiline

……

……


…….

*/

Print "An example with multi line comments";        ?>
```

Output Statements: -

The two most basic constructs for displaying output in PHP are echo and print. Both can be used either with parentheses or without them.

Echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print. The echo or print statement can be used with or without parentheses: echo or echo().

The general format of the echo statement is as follows: echo outputitem1,outputitem2,outputitem3, . . .; echo (output);

The parameterized version of echo does not accept multiple arguments. The general format of the print statement is as follows:

print output; print(output);

- Example: different ways of echo and print echo 123;
  //output: 123 echo ―Hello World!‖; //output: Hello world! echo (―Hello World!‖); //output: Hello

world! echo ―Hello‖,‖World!‖; //output: Hello World!

echo Hello World!; //output: error, string should be enclosed in quotes print (―Hello world!‖); //output: Hello world!

The command print is very similar to echo, with two important differences:

- ✓ Unlike echo, print can accept only one argument.
- ✓ Unlike echo, print returns a value, which represents whether the print statement succeeded.

1.4. Variables and Constants

a. PHP Variables

A variable is a special container that can be defined to hold a value such as number, string, object, array, or a Boolean. The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

For example:-

o $distance = 2;

o $name = “stay home”;

As shown above, Numbers are not enclosed in quotes when they are assigned to variable. However, strings should be enclosed in either single or double quotes (― or _). The quotes tell PHP that the characters are a string, handled by PHP as a unit. Without the quotes, PHP doesn‘t know the characters are a string and won‘t handle them correctly. PHP has a total of eight data types which we use to construct our variables:

1. **Integers**: are whole numbers, without a decimal point, like 4195.

2. **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
3. Booleans: have only two possible values either true or false.
4. **NULL:** is a special type that only has one value: NULL. simply assign it like this: $my_var = NULL; The special constant NULL is capitalized by convention, but actually it is case insensitive. A variable that has been assigned NULL has the following properties:
    - It evaluates to FALSE in a Boolean context.
    - It returns FALSE when tested with IsSet() function.
5. **Strings:** are sequences of characters, like 'PHP supports string operations.'
6. **Arrays:** are named and indexed collections of other values.
7. **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
8. **Resources:** are special variables that hold references to resources external to PHP (such as database connections). The first five are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Strings:

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";

$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php

$variable = "name";

$literally = 'My $variable will not print!\\n';  echo($literally);

$literally = "My $variable will print!\\n";  echo($literally);

?>
```

Find output to the above code.

There are no artificial limits on string length - within the bounds of available memory, we ought to be able to make arbitrarily long strings. Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

 Certain character sequences beginning with backslash (\) are replaced with special characters

 Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

PHP provides a large number of predefined variables to all scripts. The variables represent everything from external variables to built-in environment variables, last error messages to last retrieved headers.

- ✓ **Superglobals** — Superglobals are built-in variables that are always available in all scopes
- ✓ $**GLOBALS** — References all variables available in global scope
- ✓ $**_SERVER** — Server and execution environment information
- ✓ $**_GET** — HTTP GET variables
- ✓ $**_POST** — HTTP POST variables
- ✓ $**_FILES** — HTTP File Upload variables
- ✓ $**_REQUEST** — HTTP Request variables, and can replace $_POST, $_GET and $_COOKIE variables
- ✓ $**_SESSION** — Session variables
- ✓ $**_COOKIE** — HTTP Cookies
- ✓ **$php_errormsg** — The previous error message
- ***$HTTP_RAW_POST_DATA*** *— Raw POST data*
- ✓ **$http_response_header** — HTTP response headers

- ✓ $**argc** — The number of arguments passed to script
- ✓ $**argv** — Array of arguments passed to script

Many of these variables, however, cannot be fully documented as they are dependent upon which server are running, the version and setup of the server, and other factors.

Removing Variables

- We can uncreated the variable by using this statement: **unset(VariableName**);
- After this statement, the variable $age no longer exists. If we try to echo it, you get an ―undefined variable‖ notice. It is possible to unset more than one variable at once, as follows: **unset($age, $name, $address);**

Variable Scope:

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of three scope types:

Local variables:- The variable is only accessible from within the function (or method) that created it A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

```
<?php
$x = 4;
function assignx () {
$x = 0;
echo "\$x inside function is $x. ";
}  assignx();
echo "\$x outside of function is $x. ";
?>
```

The output will be:-

- $x inside function is 0.
- $x outside of function is 4.

Global variables:- The variable is accessible from anywhere in the script. Global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:

```
<?php

$somevar = 15;  function addit()
{

GLOBAL $somevar;  $somevar++ ;
echo "Somevar is $somevar";
 }  addit();
?>
```

The output will be:-

o Somevar is 16

Static variables:- this type of variables be either a global or local variable. Both are created by preceding the variable declaration with the keyword static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

```
<?php
function keep_track() {
 STATIC $count = 0;
$count++;  echo $count;
print " ";
}
 keep_track();  keep_track();
keep_track();
?>
```

This will produce following result. 1 2 3

Variable Naming:

Rules for naming a variable are:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc
- There is no size limit for variables.

b. PHP Constants

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If we have defined a constant, it can never be changed or undefined.

To define a constant we have to use define() function and to retrieve the value of a constant. Unlike with variables, you do not need to have a constant with a $. We can also use the function constant() to read a constant's value if we wish to obtain the constant's name dynamically.

constant () function is used to return the value of the constant. This is useful when we want to retrieve value of a constant, but we do not know its name, i.e. It is stored in a variable or returned by a function. constant () example:

<?php define("MINSIZE", 50); echo MINSIZE; echo‖
‖; echo
constant("MINSIZE"); // same thing as the previous line ?>

Only scalar data (boolean, integer, float and string) can be contained in constants. PHP provides a large number of predefined constants to any script which it runs. There are five magical constants that change depending on where they are used. For example, the value of __LINE__ depends on the line that it's used on in script.

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thusly: [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*

1.5. PHP Operators (Reading Assignment)

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

The ternary operator

One especially useful construct is the ternary conditional operator, which plays a role somewhere between a Boolean operator and a true branching construct. Its job is to take three expressions and use the truth value of the first expression to decide which of the other two expressions to evaluate and return. The syntax looks like:

test-expression ? yes-expression: no-expression;

The value of this expression is the result of yes-expression if test-expression is true; otherwise, it is the same as no-expression. For example, the following expression assigns to $max_num either $first_num or $second_num, whichever is larger: $max_num = $first_num > $second_num ? $first_num : $second_num;

Example:-

<?php

$x=2;

$y=10; If $y%$x>3?‖$y value is more than 3 folds of $x‖:‖$y is not large enough‖; ?>

1.6. Manipulate Strings

PHP has many functions to work with strings. The most commonly used functions for searching and modifying strings are those that use regular expressions to describe the string. The followings are string manipulation operations:-

i. **String concatenation operation: -** To concatenate two string variables together, use the dot (.) operator like echo $string1 . " " . $string2;

ii. **strpos() function**:- used to search for a string or character within a string. If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Written as strops(orginal string, new string) Example: the following code used to show from where the word ―world‖ started.

```
<?php

echo strpos("Hello world!","world");

?>
```

The output will be 6. As seen the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

iii. **The strrev( ) function:-** takes a string and returns a reversed copy of it.

Has a syntax:-

```
$string = strrev(string);  For
example:  echo strrev("study
hard"); drah yduts
```

iv. **The strlen() function**:- is used to find the length of a string. For example:- To find the length of "Hello world!", we can write as follows

```
<?ph

echo strlen("Hello world!");

 ?>
```

(Reading Assignment)

There are many string function in php.

Chapter 2

PHP Statements and Form Creations

2.1. PHP statements: - Use Conditionals/Decision Making, LOOP, Arrays

2.1.1. Conditionals/Decision Making Statements

The if, elseif ...else and switch statements are used to take decision based on the different condition.

- ***The If...Else Statement***

  If you want to execute some code if a condition is true and another code if a condition is false.

  We use this statement if we want to execute a set of code when a condition is true and another if the condition is not true Syntax if (condition) code to be executed if condition is true; else code to be executed if condition is false;

  Example

```
<?php
$x = "Amanuel";
 if ($x == "Dani")
 {
```

```
echo 'Hello Dani!';

 }  else {

echo 'You are not Dani ! are you Amanuel?';  echo
―yes, you are my friend‖;

}

?>
```

- ***The ElseIf Statement***

If you want to execute some code if one of several conditions is true use the elseif statement This statement uses to execute a set of code if one of several condition are true

Syntax if
(condition)

code to be executed if condition is true;

elseif (condition) code to be executed if condition is true; else
code to be executed if condition is false;

If more than one line should be executed in a condition of true/false, the lines should be enclosed within curly braces as shown below.

Example

```
<?php  $d=date("D");  if
($d=="Fri"){  echo "Have a nice
weekend!";
echo ―have good refreshment‖;
}
elseif ($d=="Sun")  echo "Have a
nice Sunday!";  else
echo "Have a nice day!";
?>
```

- ***The Switch Statement***

Avoid long blocks of if..elseif..else code.

Could execute one of many blocks of code / output

The switch statement is used to avoid long blocks of if..elseif..else code. Syntax

switch (expression)

{

case label1: code to be executed if expression = label1; break;

case label2: code to be executed if expression = label2; break;

.

.

. case label n: default: code to be executed if expression is different from both label1 and label n; }

Example:

<?php $name=‖kal‖; switch ($name)

{ case "esayas" :

echo "you are esayas "; break; case "Adane" :

echo "you are adane"; break; case " eyob " :

echo "you are eyob "; case "kal" :

echo "you are a fantastic person"; break; default :

echo "Unknown Person‖;"; break;

} ?>

2.1.2. PHP Loop Types

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following loop types with **continue** and **break** keywords which uses to control the loops execution.

- ✓ **for** - loops through a block of code a specified number of times.
- ✓ **while** - loops through a block of code if and as long as a specified condition is true.
- ✓ **do...while** - loops through a block of code once, and then repeats the loop as long as a special condition is true.
- ✓ **foreach** - loops through a block of code for each element in an array.

While Loop

 The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {   code
to be executed; }
```

The example below first sets a variable $x to 1 ($x=1;). Then, the while loop will continue to run as long as $x is less than, or equal to 5. $x will increase by 1 each time the loop runs ($x++;):

 Example

```
<?php $x=1;
while($x<=5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Do…While Loop

 The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {   code to be executed; }
while (condition is true);
```

The example below first sets a variable $x to 1 ($x=1;). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

 Example

```
<?php $x=1;  do
{
  echo "The number is: $x <br>";
  $x++;
} while ($x<=5); ?>
```

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition fails the first time.

PHP for Loops

 PHP for loops execute a block of code a specified number of times.

Syntax

```
for (init counter; test counter; increment counter) {    code to
be executed;

}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

Example

```
<?php for ($x=0; $x<=10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

The PHP for each Loop

 The for each loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

For each ($*array* as $*value*) { *code*
to be executed; }

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Example

```
<?php
$colors = array("red","green","blue","yellow");  for
each ($colors as $value) {    echo "$value <br>";
}
?>
```

Php Arrays

 An array stores multiple values in one single variable:

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . "."; ?>
```

What is an Array?

An array is a special variable, which can hold more than one value at a time.

Create an Array in PHP

In PHP, the array() function is used to create an array:

- array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

- The index can be assigned automatically (index always starts at 0):

```
$cars=array("Volvo","BMW","Toyota");
```

- or the index can be assigned manually:

```
$cars[0]="Volvo";
$cars[1]="BMW";
$cars[2]="Toyota";
```

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . "."; ?>
```

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars); ?>
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<?php

$cars=array("Volvo","BMW","Toyota");

$arrlength=count($cars);


for($x=0;$x<$arrlength;$x++) {

  echo $cars[$x];   echo
"<br>";

}

?>
```

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:
  $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

Example

```
<?php

$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old."; ?>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php
```

```
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
foreach($age as $x=>$x_value) {   echo "Key=" . $x . ",
Value=" . $x_value;   echo "<br>";

}

?>
```

MULTIDIMENSIONAL ARRAYS (READING ASSIGNEMENTS)

2.2PHP GET, POST & REQUIST Methods

- Usually used when we create forms to make communication between interfaces and databases. **The GET Method:-**

- ✓ Has restriction to send to server/ database parts up to 1024 characters only.
- ✓ GET can't be used to send binary data, like images or word documents, to the server because the GET method sends the encoded user information.
- ✓ The data sent by GET method can be accessed using QUERY_STRING environment variable.
  Never use GET method for systems which have password or other sensitive information to be sent to the server.
- ✓ The $_GET variable is used to collect values from a form with method="get".

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) For example http://localhost/xy.php?name=bekele$age=39

Example:- let we save as xy.php

```
<?php

if( $_GET["name"] || $_GET["age"] )

{

echo "Welcome ". $_GET['name']. "<br />";  echo
"You are ". $_GET['age']. " years old."; exit();  }

?>

<html>
```

```
<body>
 <form action="<?php $_PHP_SELF ?>" method="GET">
Name: <input type="text" name="name" />
 Age: <input type="text" name="age" />
<input type="submit" />
 </form>
</body> </html>
```

The above code's action attribute value can be represented as the file names itself like:-

<form action=‖xy.php‖ method=‖Get‖>

The POST Method

The POST method transfers information via HTTPs headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- ✓ The POST method does not have any restriction on data size to be sent.
- ✓ Relatively secured and could large data in requesting and responding data
- ✓ The POST method can be used to send ASCII as well as binary data.
- ✓ The data sent by POST method goes through HTTP header is secured enough on HTTP protocol. The PHP provides $_POST associative array to access all the sent information using POST method.
- ✓ Variables sent with HTTP POST are not shown in the URL
- ✓ The $_POST variable is used to collect values from a form with method="post".

Information sent from a form with the POST method is invisible to others For example http://localhost/xy.php

Example:- as example, we could change GET by post from example.

The $_REQUEST variable

- ➢ The PHP $_REQUEST variable contains the contents of $_GET, $_POST, and $_COOKIE variables

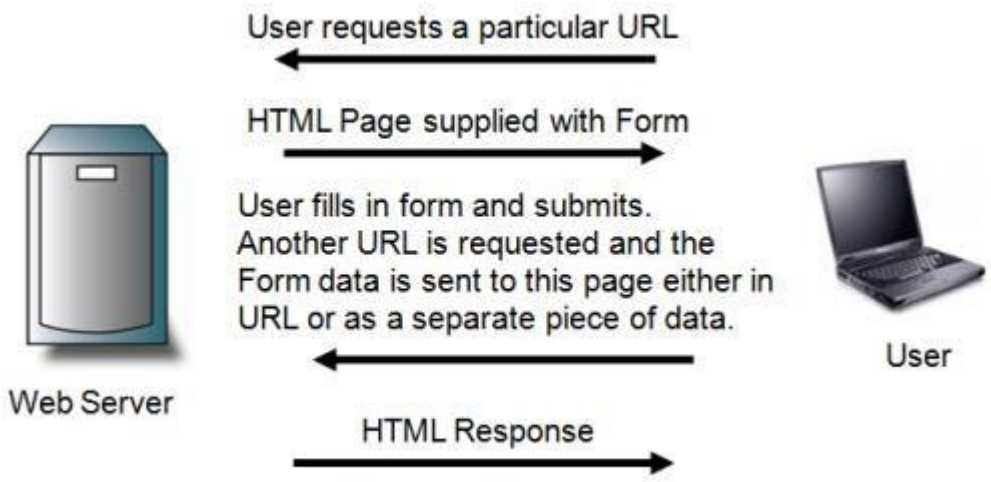
- This variable can be used to get the result from form data sent with both the GET and POST methods.

Example:- <?php

```
if( $_REQUEST["name"] || $_REQUEST["age"] )
 {
 echo "Welcome ". $_REQUEST['name']. "<br />";
 echo "You are ". $_REQUEST['age']. " years old.";  exit(); }
?><html>
<body>
 <form action="<?php $_PHP_SELF ?>" method="POST">
 Name: <input type="text" name="name" />
 Age: <input type="text" name="age" />
 <input type="submit" />
 </form>
</body>
</html>
```

2.3 Creating PHP Forms

- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to PHP scripts. It works as follows:-



- The syntax could be written as -

  <form action=—url to submit the form filled‖ method=—get‖ or —post‖ or —request‖>
  <!–- form contents -->

  </form>

- Where
  - action=—…‖ is the page that the form should submit its data to, and
  - method=—…‖ is the method by which the form data is submitted. If the method is get the data is passed in the url string, if the method is post it is passed as a separate file.

The form variables are available to PHP in the page to which they have been submitted. The variables are available in two superglobal arrays created by PHP called $_POST and $_GET.

The basic concept that is important to understand is that any form element will automatically be available to PHP scripts. See the following example:

```
<form action="action.php" method="post">

Your name: <input type="text" name="name" />

Your age: <input type="text" name="age" />

<input type="submit" />

</form>
```

When the user fills in this form and hits the submit button, the action.php page is called. In this file we would write something like this:

```
<?php echo (—you are
($_POST['name']‖); echo $_POST['age'];
years old.

?>
```

2.4 Form Validation

If a user who forgot to enter one of the fields or enter wrong input, we need to validate the form to make sure it‘s complete and filled out with valid information. We can use JavaScript for this validation. Validations can also be done with simple PHP if statements as shown below.

When the process is done, if it is done making validations, it will check to see if there is an error message. If there is, it displays the error message. If there are no errors, it displays a success message.

```
<Html>

<Body>

<form action="a.php" method="post">

Your Name: <input type="text" name="yourname" /><br />  E-
mail: <input type="text" name="email" /><br/> <p>Do you like
this website?

<input type="radio" name="likeit" value="Yes" checked="checked" /> Yes

<input type="radio" name="likeit" value="No" /> No

<input type="radio" name="likeit" value="Not sure" /> Not sure</p><br/>

<p>Your comments:<br />

<textarea name="comments" rows="10" cols="40"></textarea></p>

<p><input type="submit" value="Send!"></p>

</form>

</body>

</html>
```

Note that:- from the above code,

- All variables passed to the current script via the HTTP POST method are stored in associative array $_POST. For example, in PHP we can access data from each field using $_POST['NAME'], where NAME is the actual field name.

- If we submit the above form, we would have access to a number of $_POST array values inside the a.php file:

In php, we can check the validity of inputs such as URL, E-mail, digits, letters and other special characters etc using functions. For example preg_match() function used to match list of inputs with defined lists. For example, see the following rules:-

i. **URL Address:-** If there is an input field named "website" we can check for a valid URL address like this:

```
$url = htmlspecialchars($_POST['website']);  if
(!preg_match("/^(https?:\/\/+[\w\-]+\.[\w\-]+)/i",$url)) {
die("URL address not valid");  }
```

From the code given above, if the input held by $url is not match with the given string , then the die() function force the system to terminate the running .

ii. **Digits 0-9 only: -** This uses to check whether an input is digit/ number or not.

The following is a syntax to check if $age is a number or not. If not number, it display ―Please enter numbers only for Age‖ .

```
 $age= htmlspecialchars($_POST['age']);  if
(!preg_match("/\D/",$age)) {  die("Please enter
numbers only for Age");

}
```

iii. **Validate e-mail address:-** Used to check an email is valid, i.e to have valid forms.
There is a simple way to check if data entered into input field named "email" is an e-mail address without any unnecessary complications and fancy regular expressions.

```
$email = htmlspecialchars($_POST['email']);  if
(!preg_match("/([\w\-]+\@[\w\-]+\.[\w\-]+)/",$email)) {
die("E-mail address not valid");

}
```

iv. **Letters a-z and A-Z only:-** This code will check if $text is made of letters a-z and A-Z only (no spaces, digits or any other characters): $name = test_input($_POST["name"]); if (preg_match("/[^a-zA-Z]/",$text)) { die("Please enter letters a-z and A-Z only!"); }

Please read Lab tutorials more about the forms and form validations.

Chapter 3

Files and Directories in PHP

3.1. Reading files/ Directories

Files and directories have three levels of access: User, Group and Other. The three typical permissions for files and directories are: Read (r), Write (w) and Execute (x) A stream is a channel used for accessing a resource that we can read from and write to. The input stream reads data from a resource (such as a file) while the output stream writes data to a resource.

Once a file is opened using fopen() function it can be read with a function called fread(). This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes. The file‘s length can be found using the filesize() function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using fopen() function. Syntax:- variable= fopen(—text file‖, —mode‖); where mode mean r, r+, w, w+ etc as shown in the next page.
- Get the file's length using filesize() function. The syntax is filesize($filename);
- Read the file's content using fread() function. Has syntax variable = fread(filename, filesize);
- Close the file with fclose() function. It uses when finished working with a file stream to save space in memory.
  Syntax:- fclose(file name that contains the opened files); $handle
  from the following example.

Example:- Files modes can be specified as one of the six options in this table.

Mode	Descriptions /Purpose
r	Opens the file for reading only. Places the file pointer at the beginning of the file.
r+	Opens the file for reading and writing. Places the file pointer at the beginning of the file.

w	Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
w+	Opens the file for reading and writing only. Places the file pointer at the beginning of the file. And truncates the file to zero length. If files does not exist then it attempts to create a file.
A	Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
a+	Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

Creating Directories in PHP

- A new directory can be created in PHP using the *mkdir()* function.
- This function takes a path to the directory to be created.
- To create a directory in the same directory as your PHP script simply provide the directory name.
- To create a new directory in a different directory specify the full path when calling *mkdir().*

```
<?php

$result = mkdir ("/path/to/directory", "0777");

?>
```

Deleting a Directory

- Directories are deleted in PHP using the ***rmdir()*** function.
- ***rmdir()*** takes a single argument, the name of the directory to be deleted.
- The deletion will only be successful if the directory is empty.
- If the directory contains files or other sub-directories the deletion cannot be performed until those files and sub-directories are also deleted.

Read files in PHP

- The **fread()** function reads from an open file.

- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- Example:- <?php

  $myFile = "sampleFile2.txt";

  $fh = fopen($myFile, 'r');

  $myFileContents = fread($fh,filesize("samplefile2.txt")); fclose($fh);

  echo $myFileContents;

  ?>

Write to File in PHP

- The **fwrite()** function is used to write to a file.
- The first parameter of **fwrite()** contains the name of the file to write to and the second parameter is the string to be written.
- Example:

  <?php

  $myFile2 = "sampleFile2.txt";

  $myFileLink2 = fopen($myFile2, 'a') or die("Can't open file.");

  $newContents = "I am a student"; fwrite($myFileLink2,
  $newContents); fclose($myFileLink2);

  ?>

Open File in PHP

- A better method to open files is with the **fopen()** function.
- The first parameter of **fopen()** contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

3.2. Upload Files

- Web applications allow visitors to upload files to and from their local computer. The files that are uploaded and downloaded may be simple text files or more complex file types, such as images, documents, or spreadsheets
- Files are uploaded through an HTML form using the ―post‖ method and enctype attribute with value of ―multipart/form-data,‖ which instructs the browser to post multiple sections – one for regular form data and one for the file contents.
- The file input field creates a browser button for the user to navigate to the appropriate file to upload <form method=‖post‖ action=‖ ‖ enctype= multipart/form-data >
  <input type="file" name="picture_file" /> </form>
- The MAX_FILE_SIZE (uppercase) attribute of a hidden form field specifies the maximum number of bytes allowed in the uploaded file and it must appear before the file input field.
- When the form is posted, information for the uploaded file is stored in the $_FILES auto global array.
- Example:-The following HTM code below creates an uploaded form. This form is having method attribute set to post and enctype attribute is set to multipart/form-data
  <html> <body>

  <h3>File Upload:</h3>

  Select a file to upload:

  <form action="<?php $_PHP_SELF ?>" method="post"

  enctype="multipart/form-data">

  <input type="file" name="file" size="50" />

  <input type="submit" value="Upload File" />

  </form> <?php

  if($_FILES['file']['name'] != ""){

```
 copy( $_FILES['file']['name'], "C:\wamp\www\Leture\Test.php" ) or die( "Could not
copy file!");

} else{

 die("No file specified!");}

?>

<html>

<head>

<title>Uploading Complete</title>

</head>

<body>

<h2>Uploaded File Info:</h2>

<ul>

<li>Sent file: <?php echo $_FILES['file']['name']; ?>

<li>File size: <?php echo $_FILES['file']['size']; ?> bytes

<li>File type: <?php echo $_FILES['file']['type']; ?>

</ul>

</body>

</html>

</body></html>
```

3. 3 PHP Cookies and Session

What is a Cookie?

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

- A cookie is created with the setcookie() function.

Syntax

- setcookie(name, value, expire, path, domain, secure, httponly);
- Only the name parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

- The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).
- We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Note: The setcookie() function must appear BEFORE the <html> tag.

Example

```
<?php

$cookie_name = "user"; $cookie_value =
"John Doe";

setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day ?>

<html>

<body>

<?php

if(!isset($_COOKIE[$cookie_name])) {

   echo "Cookie named '" . $cookie_name . "' is not set!";
```

```
} else {     echo "Cookie '" . $cookie_name . "' is
set!<br>";     echo "Value is: " .
$_COOKIE[$cookie_name];

}

?>

</body>

</html>
```

PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- Session variables hold information about one single user, and are available to all pages in one application.
- If you need a permanent storage, you may want to store the data in a database.
- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.

Example

```
<?php

// Start the session session_start();

?>

<!DOCTYPE html>

<html>

<body>


<?php

// Set session variables
```

```
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat"; echo
"Session variables are set.";
?>
</body>
</html>
```

- **Note:** The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page echo
"Favorite color is " . $_SESSION["favcolor"] . ".<br>"; echo
"Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php

session_start();

?>

<!DOCTYPE html>

<html>

<body>

<?php

// remove all session variables
session_unset();  // destroy the
session  session_destroy();

?>

</body>

</html>
```

3.4 PHP - File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.

- ***The include() Function***
- ***The require() Function***

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

The include() Function

- ✓ The include() function takes all the text in a specified file and copies it into the file that uses the include function.
- ✓ If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.
- ✓ Assume you want to create a common menu for your website. Then create a file menu.php with the following content.

```
<a href="http://www.tutorialspoint.com">Home</a>

<a href="http://www.w3schools.com">w3schools</a>

<a href="http://www.ajax.com">AJAX</a>

 <a href="http://www.mysql.com">MySQL</a> <br />
```

Now create as many pages as you like and include this file to create header. For example now your **test.php** file can have following content.

```
<html>

<body>  <?php

include("menu.php");
?>

<p>This is an example to show how to include PHP file!</p>

 </body>

</html>
```

The require() function

- ✓ The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **require()** function generates a fatal error and halt the execution of the script.
- ✓ So there is no difference in require() and include() except they handle error conditions.

- ✓ It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.
- ✓ You can try using above example with require() function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

```
<html>
<body>  <?php
include("xxmenu.php");
?>
<p>This is an example to show how to include wrong PHP file!</p>  </body>
</html>
```

Output:- This is an example to show how to include wrong PHP file!

```
<html>
<body>
   <?php
     require("xxmenu.php");
   ?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
</html>
```

Output:-This time file execution halts and nothing is displayed.

NOTE:-You may get plain warning messages or fatal error messages or nothing at all. This depends on your PHP Server configuration.

Chapter 4:- PHP Database / PHP with MYSQL

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Database Queries

- ✓ A query is a question or a request.
- ✓ We can query a database for specific information and have a recordset returned.
- ✓ Example: SELECT LastName FROM Employees

Connecting to databases

To connect PHP with database, four important things must be taken place. Those are:-

 Define constants

- ✓ **Create connection using mysql_connect.**
- ✓ **Select database.**
- ✓ **Close connection.**

1. Define constants

To connect php with database, defining constants is very important. Constants that must be defined are:-

- ✓ *Define the server.*
- ✓ *Define user name of the server.*
- ✓ *Define password of the user.* *Define database name.*

To define the constant, we can use as follows:

<?php **define**("db_server","localhost");// the server that we use
define("db_user","root");// the username of the server
define("db_pass","");//password of the server
define("db_name","schoolmgt");//the database that we use

?>

2. Opening Database Connection

After defining constants using php, opening or creating connection is very important. To open or create database connection, we use **mysql_connect** function**.**

This function takes three parameters. Those are:

 db_server,db_user,db_pass.

- ✓ Db_server:-The host name running database server
- ✓ Db_user:-The username accessing the database
- ✓ Db_pass:-The password of the user accessing the database.

From the above the host name running database server is ―localhost‖, the username accessing the database is ―user‖, and the password of the user accessing the database is empty. Connection can opened or created as follows:

$connection=mysql_connect(db_server,db_user,db_pass);

3. Select database

Once you establish a connection with a database server then it is required to select a particular database where your all the tables are associated.

This is required because there may be multiple databases residing on a single server and you can do work with a single database at a time.

To select database, we use **mysql_select_db** as follows**:-**

$db_select=mysql_select_db(db_name,$connection);

4. Closing database connection

Its simplest function **mysql_close** PHP provides to close a database connection. This function takes connection resource returned by **mysql_connect** function. For example, to close the connection that you use in the above; you use **mysql_close** function as follows:- **mysql_close($connection);**

Create MySQL Database Using PHP

To create and delete a database you should have admin privilege. Its very easy to create a new MySQL database. PHP uses **mysql_query** function to create a MySQL database. For example to create the database test_db using php, you can write as follows:-

<?php **define**("db_server","localhost");
define("db_user","root");
define("db_pass","");

$connection**=mysql_connect**(db_server,db_user,db_pass); **if**(!$connection)

{ **die**("error connection to db server".**mysql_error**());

} **echo —**Connected successfully‖;

$sql = **—**CREATE Database test_db‖; $retval =
mysql_query($sql, $connection);

if(! $retval) { **die**('Could not create database: ' .
mysql_error());

} **echo** "Database test_db created successfully\n";
mysql_close $connection);

?>

Creating Database Tables

To create tables in the new database you need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using **mysql_query**() function.

```
<?php define("db_server","localhost");
define("db_user","root");
define("db_pass","");
define("db_name","test_db");

$con=mysql_connect(db_server,db_user,db_pass);
if(!$con){ die("not connected".mysql_query());

}

$sqldb=mysql_select_db(db_name,$con); if
(!$sqldb){ die("incorrectly
selected".mysql_error());

}

$sql = 'CREATE TABLE employee( '.

    'emp_id INT NOT NULL AUTO_INCREMENT, '.

    'emp_name VARCHAR(20) NOT NULL, '.

    'emp_address  VARCHAR(20) NOT NULL, '.

    'emp_salary   INT NOT NULL, '.

    'join_date    timestamp(14) NOT NULL, '.

    'primary key ( emp_id ))'; $retval =
mysql_query($sql,$con);

  if(!$retval ) {       die('Could not create table: ' .
mysql_error());

  }

  echo "Table employee created successfully\n";

?>
```

In case you need to create many tables then its better to create a text file first and put all the SQL commands in that text file and then load that file into $sql variable and execute those commands. Consider the following content in **sql_query.txt** file.

```
CREATE TABLE employee

(emp_id INT NOT NULL AUTO_INCREMENT,
emp_name VARCHAR(20) NOT NULL,
emp_address VARCHAR(20) NOT NULL,
emp_salary INT NOT NULL,  join_date
timestamp(14) NOT NULL,  primary key ( emp_id ));
```

```
 <?php define("db_server","localhost"); define("db_user","root"); define("db_pass","");
define("db_name","test_db"); $con=mysql_connect(db_server,db_user,db_pass); if(!$con){ die("not
connected".mysql_query());

}

$sqldb=mysql_select_db(db_name,$con); if
(!$sqldb){ die("incorrectly
selected".mysql_error());

}

$query_file = 'sql_query.txt';

$fp = fopen($query_file, 'r'); $sql =
fread($fp, filesize($query_file)); fclose($fp);

$retval = mysql_query($sql,$con);

  if(!$retval ) {

    die('Could not create table: '.mysql_error());

  }

  echo "Table employee created successfully\n";
?>
```

Insert Data into MySQL Database

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function **mysql_query**. Below a simple example to insert a record into **employee** table. sql = 'INSERT INTO employee '.

'(emp_name,emp_address, emp_salary, join_date) '.

'VALUES ("guest", "XYZ", 2000, NOW())';

Getting Data From MySQL Database

Data can be fetched from MySQL tables by executing SQL **SELECT** statement through PHP function mysql_query. You have several options to fetch data from MySQL.

The most frequently used option is to use function **mysql_fetch_array()**. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows. next is a simple example to fetch records from **employee** table.

```
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';

$retval = mysql_query($sql,$con);

  if(!$retval ) {       die('Could not create table: ' .
mysql_error());

  }

  while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
echo "EMP ID :{$row['emp_id']}  <br> ".

       "EMP NAME : {$row['emp_name']} <br> ".

       "EMP SALARY : {$row['emp_salary']} <br> ".

       "--------------------------------<br>";

  }
```

The content of the rows are assigned to the variable $row and the values in row are then printed.Always remember to put curly brackets when you want to insert an array value directly into a string. In above

example the constant **MYSQL_ASSOC** is used as the second argument to **mysql_fetch_array**(), so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index.

PHP provides another function called **mysql_fetch_assoc()** which also returns the row as an associative array.

Using mysql_fetch_assoc()

```
while($row = mysql_fetch_assoc($retval)) {       echo "EMP ID :{$row['emp_id']}  <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "--------------------------------<br>";
  }
```

Using MYSQL_NUM

```
while($row = mysql_fetch_array($retval, MYSQL_NUM)) {       echo "EMP ID :{$row[0]}  <br> ".
        "EMP NAME : {$row[1]} <br> ".
        "EMP SALARY : {$row[2]} <br> ".
        "--------------------------------<br>";
  }
```

Deleting Data from MySQL Database

Data can be deleted from MySQL tables by executing SQL **DELETE** statement through PHP function **mysql_query**.

Below is a simple example to delete records into **employee** table. To delete a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

```
$emp_id = $_POST['emp_id'];
```

$sql = "DELETE FROM employee WHERE emp_id= $emp_id" ;

Deleting MySQL Database and table Using PHP

If a database is no longer required then it can be deleted forever. You can use pass an SQL command to mysql_query to delete a database.

$sql = 'DROP DATABASE test_db‘; ---□deleting MySQL database

$sql = 'DROP TABLE employee‘; ------□ deleting table

Updating Data into MySQL Database

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function **mysql_query**.

Below is a simple example to update records into **employee** table. To update a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

$emp_id=$_POST['emp_id‘];

$emp_salary = $_POST['emp_salary‘];

$sql = "UPDATE employee ". "SET emp_salary = $emp_salary ". "WHERE emp_id = $emp_id" ;

Wolaita Sodo University

School of Informatics

Department of Information Technology

Advanced Internet Programming php Laboratory Manual

Prepared By Alemayehu Dereje

The manual covers

- A simple php program
- Variables and their syntax in php
- Built-in methods
- Arrays
- Strings
- Post & Get Method
- Conditions and Looping structure
-

Introduction

The following is a quick introduction and summary of many aspects of the PHP language for those who have some programming ex perience. Although this overview is not intended to be an exhaustive examination of PHP, it is comprehensive enough for you to get started building non -trivial web applications with PHP. All syntax contained in this guide is for PHP5 and may not be compatible with previous versions of PHP.

Some Basics 2
II. Comments 3
III. Variables and Data Types 3
IV. Operators 4
V. Input/Output 4
VI. Control Structures 5
VII. Arrays 5
VIII. Functions 7
IX. Strings 8
X. Regular Expressions 8
XI. Exception Handling 9
XII. File I/O 10
XIII. Importing Scripts and HTML Files 11
XIV. Web Form Input 11
XV. Maintaining State ..... 12 XVI. Uploading Files 13

Background

- PHP is a recursive acronym: PHP: Hypertext Preprocessor (originally named Personal Home Page)
- Invented by Rasmus Lerdorf in 1994 and is now under the Apache Software Foundation. Licensed under the GPL and is free.
- Popular server-side technology for Apache web servers. Competing technologies include Oracle‟s Java Server Pages, Microsoft‟s ASP.NET, and Adobe‟s ColdFusion.
- Available on a variety of web servers (Apache, IIS, NGINX, etc.) and operating systems (Windows, Linux, UNIX, Mac OS X, etc.).
- Supports many types of databases: MySQL, Oracle, ODBC (for MS Access and SQL Server), SQLite, etc.

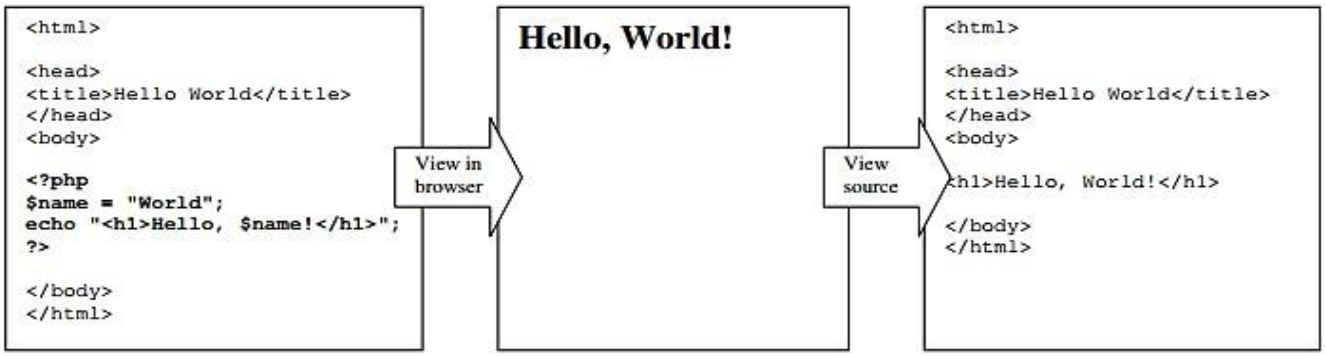
On-line Resources

- www.php.net – PHP distribution, tutorials, newsgroups, and more.
- www.phpfreaks.com - PHP and MySQL tutorials, scripts, forums, and more.
- www.phpbuilder.com – Collection of PHP resources.

Write a php script that displays "Hello World"

If your web server supports PHP, type this example into a text file called hello.php and access it in your browser by typing the complete URL (e.g., http://local host/hello.php). Depending on how your web server is configured, your .php file will need the proper permissions so the web server can access and execute the PHP scrip.

```
<html>
<head> <title>Hello World</title></head>
<body> <?php
echo "<h1>Hello, World!</h1>";
?>
</body> </html>
```



I. Some Basics

A. PHP is a *scripting* language – it gets *interpreted* instead of being *compiled* like C++ and Java.

B. Unlike JavaScript which is executed by the *web browser*, all PHP code is executed on the *web server*.

C. The syntax is very similar to Perl and C. Variables are case sensitive, function names are not, and statements must be terminated with a semicolon.

D. PHP code should be placed between <? code ?> or <?php code ?> tags. The second method is preferred so your scripts are XML compatible. There is no limitation as to where PHP code can be inserted.

E. To see information about how PHP is configured, version information, and the settings of all environment variables (e.g., HTTP_USER_AGENT and QUERY_STRING), call the **phpinfo()** function in any script.

F. The **php.ini** file is the main configuration file for PHP. It can be edited by the system administrator to change any of the configuration settings. A change to this file requires the web server be restarted since the file is only read once when the web server starts up. (The phpinfo() function reports the location of php.ini on the server.)

II. Comments

The three following styles are legal:

Perl style single line comment

// Single line comment

/* Multiple line

comments */

III. Variables and Data Types

A. Always starts with $ and letter or underscore. Can be composed of numbers, underscores, and letters.

$my_var = 10;

$a_2nd_var = "bison";

B. Data types: integers, doubles (numbers with a decimal point), boolean (true or false), NULL, strings, arrays, objects, and resources (like database connections). Variables do not have to be declared and neither do their data types.

C. Variables have a default value (0, empty string, false, or empty array) if they aren‟t initialized before trying to use them. It‟s always good practice to initialize all variables rather than relying on the default initialization value. If you try to use a variable before setting it to a value, strict errorreporting setting will give you an “Undefined variable” warning.

D. All variables have local scope (i.e., they are accessible only within the function or block in which they are initialized). Global variables may only be accessed within a function by using the global keyword.

$x = "This variable‟s scope is global";
function display() { **global** $x; echo $x; }

E. Constants are defined using **define** and by convention are usually named in ALL CAPITALS.

define("PI", 3.14); **define**("HEADING", "<h1>My
Web Site</h1>");
$area = PI * $radius * $radius; print(HEADING);

//Complete code

<html>

<head><title>This is global variable sample</title></head>

<body gbvolor="pink">

<?php

$x = "This is a global variable";
function display() { global
$x;//global variable echo $x;
}//end of function display

display();//called function display

//**Constants** are defined using define and by convention are usually named in ALL CAPITALS

```
echo "<hr  width=50% size=10 align=left color=cyan>";
define("PI", 3.14); $radius=2; define("HEADING", "<h1>My
Web Site</h1>");
$area = PI * $radius * $radius; print(HEADING."when radius= $radius
Area of circle =".$area);
echo "<hr width=50% size=10 align=left color=cyan>";//inserts a  horizontal line
```

IV. Operators

A. **Assignment**

1. = += -= /= *= %= ++ -- => like most programming languages.
2. .= => string concatenation operator (see strings section).

B. **Arithmetic**

1. + - * / % => like most programming languages.

C. **Comparison**

1. == != < > <= >= => like most programming languages. Also <> is the same as !=.
2. = = = => true if arguments are equal and the same data type.
3. != = - true if arguments are not equal or they are not of the same data type.

D. **Logical**

1. && || ! - like most programming languages (&& and || short-circuit)
2. and or - like && and || but have lower precedence than && and ||.
3. xor - true if either (but not both) of its arguments are true.

V. Input /Output

A. **print** and **echo** are used to print to the browser.

```
echo "Go Bisons"; echo("Go
Bisons"); // same thing print("Go
Bisons"); // same thing
```

B. **print** can only accept one argument, and echo can accept any number of arguments. print returns a value that indicates if the print statement succeeded.

C. Variables are interpolated inside of strings unless single quotes are used. $a = 5; echo "You have $a Messages."; // prints "You have 5 Messages." echo 'You have $a Messages.'; // prints "You have $a Messages."

D. Escape sequences: \n (newline), \r (carriage-return), \t (tab), \$ ($), \" (“), \\ (\) echo "a\\b\tc\$d"; // prints "a\b c$d" echo 'a\\b\tc\$d'; // prints "a\b\tc\$d". Only \\ is converted.

E. printf works like C‟s counter-part.

$title = "X-Men"; $amount = 54.235; **printf**("The movie %s made %2.2f million.", $title, $amount); // prints "The movie X-Men made 54.23 million.

VI. VI. Control Structures

A. Choice structures

1. **if** ($x > 0)

$y = 5; // {} not required for only one statement

2. **if** ($a) { // tests if $a is true or non-zero or a non-empty string print($b);
$b++;
}
else
print($c);

3. **if** ($a > $b) print "a is bigger than b";
elseif ($a == $b) // use "elseif" or "else if" print "a is equal to b";
else
print "a is smaller than b";

4. **switch** ($vehicle_type) { // works for integers, floats, or strings

case "car": $car++; break; **case** "truck": $truck++; break; **case** "suv": $suv++; break; **default**: $other++;

}

//if else statement

```
<?php  $score=60; if($score>=50)
            echo("Score ==>$score you are passed");
else
        echo("you are failed"); ?>
```

B. Looping structures

1. **while** ($n < 10) { print("$n ");
   $n++;
   }

2. **do** {
   print("$n "); $n++;
   } **while** ($n < 10);

//A looping structure sample

3. **for** ($n = 1; $n < 10; $n++) print("$n ");
4. **foreach** ($myarray **as** $item) print("$item ");

```
<?php
        echo("<h1>Looping structures</h1>");
        echo "<hr width=50% size=10 align=left color=frushca>while loop==>";
         $n=0; while
        ($n < 10) {
        print("$n ");
        $n
        +
        +;
        }
        echo "<hr width=50% size=10 align=left color=frushca>Do while loop==>";
        $n=0;
        do {
        print("
        $n ");
        $n++;
        } while ($n < 10);
        echo "<hr width=50% size=10 align=left color=frushca>for  loop==>"; for
        ($n = 1; $n < 10; $n++)
        print("$n ");?>
```

Viewed on browser (output look like)

Looping structures

while loop==>0 1 2 3 4 5 6 7 8 9

Do while loop==>0 1 2 3 4 5 6 7 8 9

for loop==>1 2 3 4 5 6 7 8 9

VII. Arrays

A. Summary of all array functions in the PHP core: http://www.php.net/manual/en/ref.array.php

B. Arrays can have any size and contain any type of value. No danger of going beyond array bounds.

$my_array[0] = 25;

$my_array[1] = "Bisons";

C. PHP arrays are **associative arrays** which allow element values to be stored in relation to a key value rather than a strict linear index order.

$capitals["CO"] = "Denver";

$capitals["AR"] = "Little Rock";

D. Initialize an array:

$colors = **array**("red", "green", "blue");

print("The 2nd color is $colors[1]."); // prints

green

$capitals = **array**("CO" => "Denver", "AR" => "Little Rock"); print("$capitals[CO]");

// prints Denver, no quotes around key inside ""

E. Print contents of an array for debugging:

```
print_r($colors);
```

produces:

```
Array
(
    [0] => red
    [1] => green
    [2] => blue
)
```

```
print_r($capitals);
```

produces:

```
Array
(
    [CO] => Denver
    [AR] => Little Rock
)
```

F. Pull values out of an array:

```
$colors = array("red", "green", "blue");
list($c1, $c2) = $colors;
print("$c1 and $c2");   // prints "red and green"
```

F. **Delete from an array:**

unset($colors[1]); // $colors now contains red and blue at indexes 0 and 2. **H.**

Extracting array keys and values:

$states = **array_keys**($capitals); // $states is ("CO", "AR")

$cities = **array_values**($capitals); // $cities is ("Denver", "Little Rock") **I.**

Iterating through an array:

$heroes = array('Spider-Man', 'Hulk',

'Wolverine'); **foreach** ($heroes as $name)

print("$name
"); // prints all three in order

foreach ($capitals as $state => $city)

print("$city is the capital of $state.
"); **J.**

Treat an array like a stack:

array_push($heroes, 'Iron Man'); // Pushed onto end of array

$heroes[] = 'Captain America'; // Same thing as array_push

$h = **array_pop**($heroes); // Pops off last element (Iron

Man) K. Size of an array:

$num_items = **count**($heroes); // returns 3

Complete code

<?php

$colors=array('red','green','yellow','pink','cyan');

print_r($colors);//used to print arrays with their respective index

?> output Array ([0] => red [1] => green [2] => yellow [3] => pink [4] => cyan)

if you want to display the array value at index 2,you may write as echo $colors[2], which displays yellow

- **You can add additional array elements**
  - To push blue , then type as **$colors[]='blue'** , adds blue at the last index o **array_push**($colors, „purple'); // Pushed onto end of array
- **To print the length of the array uses count($colors);** o **echo ("
Array Length=".count($colors));// returns the length of the array**
- **Using for loop to display the array elements** o for($i=0;$i<count($colors);$i++)

  {echo "
 at index $i = ".$colors[$i]."
 ";}

The complete array Code

<?php

$colors=array('red','green','yellow','pink','cyan');

print_r($colors);

$colors[]='blue';// append at the end of the array element

or array_push($colors, 'purple'); // Pushed onto end of

**array echo "
";**

print_r($colors);// displays all array elements

**echo ("
Array Length=".count($colors));// returns the length of the array**

for($i=0;$i<count($colors);$i++)

```
{echo "<br> at index $i = ".$colors[$i]."<br> ";     }
```

```
Array ( [0] => red [1] => green [2] => yellow [3] => pink [4] => cyan )
Array ( [0] => red [1] => green [2] => yellow [3] => pink [4] => cyan [5] => blue [6] => purple
Array Length=7
at index 0 = red

at index 1 = green

at index 2 = yellow

at index 3 = pink

at index 4 = cyan

at index 5 = blue

at index 6 = purple
```

?>

output

```
 <?php
//a loop tha adds the elements of an array
$numbers=array('12','50','70','100');
print_r($numbers);
for($num=0;$num<count($numbers);$num++)//adds/sum elements of arrays
{global $sum;
$sum=$sum+$numbers[$num];
}print "<br> Sum of the array elements are=$sum";
?>
```

```
Array ( [0] => 12 [1] => 50 [2] => 70 [3] => 100 )
Sum of the array elements are=232
```

VIII. Functions

A. PHP pre-defined functions are documented at http://www.php.net/manual/en/funcref.php .

B. Functions may be declared anywhere in the source code (i.e., they do not need to be defined before they are called as C++ requires).

C. Function names are case-insensitive, though it is usually good form to call functions as they appear in their declaration.

D. Defining and calling 1. General form: **function** func_name($param_1, $param_2, ..., $param_n) {
// code
return $retval; // optional: can return a scalar or an array
}
2. Call: $result = func_name($arg1, $arg2, ..., $argn);

E. Parameter passing and returning values

1. Arguments may be passed by **value** (default) or by reference (using &). Default argument values can also be used which must be initialized in the parameter list. Variable-length argument lists are also supported but are not covered here.

```
// Pass by value
function sum($a,
$b) { return $a +
$b;
}
<?php // Pass by reference
function swap(&$a, &$b) {
$a=300;$b=400;
$temp = $a;
$a = $b; $b =
$temp; echo
$a."<br>".$b;
}echo
swap($a,$b);?>
```

```
<?php // Default arguments must be on right side

function say_greeting($name, $greeting="Hello") {

print "$greeting, $name!";

} say_greeting("Yonatan"); // Hello, Yonatan!

say_greeting("Temu", "Welcome"); // Temu,

Welcome!

?>
```

2. Passing an array by value and by reference

```
// Pass by value function
sum_array($values) {
$sum = 0;
foreach ($values as $num)
$sum += $num; return
$sum;
}
$nums = array(1, 2, 3); print "Sum of array
= " . sum_array($nums); // 6 // Pass by
reference function randomize(&$nums) {
for ($i = 0; $i < 10; $i++) $nums[$i] =
rand(0, 100); // 0-100
}
$n = array();
randomize($n); // Place 10 random nums in $n
Complete code
<?php
function randomize(&$nums) { for ($i = 0;
$i < 10; $i++) $nums[$i] = rand(0, 100); //
0-100
}
$n = array();
randomize($n); // Place 10 random nums in $n
print_r($n); ?>
```

IX. Strings

A. Concatenation

$full_name = $first_name . " " . $last_name; // results in "Yona Alemayehu"

B. Some PHP string functions. View the complete list at http://www.php.net/manual/en/ref.strings.php int **strlen**($str) Returns string length.

```
<?php

$FirstName="Yonatan";

$LastName="Alemayehu";

//concatnation echo $FirstName." ".$LastName;

//returns length of the string echo "<br> Length of the

first name is =".strlen($FirstName); echo "<hr>";

$str1=$FirstName; $str2=$LastName; echo

strrev($str1)."<br> "; // Reverse string echo

str_repeat($str2, 3)." <br>"; // Repeat string

echo strtoupper($str1)."<br>"; // String to

uppercase echo strtolower($str1);//return back

to small letters echo "<hr>";


?>
```

IX. Regular Expressions

A. Regular expressions are patterns that can be used to match text in a string. They can be used, for example, to determine if a string contains a legal email address or phone number. PHP regular expressions are implemented very similarly in other programming languages. For a complete reference, see http://us2.php.net/manual/en/ref.pcre.php.

B. The examples here use Perl regular expressions which require forward slashes (“/”) around the pattern.

C. **Matching patterns**

1. Find the given pattern anywhere in the string

```
if (preg_match("/ard/", "Harding")) echo
"Matches"; else

echo "No match";
```

2. Special symbols

\d \s \w	any digit (0-9) any white space (space, tab, EOL) any word char (a-z, A-Z, 0-9, _)
.	any character except EOL
[abc]	a, b, or c
[^a-z]	not any char between a and z

{3} ? * + ^abc abc$	match only three of these match zero or one character match zero or more characters match one or more characters match at the beginning of the string match at the end of the string

3. Email address example

$email = 'first_name.last_name@domain.Com';

$regexp = "/^[\w.]+@[\w.]+[a-z]{2,4}$/i"; // i switch for case-insensitive match

if (preg_match($regexp, $email)) echo "Match email";

4. Remembering matched patterns

if (**preg_match**('/(\d\d):(\d\d) (am|pm)/', '03:15 pm', **$matches**))

{ echo "Hour: $matches[1]\n"; // 03 echo "Min: $matches[2]\n";

// 15 echo "Ending: $matches[3]\n"; // pm

}

5. Match all occurrences preg_match_all('/.ar/', 'the car was far from the bar', $matches);

print_r($matches[0]); // Prints car, far, bar

D. Replacing patterns

1. **Simple replacement**

$new = **preg_replace**('/hard/', 'easy', 'this is hard!'); // Returns "this is easy!"

2. **Replacement with remembered matches**

// Convert MM/DD/YYYY to YYYY-MM-DD (must escape / in regex)

$date = **preg_replace**("/(\d\d)\/(\d\d)\/(\d\d\d\d)/", "$3-$1-$2", "08/15/2008");

E. Array processing

1. Split string into an array

$names = 'Fred, Erin, Alex, Sunshine';

$names_array = **preg_split**('/[,]+/', $names); // Returns one name in each slot

2. Find all items in an array that match a regex

// Returns only Erin and Alex

$starts_with_vowel = **preg_grep**('/^[aeiou]/i', $names_array);

Complete code

<?php //Split string into an array

```
$names = 'Fred, Erin, Alex, Sunshine';

$names_array = preg_split('/[, ]+/', $names); // Returns one name in each slot print_r(

$names_array);

// Returns only Erin and Alex

$starts_with_vowel = preg_grep('/^[aeiou]/i', $names_array); print_r($starts_with_vowel);

?>
```

XI. Exception Handling

A. PHP uses exception (error) handling much like C++, Java, and many other HLLs. function divide($x, $y) { if ($y == 0) **throw new Exception**('Division by zero.'); else return $x / $y; }**try** { echo divide(5, 2) . "\n"; // Prints 2.5 echo divide(4, 0) . "\n"; // Causes exception to be thrown

}

catch (Exception $e) { echo 'Caught exception:

', $e->getMessage(), "\n";

}

B. Exceptions that are not caught cause the script to fail with a fatal error.

C. Other information about the exception like the line number and the file name in which the exception occurred is also available. See http://us.php.net/manual/en/language.exceptions.php.

XII. File I/O

A. PHP can access any file that is stored on the web server, as long as it has the proper permissions.

B. HTTP, FTP, and STD **read/write** can also be used with file functions.

C. See http://www.php.net/manual/en/ref.filesystem.php for functions that set file permissions, copy and delete files, access file modification times, and a lot more. D. Open file with fopen, close with fclose. File open **modes**:

1. "r" – Read from existing file.
2. "r+" – Read and write to already existing file.
3. "w" – Write to file, creating the file if it doesn"t already exist, erasing it if it does.
4. "w+" – Write and read, creating the file if it doesn"t already exist, erasing it if it does.
5. "a" – Append to end of file whether it exists or not.
6. "a+" – Append to end of file, doubling file contents if you read the file in as a string, edit it, and write it back to the file.

E. Reading from a file. File must have proper read permissions. If file is not owned by "nobody", it must have world read permissions.

1. Read entire file:

```
$fd = fopen("myfile.txt", "r") or die("Can't open myfile.txt for reading.");
$entire_file = fread($fd,
filesize("myfile.txt")); print $entire_file;
fclose($fd);
```

*The die operator is useful for halting executing of a PHP script and giving an error message. You may also call exit which does the same thing.

2. Read line by line:

```
while (!feof($fd)) {
$line = fgets($fd, 4096); // 4096 is the max bytes per line
print "$line<br />";
}
```

Complete code//Reading a file

```
<?php
$fd = fopen("myfile.txt", "r") or die("Can't open myfile.txt for reading.");
$entire_file = fread($fd,
filesize("myfile.txt")); print $entire_file;
fclose($fd);
?>
```

Reading a file line by line

```
<?php
$fd = fopen("myfile.txt", "r") or die("Can't open myfile.txt for reading."); while
(!feof($fd)) {
$line = fgets($fd, 4096); // 4096 is the max bytes per line
print "$line<br />";
}?>
```

F. Writing to a file. The directory containing the file to be written must have at least world execute and write permissions. Default owner of a created file will be "nobody".

```
<?php
$fd = fopen("myfile.txt", "w") or die("Can't write to myfile.txt.");
fwrite($fd, "This is output."); fclose($fd);
?>
```

XIII. Importing Scripts and HTML Files

A. A file containing HTML and/or PHP code can be imported into another PHP script by using the **require** statement. If the file being included cannot be found, the script halts with a fatal error. heading.php

<h3>Date: <?php $today = date("D M d Y"); echo $today; ?></h3>

The code above can be used in a PHP script like this (assuming it resides in the same directory):

<?php **require** 'heading.php'; // Prints Date: Fri Aug 29 2008 ?>

<h3>Date: <?php $today = date("D M d Y"); echo $today; ?></h3>

<?php

require 'string.php'; // imported into this webpage,you can also use include instead of require **?>**

B. The **require_once** and **include_once** statements do the same thing as require and include, but they will not reload a file that has already been included.

XIV. Web Form Input

A. Data from web forms can be accessed using the superglobal arrays **$_GET** and **$_POST**.

B. **$_GET**: for accessing data in the **query string** – the key=value pairs that appear in the URL after the "?" character. All values are unescaped (+ is converted to space, etc.).

Example: http://www.example.com/myscript.php?**custname=Bob+Smith&custage=21**

$name = **$_GET**["custname"]; // $name is "Bob Smith"

$age = **$_GET**["custage"]; // $age is 21

C. **$_POST**: for accessing posted for data from **standard input** (values are unescaped).

Example STDIN: custname=Bob+Smith&custage=21

$name = **$_POST**["custname"]; // $name is "Bob Smith"

$age = **$_POST**["custage"]; // $age is 21

Complete code named as webform.html

```
<html>
<head>
<title>webform</title></head>
<body bgcolor=ffccff>
<form action="retrieve.php" method='POST' name="F1">
First Name:<br><input type="text" name="FN" value=""><br>
Last Name:<br><input type="text" name="LN" value=""><br>
E-mail:<br><input type="text" name="email" value=""><br>
<input type="submit" name="sbt" value="Submit">
<input type="reset" name="rst" value="Reset">
```

```
</form>
</body>
```

Php code saved as retrieve.php

```
</html>
<?php
$FN=$_POST['FN']; $Lname=$_POST['LN']; echo"Your
First Name is $FN <br> Father Name $Lname";
echo("<br>Your Email is ".$_POST['email']); ?>
```

D. It‟s always a good idea to use **isset** to check if the variable exists in $_POST and $_GET before accessing to avoid PHP warnings:

```
if (!isset($_POST["custname"]) || trim($_POST["custname"]) == "") echo
"The customer's name was left blank.";
```

XV. Maintaining State

A. To keep track of data between HTTP requests, data can be stored in cookies using the **$_COOKIE** array,or it can be stored on the web server via session variables in the **$_SESSION** array.

B. setcookie() and session_start() functions below *must* be called before any other output is produced unless output_buffering is turned on in php.ini.

C. **$_COOKIE** – for accessing HTTP cookies which are stored on the client and are transmitted back to the web server in every HTTP request.

1. By default, cookies expire with the session. Closing the browser (not just the tab) ends the session.

```
setcookie("age", "21");
<?php  //complete code to set and access cookie
setcookie('age', '21'); // to access cookies echo "The
value of the cookie is ".$_COOKIE['age'];
?>
```

2. Cookies can be given expiration dates so they persist even after the browser is closed:

```
<?php
setcookie('age', '21'); // to access cookies echo "The value of the
cookie is ".$_COOKIE['age'];echo "<br>";
// Cookie expires in 24 hours
setcookie('name', 'Betty', time() + 60 * 60 * 24);
```

// to access cookies echo "\nThe value of the cookie is
".$_COOKIE['name']; ?>

3. **Getting the value of a cookie:** echo **$_COOKIE**["name"]; // Print contents of name cookie Note that $_COOKIE will not have values set from setcookie() until the php script is requested *after* a call to setcookie. **4. Deleting a cookie: setcookie**("name", FALSE);

5. *Warning:* Since cookies are stored on the client, they should not be used to store sensitive data.

D. **$_SESSION** – for accessing session variables which are stored on the web server. Variables are associated with a unique session ID which is stored in a cookie or passed in the URL if cookies are disabled. This technique is ideal for storing sensitive data since the data is not stored on the client.

1. Create a session (and a session ID) if one doesn‟t already exist:
session_start();

2. If you are curious what the session ID looks like: echo **session_id**();

3. Set a session variable that is stored on the web server and tied to the user‟s session ID:
$_SESSION["name"] = "Betty";

4. Get the value of a session variable that is tied to the user‟s session ID:
$auto = **$_SESSION**["name"];

5. Session variables are cleared when the browser is closed or when destroying the session explicitly:
session_destroy();

XVI. Uploading Files

A. Files may be uploaded to the web server from the browser by using a specially-encoded form and the **$_FILES** superglobal array.

```
<form method="POST" action="upload.php" enctype="multipart/form-data">
<!-- max files size in bytes -->
<input type="hidden" name="MAX_FILE_SIZE" value="5000000" />
<input type="file" name="myfile" /><br />
<input type="submit" value="Submit" />
</form>
```

When submitted to upload.php, the **$_FILES** array contains an entry for each file uploaded.
print_r(**$_FILES**); **produces**

```
Array
    ([myfile] => Array( ) )
```

[name] => archive-3.png	Name of file on client
[type] => image/png	File‟s MIME type as determined by file extension
[tmp_name] => /tmp/phpUajj5s	Temporary path of file stored on the server
[error] => 0	Number indicating an error (0 means no error)
[size] => 131377	Size of file in bytes

You must copy the file from the temporary location ($_FILES['myfile']['tmp_name']) to a more permanent location if you wish to store the file contents for later use. A complete function that does this and more can be obtained from http://frankmccown.blogspot.com/2009/04/upload-image-inphp.html.

```
//A complete php script to upload images
<?php
$name = $_FILES['fileToUpload']['name'];
$temp_name=$_FILES['fileToUpload']['tmp_name'
]; echo $name."<br/>"; echo $temp_name;
$location='uploads/';
/*The move_uploaded_file() function moves an uploaded file to a new location*/
move_uploaded_file($temp_name,$location.$name); echo "the image is
successfully uploaded";
?>
```



Wolaita Sodo University

School of Informatics

Department of Information Technology

Mobile Application Development –National Exit Exam Module

Prepared By: Biruk S. (MSc.)

Reviewed By: Nuhamin Nigussie(MSc.)

Alemayehu Dereje(MSc.)

WSU, Ethiopia

March, 2023

Chapter One

1. Introduction

After Completed this session, students able to understand:

- Define all about mobile application development
- Explain about Major Mobile Development Platforms
- Understand how to install IDE android studio
- Explain Mobile Computing

Every day the new devices are incoming to the market with innovative options thanks to growing technology. The evolution of mobile application development technology with new devices made our lives much easier.

In the smartphone world, simply having a running website is not enough. A recent study, it has shown that about 45% and more of Google search happens using smartphones.

The number is spectacular and there is growth within the mobile business. Being obtainable on an internet-enabled device is needed for every and each business which has given the kicking start to mobile application development.

We can start explaining mobile development, which is not about building phone apps, though it is a huge part of it. Actually, it’s doing any reasonable development for any kind of mobile device such as developing apps for phones, tablets, smartwatches, and every form of wearable device that run any kind of mobile operating system.

Mobile development presents a reasonably distinctive chance for a one-person development team to build an actual, usable, significant app end-to-end during a comparatively short period. However, mobile app development represents more than just a chance for the solo developer to create their own project as it is arguably the longer term of development, as mobile devices are getting larger and bigger parts of our lives.

Major Mobile Development Platforms

iOS

iOS is kind of arguably the “big dog” once it comes to major app development platforms, partially because it had been the platform that finally brought mobile development into the modern day and age by fully reworking the concept of a mobile device and mobile software system. iOS is of course developed by Apple, and it runs exclusively on Apple products.

Apple provides iOS developers with many native tools and libraries to develop iOS applications, and, though you do not have to be enforced to use Apple’s development tools to create your apps, you just need to have a mac running OS X to build your application.

Android

Android is the other dominant player in this space, it was a bit later participant to the game, first being released in Sept 2008, virtually a year later than iOS but it has managed to achieve a reasonably massive share of the mobile market.

Technically, Android is the mobile OS with the largest most dominant share of the market with around 80% share compared to iOS’s 18 % share. Those numbers are a bit deceiving since android may be a fragmented market consisting of the many different devices created by different manufacturers, running completely different versions of the Android OS.

What are the primary differences between iOS and Android?

- Android is backed by Google.
- iOS is backed by Apple.
- Anyone can build an Android device, and it is designed to run on a variety of different hardware platforms and devices with very different form factors and capabilities.
- iOS is designed to run only on a specific set of Apple devices.
- Android is based on the Linux kernel, and Google releases the source code for Android as open source.
- Like Apple, Google provides some native tools for Android development but again you are not required to use them.

Top mobile application development trends in 2018:

Augmented Reality

Augmented reality (AR) is used to explain a true world surrounding wherever computer-generated elements are familiar enhance what we are able to see through a camera and in recent years, this technology has been booming with popularity.

Not solely is AR ofttimes used in amusement apps, however, has additionally been distinguished in education and in many varieties of sporting events. For example, in common social media apps nowadays like Snapchat, Instagram and Pokémon Go.

AR has helped give a recent and interactive expertise by enriching a clear image with 3D animations, fancy text, and funky filters by differentiating themselves from the other apps offered on the market, this technological feature has helped them stand to move into a unique manner and appeal to users of all ages particularly to the younger generation who are more updated with the most recent technologies.

Artificial Intelligence

Artificial intelligence (AI) has been featured in several technological advancements as well as machine learning, language process, automation and much more. With the fast and continuous growth of AI, big-name corporations have been adding this feature to their mobile apps with notable success like Amazon's Alexa and Apple's face recognition software system.

AI create apps more intuitive and it also helps deliver an improved user experience as all of your info is funneled through algorithms in order to form suggestions tailored to your interests and placement. If your company is looking to be at the head of innovation, take into account incorporating AI into your mobile app or concern being left behind.

Hybrid Development

In hybrid development, standard coding languages like JavaScript are used and only one code base must be written, as the utilized framework will neatly wrap everything along in a

very view, which may then be displayed in their respective native Android and iOS environments.

Over the past few years, there has been an incredible increase in corporations willing to form hybrid mobile apps because of the cross-platform capabilities offered by hybrid development and the way apps may be launched onto the Google Play Store and Apple AppStore with negligible resources.

On the other hand, native app development is extremely costly, as they will only be designed for one platform at a time but corporations who attempt to deliver the simplest user experience with raw speed can forever prefer them. Popular mobile development frameworks like React-Native and Ionic are leading this race and be attention-grabbing to see which one can dominate the market within the near future or if a new contestant will boost up their game and be willing to compete.

Two-Factor Authentication

With Two-Factor Authentication (2FA), users are prompted to verify a login attempt was meant by getting into a pin through their phone or email that is connected to their account. In modern society, our data is our identity and with over five billion distinctive mobile phone users with an increase of four-dimensional annually, data security is essential and more necessary than ever.

Several internet and mobile applications like Gmail and Steam are using 2FA to verify a user's identity throughout login or whenever suspicious activities occur on their account like an IP address modification. With data breaches and account phishing attempts occurring daily in our modern society, this additional layer of protection can modify corporations to preclude cyber-crimes and offer tighter security.

Blockchain

With blockchain evolving at a speedy pace, several firms and startups are incorporating this technology into their mobile applications creating what is familiar these days as DApps(decentralized applications).

The blockchain is redistributed as a result of multiple devices that are connected with each other can store an identical copy of data packed into blocks chained together. Hence, what we call a Peer-to-Peer (P2P) network is created and has revolutionized, however, firms control financial risks and shield user data.

In 2008, this technology helped to create the start of cryptocurrency and in present time, mobile applications like Fold permits you to pay bitcoin in the real world.

These superb features have drawn developers and corporations into utilizing blockchain technology, and within the near future, we will expect this trend to be improved.

Mobile device is replacing all traditional channels to access the information. To align with this trend, enterprises too are designing the digital applications to cater to wide array of mobile devices and platforms.

Mobile application development is the process of creating software applications that run on a mobile device, and a typical mobile application utilizes a network connection to work with remote computing resources. Hence, the mobile development process involves creating installable software bundles (code, binaries, assets, etc.), implementing backend services such as data access with an API, and testing the application on target devices.

Key Drivers for Mobile Applications

Introduction to Mobile Application Development 8 The following are the key drivers of mobile apps: · **Innovation** in mobile space such as proliferation of smart phones, higher bandwidths offered by 3G (Third generation) and 4G (Fourth generation) technologies are coupled with higher capacity storage technologies with higher speed chips would keep powering mobile devices.

· **Consumer behavior:** Customers are more used to mobile devices and is easy for them to access information on the move.

· **Personalized content delivery:** Enterprise can leverage the location and sensors to offer more contextualized, relevant and personalized content, offers and advertisements.

· **Mobile ecosystem:** An explosive growth in Mobile Applications stores such as Apple store, Google Play store, Windows marketplace store was coupled with availability of games, utilities and other apps.

· **Social Networking:** With the popularity of web 2.0 and social media technologies such as Facebook, Twitter users are increasingly using the location-based features in the social media platforms.

1.2. **Introduction to Mobile Computing**

Mobile computing can be defined as a computing environment over physical mobility. The user of the mobile computing environment will be able to access data, information or logical objects from any device in any network while on move. Wireless telecommunications networks are generally implemented and administered using radio communication. This implementation takes place at the physical level (layer) of the **OSI** model network structure.

Examples of wireless networks include cell phone networks, wireless local area networks (WLANs), wireless sensor networks, satellite communication networks, and terrestrial microwave networks.

Mobility/Nomadic is that someone / something can move or be moved easily and quickly from place to place.

Computing – is the operations of the computer. Types of computing are.

Mobile Computing is a computing that connects a mobile device to network or another computing device, anytime, anywhere. The user of mobile computing environment will be able to access data, information or other logical objects from any device in any network while on the move. The computing environment is mobile and moves along with the user. This is similar to the telephone number of a **GSM (Global System For Mobile Communication)** phone, which moves with the phone. The offline (local) and real time (remote) computing environment will move with the user. In real time mode user will be able to use all his remote data and service online. **Nomadic Computing:** The computing environment is nomadic and moves along with the mobile user. It refers to limited migration is within a building at a pedestrian

speed, interleaved pattern of user relocation and indoor connections and users carrying laptop with wireless access are engaged in nomadic computing.

Pervasive Computing: A computing environment, which is pervasive in nature and can be made available in any environment. It refers to access to computer network all the time at any location by any person. It cannot be realized unless mobile computing matures.

Ubiquitous Computing: A disappearing (nobody will notice its presence) every place computing environment. User will be able to use both local and remote services. Global System for Mobile Communications (GSM): The GSM network is divided into **three major systems**: the switching system, the base station system, and the operation and support system. The cell phone connects to the base system station which then connects to the operation and support station; it then connects to the switching station where the call is transferred to where it needs to go. GSM is the most common standard and is used for a majority of cell phones. **Personal Communications Service** (PCS): PCS is a radio band that can be used by mobile phones in North America and South Asia. Sprint happened to be the first service to set up a PCS. D-AMPS: **Digital Advanced Mobile Phone Service**, an upgraded version of AMPS, is being phased out due to advancement in technology. The newer GSM networks are replacing the older system. The architectural model of a mobile computing environment is shown in Figure 1.1 and consists of stationary and mobile components. Fixed hosts are connected together via a fixed high-speed network (Mbps to Gbps). ***Some of the fixed hosts are special computers equipped with wireless interfaces, and are known as base (radio) stations (BS). They are also known as mobile support stations (MSS).*** Base stations, which are placed in the centre of cellular coverage areas, act as access points between the mobile computers and the fixed network. Mobile computers can be in one of three states.

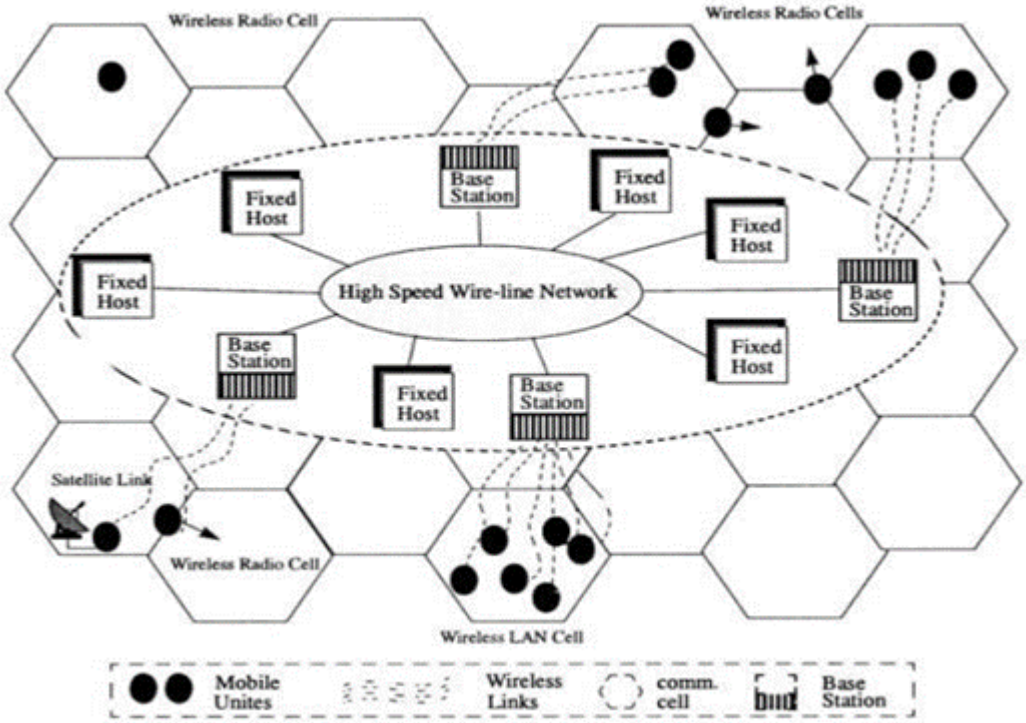


fig 1.1. Mobile Computing Environments

- The first state places a mobile computer within a cell and capable of communicating.
- The second state places the mobile computer out of range of any service cell and not capable of communication.
- The third state places a mobile computer in a cell, communicating, but just ready to cross a cell boundary. These scenarios are depicted in Figure 1.1.

A computing environment is defined as mobile if it supports one or more of these characteristics:

1**. User mobility**: User should be able to move from one physical location to another location and use same service

2. **Network mobility**: User should be able to move from one network to another network and use same service

3. **Device mobility:** User should be able to move from one device to another and use same service

4. **Session mobility:** A user session should be able to move from one user-agent environment to another.

5. **Service mobility:** User should be able to move from one service to another.

6. **Host mobility:** The user should be either a client or server. The mobile computing functions can be logically divided into following major segments

1) **User with device**: The user device, this could be fixed device like desktop computer in office or a portable device like mobile phone. E.g Laptop Computers, Desktop Computers, Fixed Telephones, Mobile Phones, Digital TV with set top box, palmtop computers, pocket PCs, two-way pagers, handheld terminals etc.

2) **Network**: Whenever a user is mobile, he will be using different networks at different places at different time eg GSM, CDMA, iMode, Ethernet, Wireless LAN, and Bluetooth etc.

3) **Gateways:** This is required to interface different transport bearers. These gateways convert one specific transport bearer. These gateways convert one specific transport bearer to

1.2. Introduction to Android Development Environment

1.2.1. **System Requirements**

Android application development may be performed on any of the following system types:

- Windows Vista (32-bit or 64-bit)
- Windows 7 (32-bit or 64-bit)
- Windows 8/10 / 32-bit or 64-bit Windows 8.1 or later
- Mac OS X 10.8.5 or later (Intel based systems only)
- Linux systems with version 2.15 or later of GNU C Library (glibc)
- Minimum of 2GB of RAM (4GB is preferred)
- Approximately 4.5GB of available disk space

Before any work can begin on the development of an Android application, the first step is to configure a computer system to act as the development platform. This involves a number of steps consisting of installing the Java Development Kit (JDK) and the Android Studio Integrated Development Environment (IDE) which also includes the Android Software

Development Kit (SDK). This chapter will cover the steps necessary to install the requisite components for Android application development on Windows, Mac OS X and Linux based systems.

1.2.2.Installation of Android Studio.

Android Studio runs on Java Runtime Environment (JRE). JRE can be installed on Windows, Mac and Linux computers. We need to follow the steps given below for the installation of Android Studio independent of our operating system:

1. **Installation of Java:**

   Java is developed by Oracle Inc. There are basically two Java packages: Java Runtime Environment (JRE) and Java Software Development Kit (JDK). JRE is used for running software written in Java programming language whereas JDK is utilized for developing Java software. Therefore, installing JRE is adequate for running Android Studio because we will not develop Java software here. Please navigate to the following website to download the JRE:

   http://www.oracle.com/technetwork/java/javase/downloads/jre8- downloads-2133155.html.

- **Installation of Android Studio and Android SDK**

  Android Studio is bundled with Android Software Development Kit (SDK). Please navigate to the official download site located at: https://developer.android.com/studio/index.html. The download link for the Windows version is shown at the top of this site but if you scrolldown, you can find the setup files available for download for other operating systems as shown in Figure 1.2.

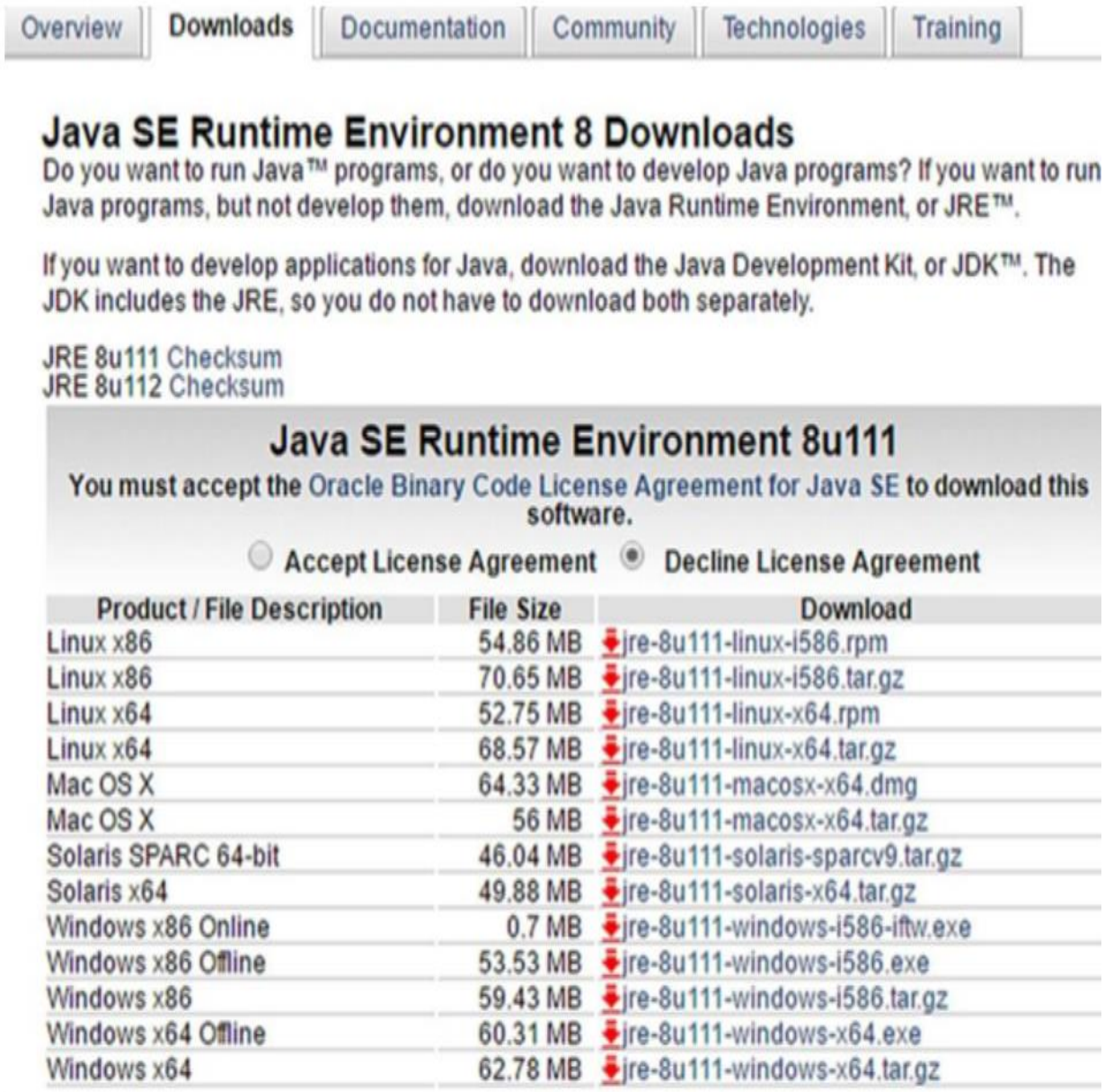


Figure 2: Download options for Java Runtime Environment

Select a different platform

Platform	Android Studio package	Size	SHA-1 checksum
Windows (64-bit)	android-studio-bundle-145.3537739-windows.exe Includes Android SDK (**recommended**)	1.674 MB (1,756,130,200 bytes)	272105b119adbcababa114abeee4c78f3001bcf7
	android-studio-ide-145.3537739-windows.exe No Android SDK	417 MB (437,514,160 bytes)	b52c0b25c85c252fe55056d40d5b1a40a1ccd03c
	android-studio-ide-145.3537739-windows.zip No Android SDK, no installer	438 MB (460,290,402 bytes)	8c9fe06aac4be3ead5e500f27ac53543edc055e1
Windows (32-bit)	android-studio-ide-145.3537739-windows32.zip No Android SDK, no installer	438 MB (459,499,381 bytes)	59fba5a17a508533b0decde584849b213fa39c65
Mac	android-studio-ide-145.3537739-mac.dmg	434 MB (455,263,302 bytes)	51f282234c3a78b4afc084d8ef43660129332c37
Linux	android-studio-ide-145.3537739-linux.zip	438 MB (459,957,542 bytes)	172c9b01669f2fe46edcc16e466917fac04c9a7f

figure 3. Download options for Android Studio

When you download and install Android Studio, Android SDK will also be automatically installed.

3. Installation of SDK updates: After the installation of Android Studio, it is better to check SDK updates. For this, run Android Studio and open the SDK manager from Tools à Android à SDK Manager as shown below:

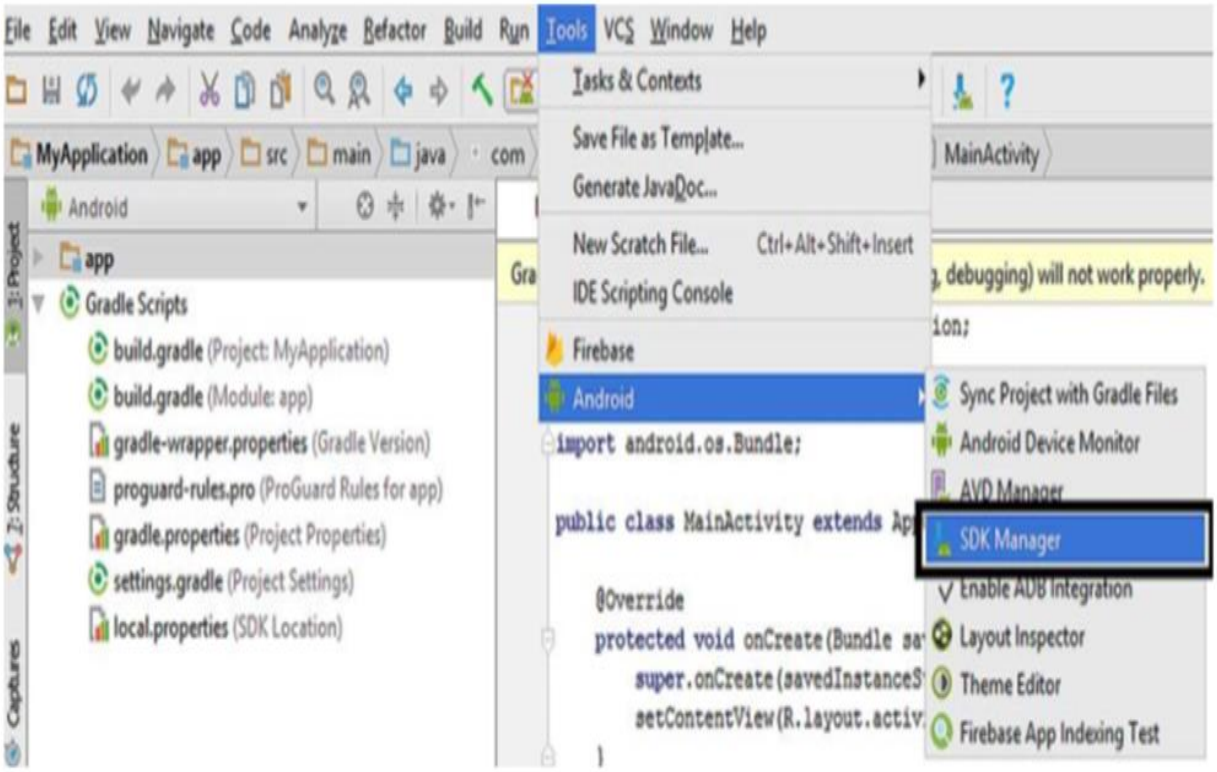


figure 4. opening the SDK manager

The SDK Manager window will appear as shown in Figure 1.5.

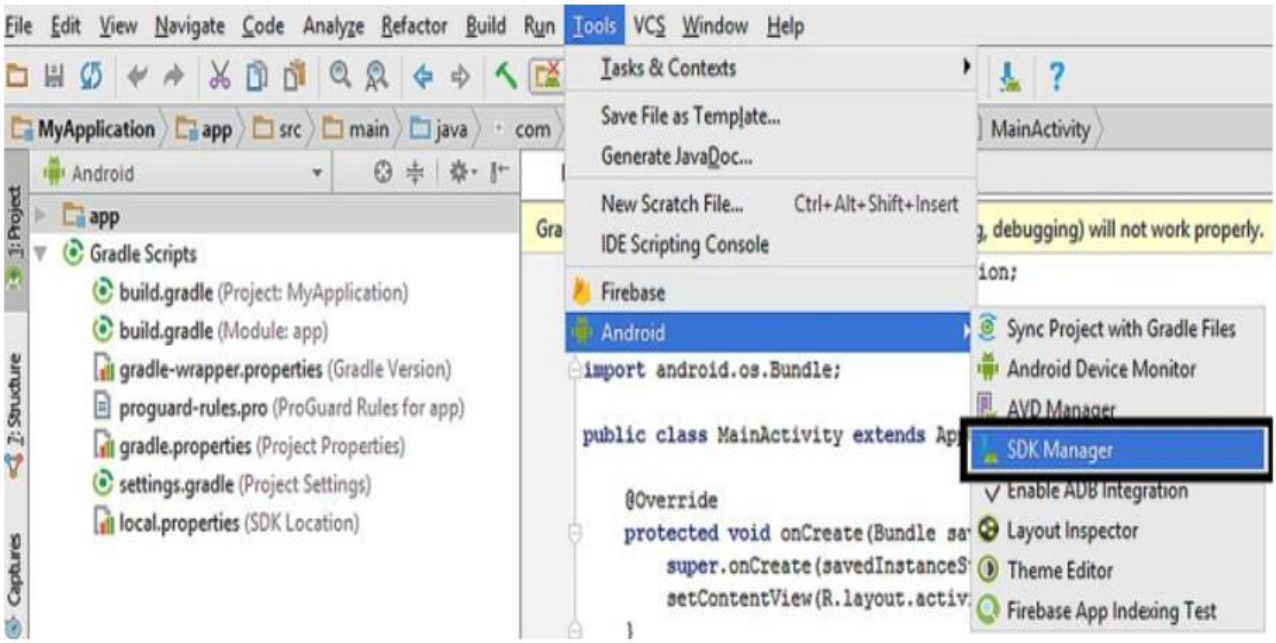


Figure 4: Opening the SDK Manager

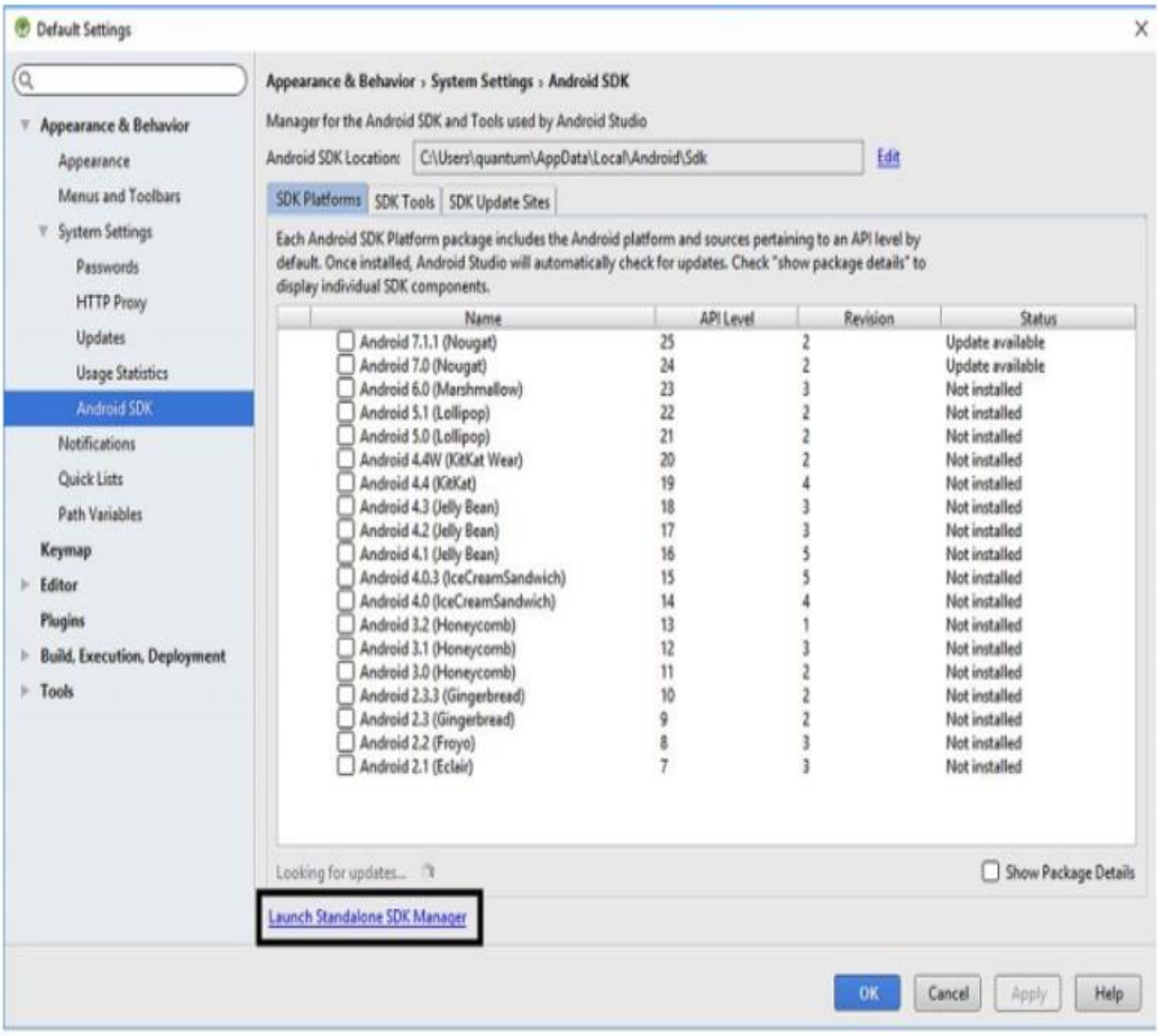


Figure 5:Android SDK Manager

Please open the standalone SDK Manager by clicking the link indicated in Figure 2.4. In the standalone SDK Manager, click on the "Install … packages" as shown below:

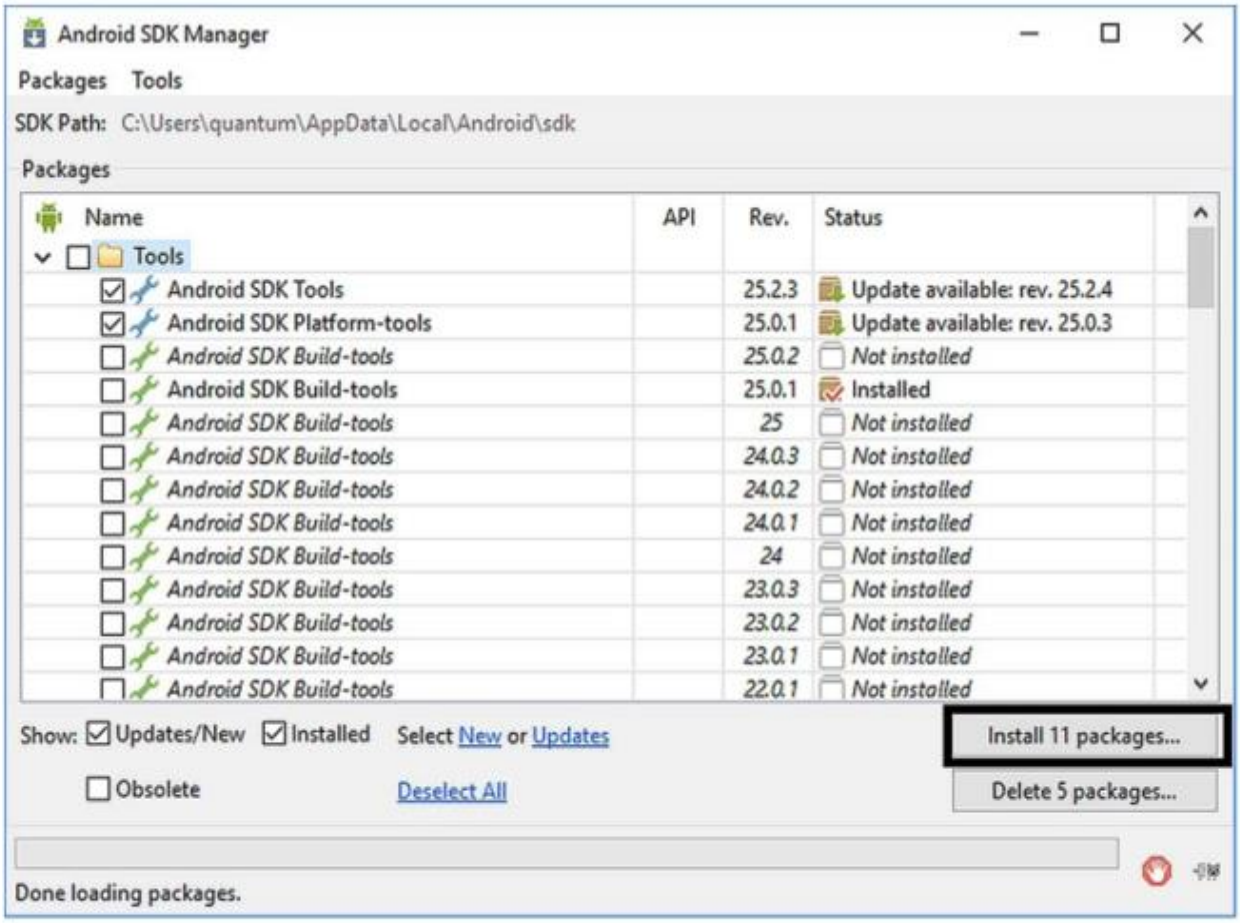


Figure 6:Standalone SDK Manager

After you install the packages, you‟ll have the latest SDK and be ready to develop apps. However, before our test drive app one more step is needed: setting up the emulators.

Installation of Emulators

Emulators are software that mimics the behavior of real devices. When we develop an app, we obviously won't have all the possible devices (Android phones, tablets, etc.) available at hand. Because of this, we run the apps on emulators for testing on various devices. Emulators are also called as "Android Virtual Devices (AVDs)" in Android Studio. When Android Studio is first installed, there is no default AVD. We need to create one before testing our apps. For this, select **Tools à Android àAVD** Manager as shown in Figure 2.6.

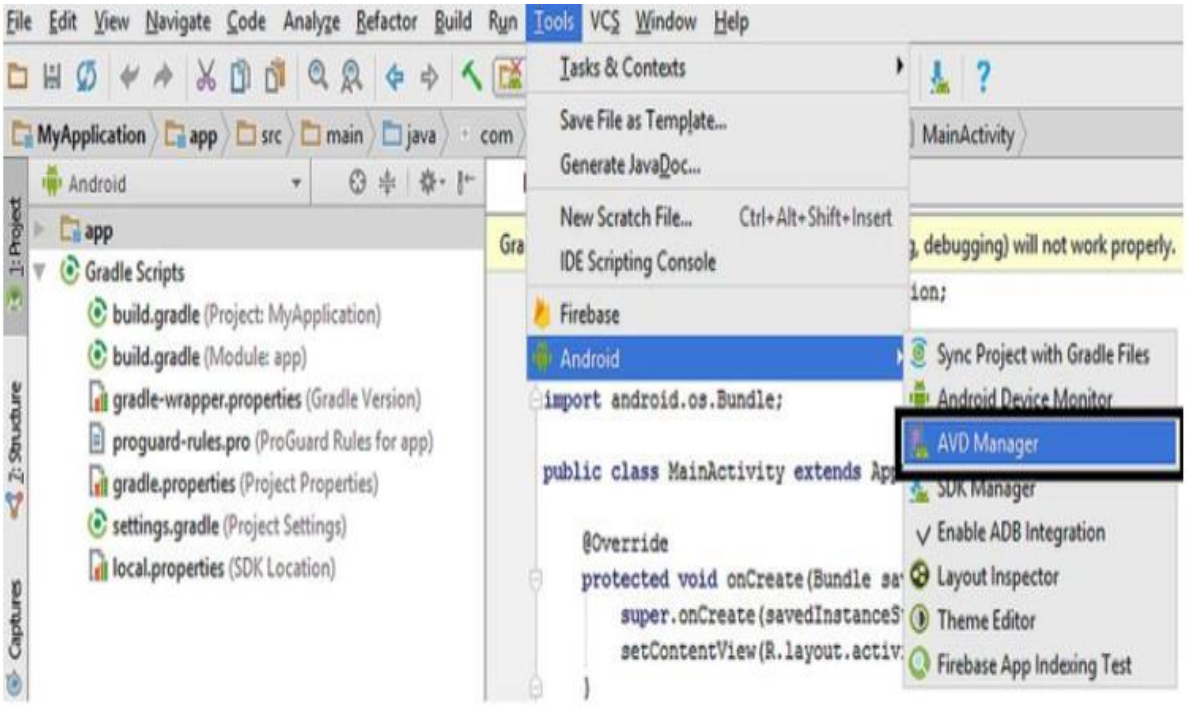


Figure 7:Launching the AVD Manager

When AVD Manager appears, there won't be any AVDs created or installed. Please click on the **+ Create a Virtual Device** button as shown below:

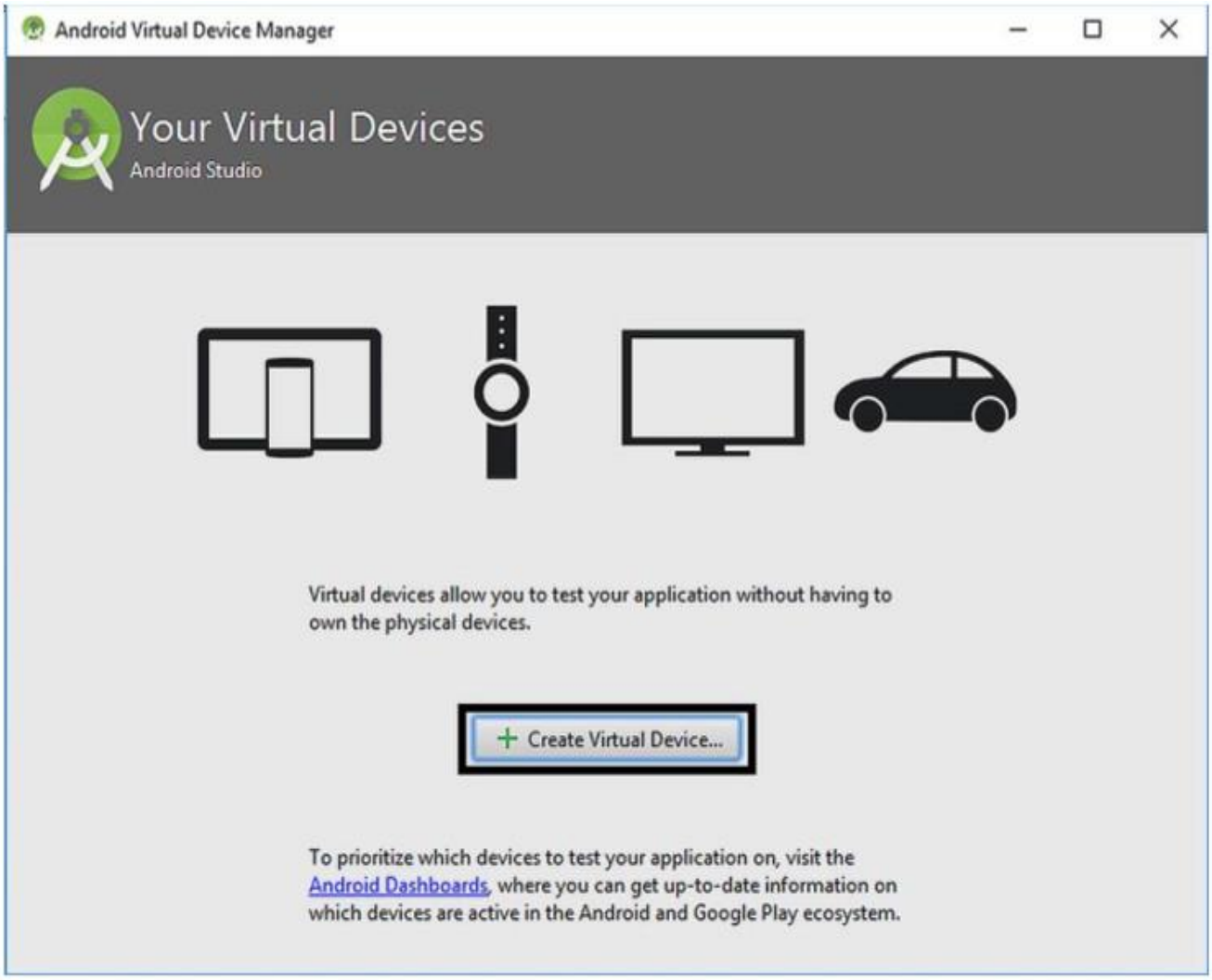


Figure 8:Creating a new AVD

AVD Manager will show a detailed window as in Figure 1.8. You can select various devices with different screen sizes and other hardware properties. You can select device groups from the left pane as TV, Phone, etc. Phone group is the default selection. In this group, Nexus 5 is also selected by default. When you click "Next", you'll be presented by choices for the Android version of the AVD as shown in Figure 1.9.

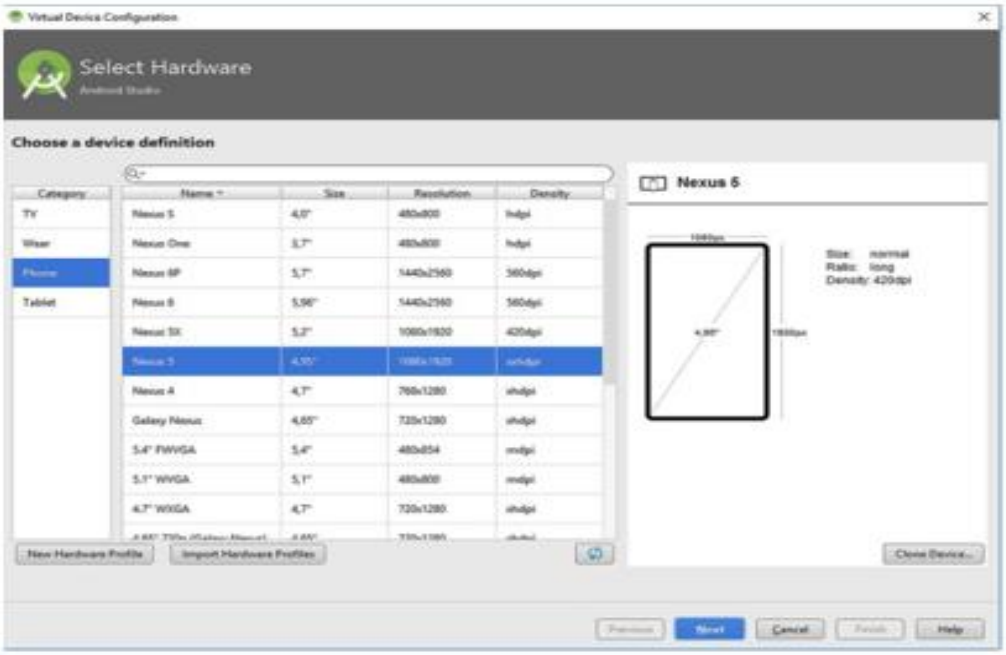


Figure 9:Creating a new AVD – selecting

The recommended targets start from Android 5.1. We can Android 7.0 with Google APIs (Nougat) as shown in the figure. Then, please click “Next” and give a name you like to the AVD. I didn’t change the defaults in the next screen as shown in Figure 2.10. After clicking “Finish”, the AVD is created and shown in the AVD Manager as in Figure 2.11. You can now try your Android apps on this AVD, which will accurately mimic the behavior of a real Nexus 5 phone. We can run the AVD by clicking the “Play” button shown inside the square in Figure 2.11. The virtual device will appear as in Figure 2.12 which you can use like a real Nexus 5 phone. After installing both the development environment and the emulator, we’re now ready to develop our test drive app, Hello World, in the next chapter.

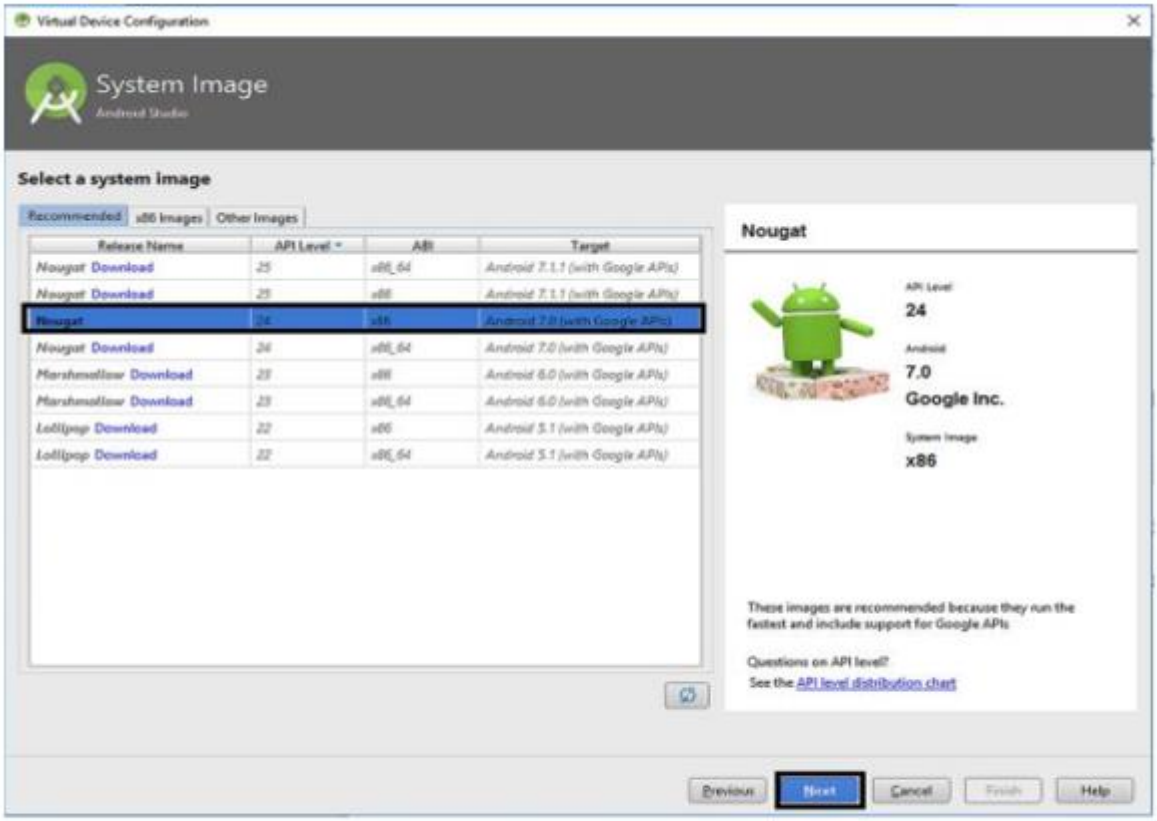


Figure 10:Selecting the Android version of the AVD

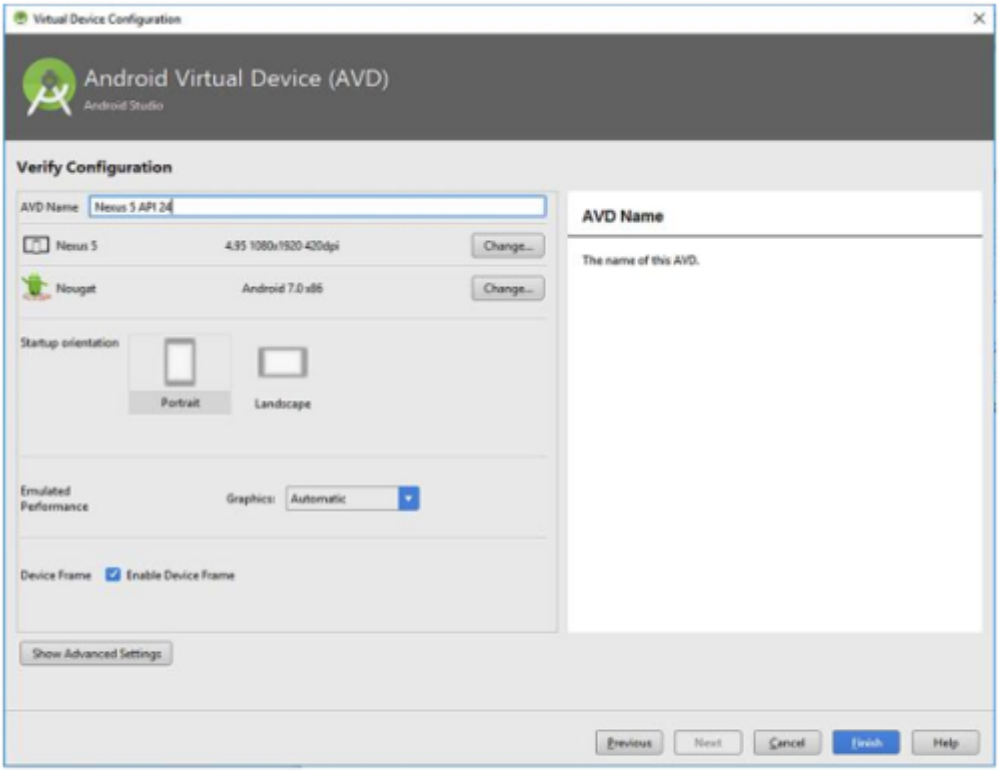


Figure 11:Final settings of the AVD

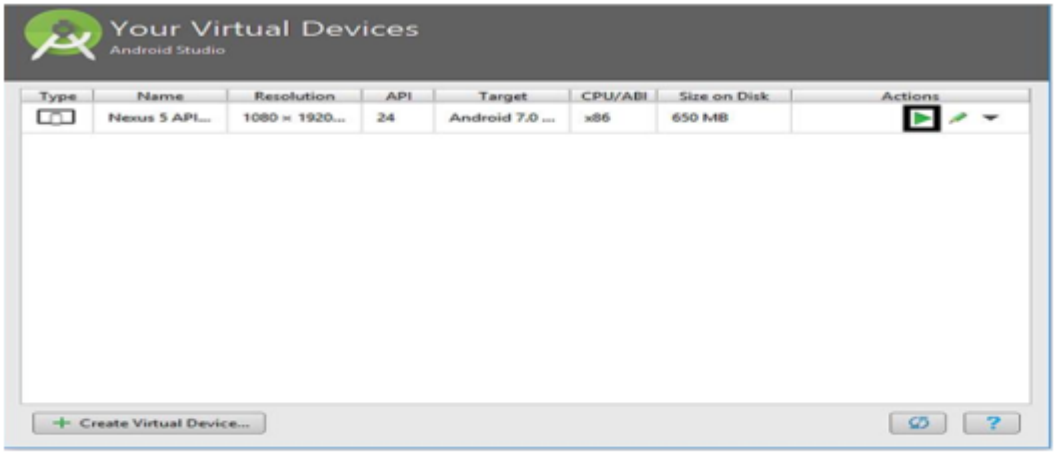


Figure 12:Newly created AVD shown in the



Figure 13:Nexus 5 emulator window

Within the Android SDK Tools screen, make sure that the following packages are listed as

Installed in the Status column:

- Android SDK Build-tools
- Android SDK Tools
- Android SDK Platform-tools
- Android Support Repository
- Android Support Library
- Google Repository
- Google USB Driver (Windows only)· Intel x86 Emulator Accelerator (HAXM installer)

In the event that any of the above packages are listed as Not Installed or requiring an update, simply select the checkboxes next to those packages and click on the Apply button to initiate the installation process.

Once the installation is complete, review the package list and make sure that the selected packages are now listed as Installed in the Status column. If any are listed as Not installed, make sure they are selected and click on the Install packages… button again.

An alternative to using the Android SDK settings panel is to access the Standalone SDK Manager which can be launched using the link in the lower left-hand corner of the settings screen. The Standalone SDK Manager (Figure 1.13.) provides a similar list of packages together with options to perform update and installation tasks:

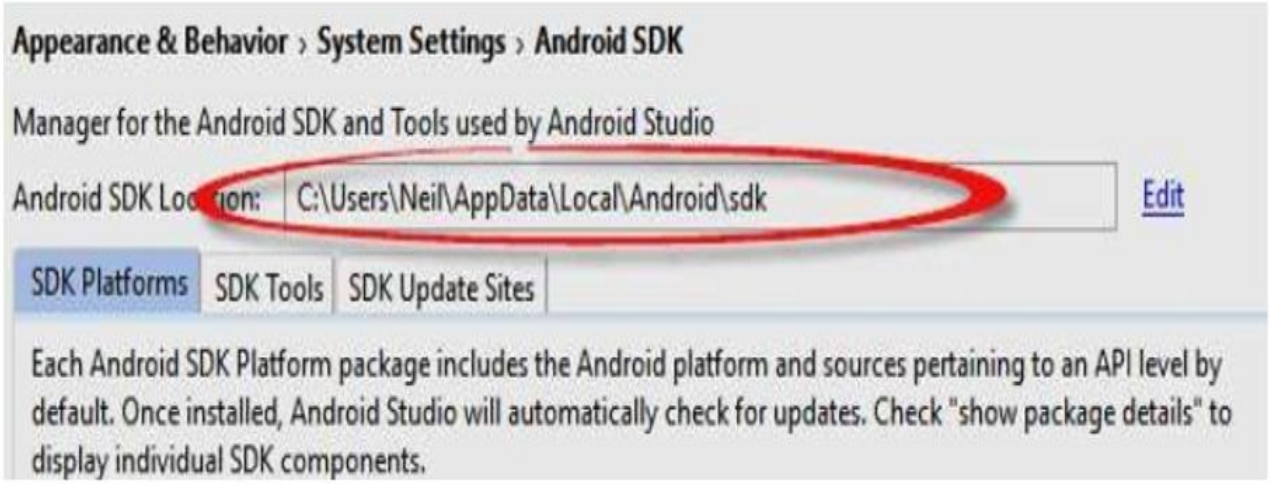


figure 14. SDK location Editor

Once the location of the SDK has been identified, the steps to add this to the PATHvariable is operating system dependent.

Chapter Two

2. Factors in Developing Mobile Applications

After completing this session(chapter), Students must able to:

- Identify the factors of developing mobile application
- Explain Mobile Software Engineering
- Understand mobile frameworks and tools
- Develop mobile generic UI

2.1. Introduction

This unit introduces learner to the subject of development of mobile applications. With the increased usage of mobile devices across the world, more and more applications are being targeted at them. If an application that is developed for computers can also run on the mobile devices, then the entire paradigm of application usage changes as it enables the user to run application from wherever he is. There were different standards on which the mobile networks are based. One of the popular standards is GSM. In this unit, we introduce the GSM architecture. There were a number of environments under which mobile applications can be developed. One of the popular environments is Java. We introduce J2ME (Java 2 Micro Edition) in this unit in conjunction with wireless application development with it. Any application should be tested before being put to full-fledged use. In this unit, Java Device Test Suite is introduced with the help of which wireless applications can be tested.

2.2. Mobile Software Engineering

Mobile application development is the set of processes and procedures involved in writing software for small, wireless computing devices, such as smartphones and other hand-held devices.

Learn about the factors that influence your mobile app development process by taking note of the items on this list.

1. Distribution Channel

As a developer, you must first identify which platform best supports your grand idea. Though Google Play Store and Apple App Store serve the same function, they have different guidelines. You can view the description, icon, logo, screenshots or image gallery, and a snippet or a trailer of the app on both platforms. However, there are few elements that differentiate one from the other.

For instance, businesses generate more income when they first release their app on the App Store than in Google Play Store. In terms of description, Google Play Store gets more taps because of their read more button is in a prominent position. You can find the description hidden above the image gallery and screenshots. The App Store, on the other hand, optimizes their description for iPhone screens. You can read the description directly without needing to tap on a button or scroll down. In terms of screenshots or image gallery, Apple App Store has an advantage because the Play Store places them below the fold. These factors play a part in your app's user engagement. The setup of the platforms can influence how people respond to your app. Moreover, the platforms are also essential when thinking of other factors such as monetization, security, and the availability of different types of apps.

2. Audience Research

For your idea to become successful, you must first conduct audience research. Knowing your audience involves understanding the industry and getting in-depth knowledge about the people that might take an interest in your apps. It is important because users tend to be diverse. You must track the demographics of the users of your app. That way, you can identify the age, gender, relevant businesses, and potential customers you can target. If you want your app to solve a problem, you should not base your decisions on assumptions. Guessing users' tastes and preferences lead to poorly executed app development. You can make an approximation of the gender that is likely to use your app, but it's best if you conduct surveys, look into online communities, and use tools to get to know your market. Creating different personas also helps you visualize the needs of different users. It helps you identify which parts of the project you must prioritize and which features to include in your app.

3. App Idea

Apps don't come out of thin air. They start as brilliant ideas. Unfortunately, not all ideas become great and successful once you turn them into an app. Remember that working on an app development project requires money, effort, and time. Therefore, you must first examine your idea before you can begin working on it. You must first check if your design is unique. With thousands of apps on both platforms, you are sure to find one that already has the concept you have in mind. Besides being unique, your app idea must satisfy a need. There's no use wasting your resources if no one needs and wants your concept. Of course, you must also find out if your idea has a lot of competition. It may be difficult to break through in a saturated market. Most of all, you must be able to differentiate your idea from the what's already available on the platforms. It must also be able to resolve the weaknesses of the existing competitive apps.

A Nielsen survey on consumer preference for mobile apps vs. mobile sites found that **Indian consumers prefer apps that personalize their experience**. You can improve your app idea when you research your audience as well as your competition. It helps strengthen your app idea and push you to exhaust your brain juices to generate innovative solutions.

4. App Development Strategy

To properly execute your app idea, you must come with a plan. In this case, you must have a solid app development strategy. A cohesive strategy answers all the questions you have towards the app development process. It clarifies how much of the design work you will undertake for the app development. It covers your timeframe for working in the project. It also factors in the technology you will use to develop your app idea. Part of your app development strategy must include the strengths and weaknesses of each mobile device. You need all of these to step up your game and help you execute an app development plan. If you over or underestimate in any of the factors, your app can end up becoming a failure.

5. UI/UX

User interface (UI) and u**ser experience** (UX) are part of what keeps users hooked on an app. According to the founder and CEO of Apps Discover Technologies, **Ashish**

Bahukhandi shared that successful apps are those that provide a hassle-free experience aside from serving its purpose. Users prefer apps that are "*handier*" compared to those that require too much effort to operate. Navigation and usability aren't the only ones that matter to the users. They are also receptive to several UI/UX factors. For instance, an app with low-res images, slow loading times, broken sign up/ login functionality and app navigation, and poor connectivity to social profiles are all signs of poor UI/UX. Unnecessary micro-interactions, lags or unresponsive clickable elements, inappropriate spaces, and mismatch fonts types and size are some of the factors you should watch out for when developing your app.

6. Testing

Developing and designing your mobile app idea is only half of the job in your mobile app development journey. You must also run a few tests to ensure your design runs smoothly. It's a way to secure that your mobile app has no exceptions and bugs. If you launch your app without testing it, you can run into problems that can spark fires that are difficult to put out. There are **seven ways** to test your app and check if it doesn't have bugs. You should see if each of the app's function. Each time you add a function, you must see if the existing features are still working correctly. You must also look into the app's integrations and usability. It's also important to find out if your app is compatible with different platforms and devices. Most of all, you should test if the app functions properly at the user's end before going live. **Avinash Tiwari,** Co-founder and Director of pCloudy, states that developers must know the app development ecosystem. You must stay informed of the latest updates in OS versions as well as market trends. These practices help you anticipate problems with latest released apps. You should always check release notes to identify potential issues that can encourage users to abandon your app.

7. Security

Security must be part of your app development process. Users should not have to think of the threat of malware when they use your app. Thus, you should leverage the security features of the device's OS. You can also keep the user's information safe through encryption. You should also restrict your app's permissions to a minimum. Always prioritize security to ensure your app isn't vulnerable to external threats.

8. Monitoring

You still have work to do after you publish your mobile app. You must be hands-on with fixes, performance improvements, changes, and integrating new features. As a developer, you must track crashes and look at technical information to help you resolve the problems users encounter. You can also check into analytics as well as recommendations and reviews by users to see how well the market responds to your app. The information you get from these platforms is what you need to identify trouble spots and make room for improvement.

Lifecycle management – Once an app is deployed, developers must test, host, deploy, maintain and analyze its usage throughout its lifecycle. As a result, it's important for developers to consider the long-term capabilities and long-term costs of tools.

Delivery – Apps aren't done when they go into production. They must be dependable. They must scale. They will need updating based on evolving requirements and end user feedback. Developers need to make sure the tool they select will allow for easy upgrades, updates, availability, scalability and performance.

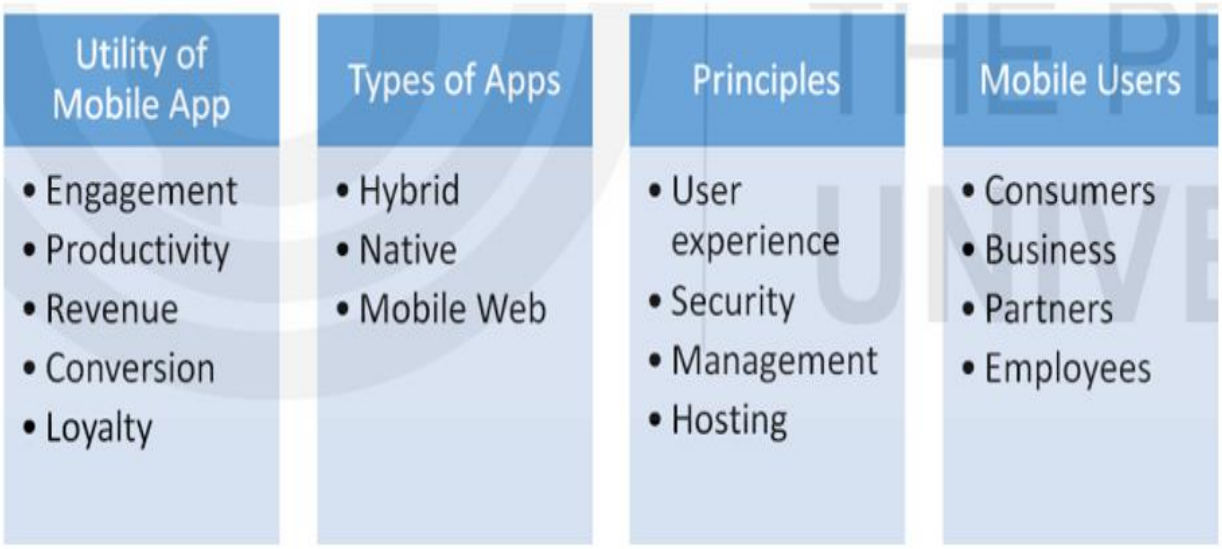


Figure 16: Mobile App Considerations

The main considerations for mobile app design are listed below:

· Intended utility of the mobile app

- Consumer engagement with richer user experience
- Productivity through efficient flows
- Driving incremental revenue through user stickiness
- Customer conversion
- User loyalty through targeted and personalized offers

· App Architecture

- Native vs hybrid vs web based on the requirements
- Middleware requirement for centralized configuration
- Offline vs online capability for storing data

· App Development Principles

- User experience through richer controls and interactive components
- Compatibility on various devices and platforms
- Performance for each screen and task
- Security for data o Productivity enhancement tools

· Target users

- Consumers for B2C apps
- Business for Business to Business (B2B) apps
- Partners for B2B apps o Employees for Business to Employee (B2E) apps

· Testing

- Device testing
- Performance testing
- Various testing scenarios The main challenges in mobile app strategy are given below:

· **Diversity of devices and heterogeneous technologies**: There are various mobile platforms and devices. The app should provide optimal experience in all the scenarios.

· **Security**: Mobile app should ensure data security during transmission and during storage.

· **User experience**: Mobile app should provide optimal user experience leveraging the device capabilities to provide highest engagement possible.

· **Network**: Mobile app should be designed to work in regions with network, latency and bandwidth challenges.

· Compliance to diverse standards, OS, mobile platforms and devices.

2.3.Frameworks and Tools

Mobile engineers design, develop, and implement software programs for smartphones and other mobile devices. They often specialize in a certain type of operating system, such as Android or iOS. They use data-driven insights to guide development of programs and apps that fill a user need. They create specifications and detailed plans, write code, oversee testing and debugging, and play an active role in the rollout and launch. They commonly work in close collaboration with other engineers, designers, R&D, analysts, and other teams and stakeholders.

Mobile engineers typically have at least a bachelor's degree in an engineering or computer science field. They generally have a background in engineering or computer programming. Proficiency with common programming languages is required, and experience with automated testing tools on a mobile platform is helpful.

The rapid availability of information and the possibility of rapid processing of data outside of operational areas and office environments are important competitive factors for a growing number of companies today.

Mobile software solutions for laptops, tablet computers and smartphones with touch or keyboard operation represent an important technical basis for this.

The apps and applications developed by us are characterized by their simple user guidance and intuitive operation. The technical possibilities of the available mobile devices are always taken

into account and optimally used. This can be reflected both in the type of operation (touch or keyboard) and in the type of data processing (online or offline).

platforms

In the field of mobile applications, we primarily develop apps and programs for the following platforms:

- iOS
- Android

1.1. Frameworks and Tools

Apps are a modern necessity and I seriously think most of us would be lost without them (Even literally lost without Google Maps!). And since apps are so popular, it stands to reason that it is equally popular (and profitable!) to develop them. Apps are an excellent way to create a brand and build a loyal fan base (and also have fun while doing so!). That is the reason this article deals with the **Top Android Development Frameworks** that are currently most popular and so important to learn in 2019.



Figure 17:. framework

2.4.Android Development Frameworks

In general, the basic method of creating an Android app is to use Android Studio (As it is the official IDE for Android). However, there are many other Android Development

Frameworks that are quite popular in and to provide you a little variety, the list is given below…

1. Ionic

Ionic Framework is a very popular open-source mobile framework that was released in 2013. It's also free and MIT open source (so it will most probably remain free!). According to its website, Ionic can be used to **"Build and ship beautiful cross-platform hybrid and Progressive Web Apps with ease."** And that's not all! Ionic is also a cross-platform framework, which means that you can create apps and then customize them for Android, iOS, Windows, etc. Also, if you have a problem with your documentation, fear not! Ionic has documentation support with real-life examples.

2. Xamarin

Xamarin is an extremely popular Mobile framework (With over 1.4 million developers using it!) that was acquired by Microsoft in 2016 and is currently a Microsoft subsidiary. According to Xamarin, **"Apps built using Xamarin look and feel native because they are."** So basically, Xamarin can be used to write native Android, iOS and Windows apps with a C# codebase (Xamarin is great for C# developers!) that can also be shared across multiple platforms. Xamarin is also popular (and only!) IDE with Microsoft Visual Studio and can be used to create native apps with AutoComplete and IntelliSense.

3. Adobe PhoneGap

Adobe PhoneGap is an open-source mobile application development framework that was purchased by Adobe Systems in 2011. Its unique feature is that it can be used to build mobile applications using HTML5, CSS3, and JavaScript instead of using API like Android, iOS, or Windows Phone. According to PhoneGap, it can be used to **"Reuse existing web development skills to quickly make hybrid applications built with HTML, CSS, and JavaScript."**

4. React Native

React Native is an open-source mobile framework that was created by Facebook and released in 2015. According to Mark Zuckerberg, **"The biggest mistake we made as a company was betting too much on HTML as opposed to native"** and that lead to the creation of React

Native. It provides support for various IDEs and mobile app development tools and that is one of the main reasons it is used by many (popular!) companies such as Instagram, Tesla, Walmart, Baidu, etc. to develop their mobile apps.

5. Flutter

Flutter is a comparatively recent open-source development framework that was released by Google in 2017. According to Google, **"Flutter is Google's portable UI toolkit for building beautiful, natively-compiled applications for mobile, web, and desktop from a single codebase."** Flutter is written in the Dart programming language (Also developed by Google) and it is currently used by famous companies such as Alibaba, Tencent, Google (Of course!). Flutter provides rendering support by using the 2D rendering engine Skia.

6. Titanium SDK

Titanium SDK is an open-source framework that was created by Appcelerator, Inc. and released in 2018. (It's quite new!) It is basically a development environment that can be used to create native apps across various operating systems like iOS, Android, BlackBerry, etc. Titanium is quite popular and has been used in around 70, 000+ Mobile Applications till now. And that's not all! According to the Titanium website **"The Titanium environment helps organizations get to market 60% faster and achieve a significant competitive advantage."**

7. JQuery Mobile

jQuery Mobile is a Touch-Optimized Web Framework that was developed by the jQuery team and released in 2010. It is based on HTML5 and can be used to create responsive web sites and apps that can be accessed using multiple devices such as smartphones, tablets, desktops, etc. According to the jQuery team, this takes the **"write less, do more"** mantra to the next level!!! In addition to that, jQuery also provides support for different operating systems such as Windows, Firefox, Mac OS, Blackberry, Kindle, etc.(So make your choice!)

8. Corona SDK

Corona Software Development Kit was developed by Corona Labs Inc. and released in 2009. It is a great platform for creating apps for mobile as well as desktop systems. Corona Labs states that **"You can create your project once and publish it to multiple types of devices,**

including Apple iPhone and iPad, Android phones and tablets, Windows Desktop, etc." Also, Corona is incredibly fast as it is supported by Lua, an open-source scripting language that is lightweight, fast, and powerful as well (Triple whammy!)

9. Native Script

Native Script is an open-source framework for creating native (duh!) mobile apps that were developed by Progress and initially released in 2014. It was developed using JavaScript and TypeScript (which is basically a superset of JavaScript). Native Script also supports Angular and *Vue.js* frameworks and these can be used to share existing web-based code. It also doesn't hurt that Native Script is **100% free and open source**!

10. Mobile Angular UI

Mobile Angular UI is a mobile UI framework (As is clear from the name!) that allows the creation of HTML5 hybrid mobile apps using a combination of Angular JS and Bootstrap. According to the Mobile Angular UI website, **"If you know Angular JS and Twitter Bootstrap you already know Mobile Angular UI!!!"** In essence, Mobile Angular UI provides some components that are not available in Bootstrap 3 such as switches, scrollable areas, sidebars, overlays, etc. and also provides Angular.js directives to provide a better mobile experience.

2.5.Generic UI Development

A good User Interface (UI) focuses on making user's interactions simple and efficient. User would appreciate a website with intuitive user interface that leads them towards their task in most engaging way. User Interface (UI) design focuses on thinking of a user, what they might need to do when they visit website and ensure that the interface has elements that are easy to access and understand. Being a UI designer, one need to understand the goals, skills, preferences and tendencies of the user to make a better interface.

The Generic User Interface (Generic UI, GUI) framework allows you to create UI screens using Java and XML. XML is optional but it provides a declarative approach to the screen layout and reduces the amount of code which is required for building the user interface.

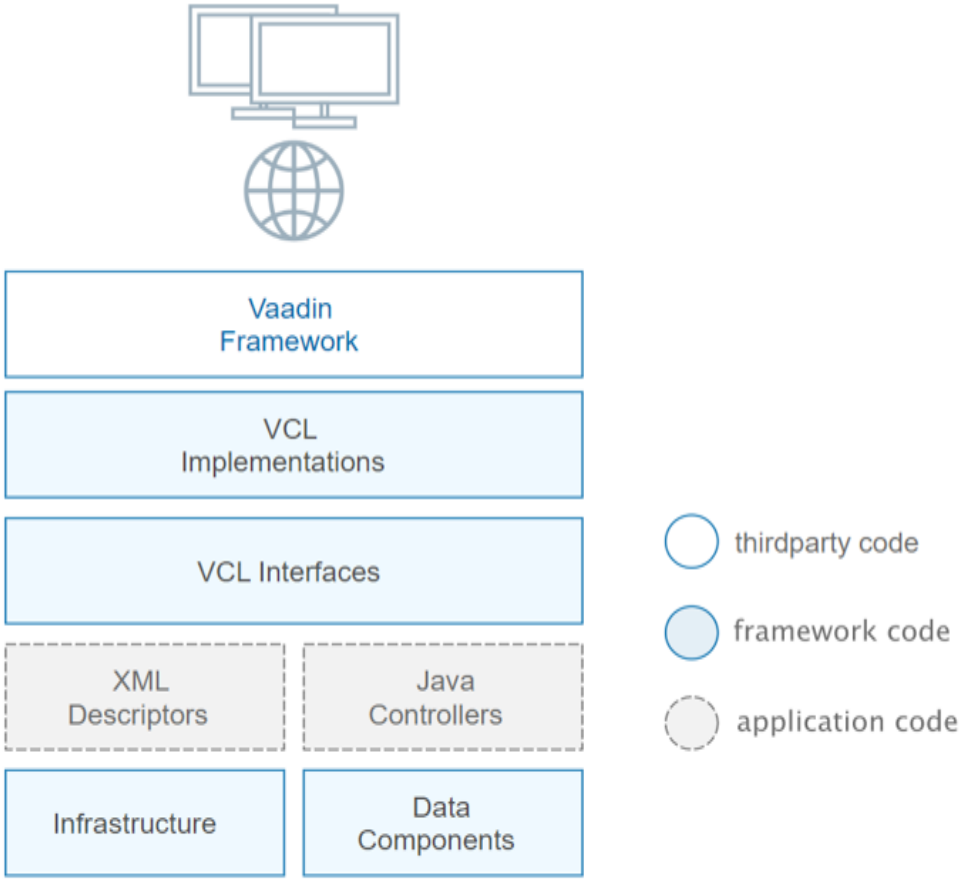


Figure 18: The Structure of Generic User Interface

The application screens consist of the following parts:

- Descriptors – XML files for declarative definition of the screen layout and data components.
- Controllers – Java classes for handling events generated by the screen and its UI controls and for programmatic manipulation with the screen components.

The code of application screens interacts with visual component interfaces (VCL Interfaces). These interfaces are implemented using the Vaadin framework components.

- Visual Components Library (VCL) contains a large set of ready-to-use components.
- Data components provide a unified interface for binding visual components to entities and for working with entities in screen controllers.
- Infrastructure includes the main application window and other common client mechanisms.

A screen is a main unit of the generic UI. It contains visual components, data containers and non-visual components. A screen can be displayed inside the main application window either in the tab or as a modal dialog.

The main part of the screen is a Java or Groovy class called controller. Layout of the screen is usually defined in an XML file called descriptor.

In order to show a screen, the framework creates a new instance of the Window visual component, connects the window with the screen controller and loads the screen layout components as child components of the window. After that, the screen's window is added to the main application window.

A fragment is another UI building block which can be used as part of screens and other fragments. It is very similar to screen internally, but has a specific lifecycle and the Fragment visual component instead of Window at the root of the components tree. Fragments also have controllers and XML descriptors.

A screen controller is a Java or Groovy class that contains the screen initialization and event handling logic. Normally, the controller is linked to an XML descriptor which defines the screen layout and data containers, but it can also create all visual and non-visual components programmatically.

All screen controllers implement the Frame Owner marker interface. The name of this interface means that it has a reference to a frame, which is a visual component representing the screen when it is shown in the main application window. There are two types of frames:

- Window – a standalone window that can be displayed inside the main application window in a tab or as a modal dialog.
- Fragment – a lightweight component that can be added to windows or other fragments.

Controllers are also divided into two distinct categories according to the frames they use:

- Screen – a base class of window controllers.
- ScreenFragment – a base class of fragment controllers.

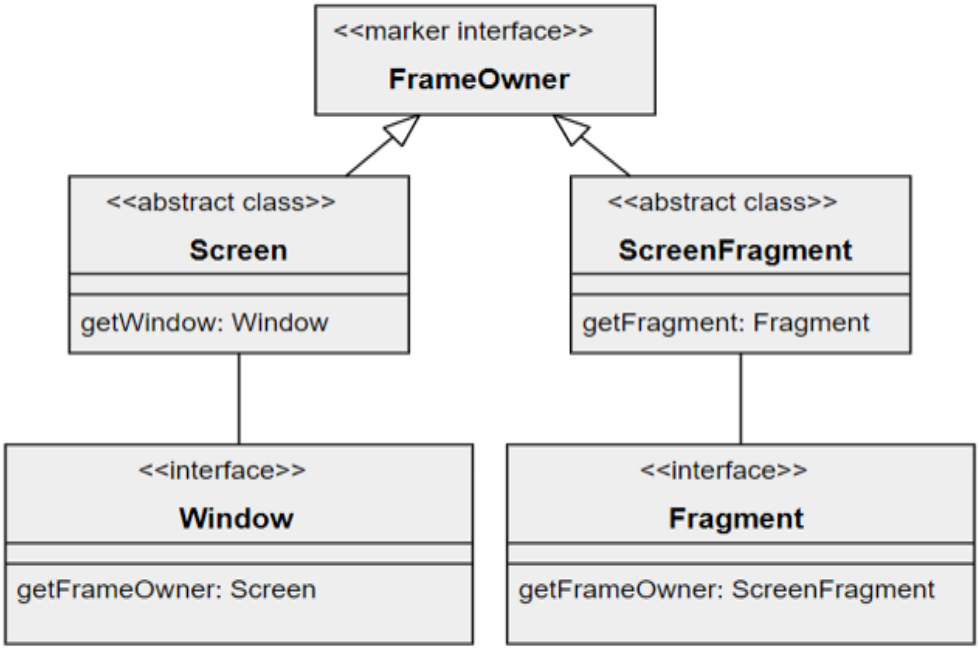


Figure 19:Controllers and Frames

Most of these user interactions are touch-based and happen on colorful touch screen displays that are bursting with high-level interactions. Naturally, basic mobile UI design principles differ from those of a traditional desktop UI. After all, users are, by definition, on the move; control is limited, giving new meaning to the phrase 'all thumbs. Actions and information need to be big, bold, clear, and simple.

As mobile adoption continues to rise year-by-year, it's time to develop a mobile-first strategy, embraced by the likes of Facebook or other social networks, who make sure their iOS and Android apps offer a polished user experience on hand-held devices. After all, when users have more choice and freedom to find mobile applications that work for them, a poor user experience can easily devalue your brand, hurt your revenue, and disengage your users.

Aside from investing in mobile applications, many ecommerce stores see the increase of purchases coming from mobile. If an online store doesn't optimize checkout experience, usability or their mobile app design, they may lose market share or even render themselves obsolete.

2.6.Android User

Android is the most popular operating system in the world, with over 2.5 billion active users spanning over 190 countries. Created by Andy Rubin as the open-source alternative to iPhone and Palm OS, Android quickly became the favorite operating system for most mobile manufacturers in the early 2010s.

With titans of the industry such as Samsung, LG, Motorola and HTC all launching phones running Android, it quickly became the most popular mobile OS, hitting over one billion active users by 2014.

Android has added more users every year since inception. Even though in more mature markets user growth has stagnated, Android has continued to grow at a remarkable rate through its popularity in developing countries. In Brazil, India, Indonesia, Turkey and Vietnam, Android has over 85 percent market share.

Year	Active users
2012	0.5 billion
2013	0.7 billion
2014	1 billion
2015	1.4 billion
2016	1.7 billion
2017	2 billion
2018	2.3 billion
2019	2.5 billion
2020	2.8 billion

Figure 20:More on UIs

User interface (UI) design is the process designers use to build interfaces in software or computerized devices, focusing on looks or style. Designers aim to create interfaces which users find easy to use and pleasurable. UI design refers to graphical user interfaces and other forms—e.g., voice-controlled interfaces.

User interfaces are the access points where users interact with designs. They come in three formats:

Graphical user interfaces (GUIs)—Users interact with visual representations on digital control panels. A computer's desktop is a GUI.

Voice-controlled interfaces (VUIs)—Users interact with these through their voices. Most smart assistants—e.g., Siri on iPhone and Alexa on Amazon devices—are VUIs.

Gesture-based interfaces—Users engage with 3D design spaces through bodily motions: e.g., in virtual reality (VR) games.

To design UIs best, you should consider:

Users judge designs quickly and care about usability and likeability. They don't care about your design, but about getting their tasks done easily and with minimum effort. Your design should therefore be "invisible": Users shouldn't focus on it but on completing tasks: e.g., ordering pizza on Domino's Zero Click app.

So, understand your users' contexts and task flows (which you can find from, e.g., customer journey maps), to fine-tune the best, most intuitive UIs that deliver seamless experiences. UIs should also be enjoyable (or at least satisfying and frustration-free).

When your design predicts users' needs, they can enjoy more personalized and immersive experiences. Delight them, and they'll keep returning. Where appropriate, elements of gamification can make your design more fun.

- UIs should communicate brand values and reinforce users' trust.
- Good design is emotional design. Users associate good feelings with brands that speak to them at all levels and keep the magic of pleasurable, seamless experiences alive.

A user interface can be judged by three main properties:

- Ease-of-use; how easy it is to use a certain user interface
- Efficiency for inputting information; how fast information can be input through the user interface
- Efficiency for outputting information; how fast the user can process the information coming back from the system

2.7.VUIs and Mobile Apps, Text-to-Speech Technique

Voice user interfaces are user interfaces that are used through speech. Typical examples of voice user interfaces include smart speakers and voice assistants.

Voice UIs employ speech recognition and natural language understanding technologies to transform user speech into text and meaning. Speech is a tool for enhancing traditional touch user interfaces into multimodal voice user interfaces.

Voice user interfaces are highly intuitive as they use the most natural way for us to communicate: speech. They are significantly faster than typing to input information but significantly slower than reading or seeing to output information from the computer system back to the user.

The first voice user interfaces were IVR, Interactive Voice Response, systems that enabled users to interact with a phone system by using speech. Typically, IVRs recognized only digits, but nonetheless they were early voice user interfaces.

Type	Purpose	Output channel	Input channel	Use case
IVR	Specific	Voice	Voice	Narrow
Smart speakers	General	Voice	Voice	Wide
Voice command device (VCD)	Specific	Voice, display	Voice	Narrow
Multimodal voice UIs	Specific	All	All	Wide

The first applications of voice UIs were interactive voice response (IVR) systems that came into existence already back in the 80s. These were systems that understood simple commands through a telephone call and were used to improve efficiency in call centers. Last time I encountered one was a customer support call center for home appliances where a synthesized voice asked me to say the name of the manufacturer of the device I needed help with. Once it recognized correctly my utterance “Siemens”, it directed me to a correct person.

Current voice user interfaces can be a lot smarter and can understand complex sentences and even combinations of them. For example, Google Assistant is perfectly fine with something like “Turn off the living room light and turn on the kitchen light”.

However, as these smart speakers always wait until the end of the user utterance and only then process the information and act accordingly, they will fail if the user hesitates with their speech or says something wrong. For instance, you can try something like “Turn off the kitchen light, turn on the living room TV, turn on the bedroom and terrace lights and turn off the radio in the kitchen.”

2.8.Designing the Right UI

In mobile app development, the UI (user interface) and UX (user experience) play the most significant roles in a mobile app’s success. From the perspective of a designer, an exceptional UI for mobile app can only be ensured when you follow some tried and tested UI design principles. It is always advisable to stick to certain principles that are accepted by the designers’ community. Such UIs not only attract the user, but also help in retaining them in the long run. Check the UI principles listed below:

- **Consistency of the design layout:** Different sections of the app must be coherent in the design, and there must be a consistent flow of the layout throughout the app. Apps like Evernote, Netflix, and Dropbox deliver great consistency in their design layout. This also applies to the common features of the apps built for the same OS.
- **Unambiguous Interactive Elements:** The design elements that are interactive in nature must be clearly depicted. Ambiguity should be avoided to make sure that the users are going to access such elements when required by them. Options hidden in menus may sometime be forgotten. Several surveys and studies have revealed that conspicuous menu options are not good for mobile apps. At least, the key navigation elements must be visible.
- **Single Trial Learning Experience:** The term ‘Intuitive’ is often used to describe mobile apps when suggesting that they are very user-friendly. Actually, this word means that users understand the flow of the app without remembering the steps. Options should be placed in a manner where users can infer what the next step would be.
- **Anticipate and Answer Users’ Queries:** When a user’s tap a button, they want to know whether the process has started and how long will it take. As technology abstracts these actions, the users are usually kept in the dark while the process is going on. Use Toast in Android and

notifications in iOS so users will be notified when a process has completed. The Gmail app is a good example of such a design, as it displays messages like “saved in draft, message sent, etc.” to inform the users about the completion of the task.

- **Layered User Experience:** The UX should be layered. Layered here refers that all the features of an app must not be exposed at once. It should surface out as the users delve deeper into the app and discover new things. This would help in keeping the users’ interest intact for a longer time. WhatsApp offers a good layered experience as users explore new features as they go deeper into the app.

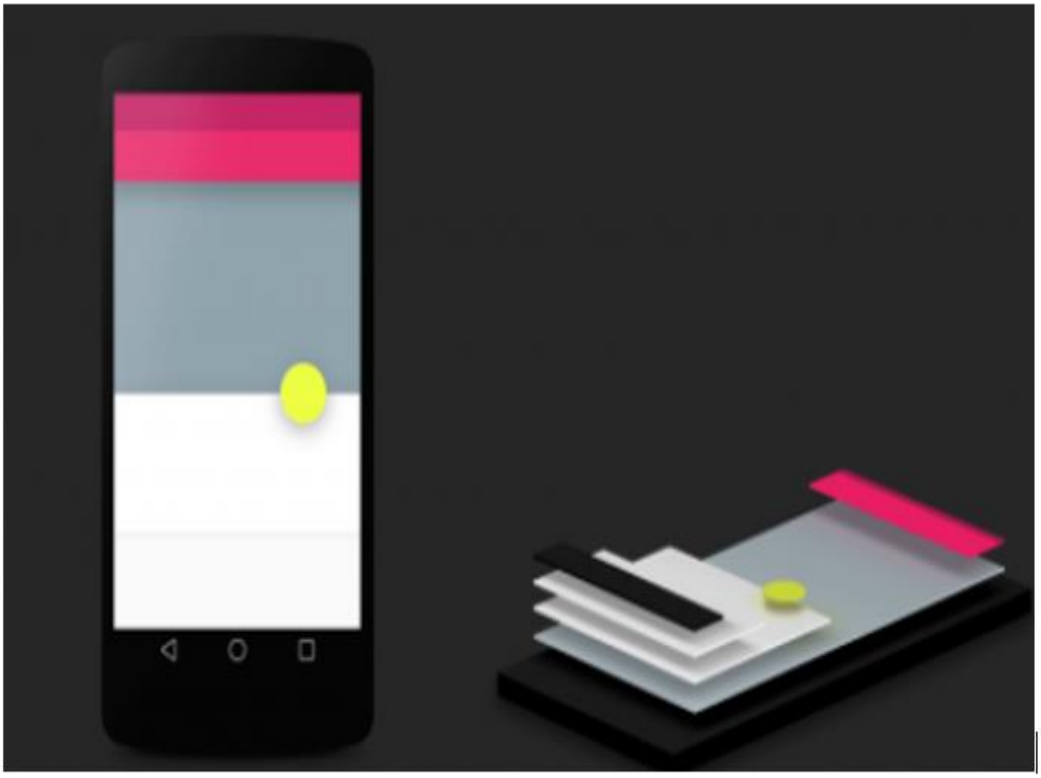


Figure 21:Design an interface for a mobile app with more chances to succeed.

Design an interface for a mobile app with more chances to succeed.

1. Uniqueness



There is no point in creating something that already exists, and it is the same with mobile app design. Your app should definitely have a number of advantages, but presenting them the right way is another story. Take Clear, for example, it is a to-do list, but its interface fundamentally differs from all other to-do lists. There are no buttons and there's no navigation menu, just tasks. Users can only navigate and interact with the app with the help of gestures, which was their unique feature. Uniqueness is one of the most important mobile design principles and that is why it stays on the first place in our list.

2. Structure



User interfaces for mobile applications should be organized in such a way that users will find it helpful and meaningful. Make sure similar items are grouped together and users won't spend tons of time trying to find what they need. You also need to keep in mind that the number of features and sections should not overwhelm the user, so be ready to sacrifice some less relevant features.

3. Contexts



Some apps can be used only on the go, others can be used in a relaxed atmosphere, and some apps can be used in both situations. Context seriously affects mobile user interface design, such as a bigger font, highlighted buttons, and only vital content. On the other hand, the interface can be entertaining and full of engaging options. Before moving on, you need to find out in what context your app will be used and then build the interface according to this information.

4. Gestures



Gestures allow users to navigate apps intuitively. Gestures can also add some gamification and may help you gain particular audiences, like teenagers. Mobile devices are designed for gestures, so try to get the most out of it and become a source of inspiration for other apps. Simultaneously, try use gestures that are familiar to your users and keep in mind how your users will be handling their device and in what context. Let's say if the user is holding the handrail on a public transport with one hand and trying to work an app with the other hand, it might be annoying if there are no other gestures but to pinch to expand, since this gesture requires two hands.

5. Tolerance



App interface design should be created in a way that if some options come up along the way, the system would suggest by default such variants that will be most beneficial to the user. Make sure your app allows users to undo certain actions and there are notifications showing that some changes will be made and can be undone. Tolerance means allowing users to make mistakes and giving them the opportunity to revoke changes.

6. Consistency



If there are several elements serving one and the same purpose in the app, they should look and function the same way. No exceptions. Changing one element means changing all other variants. If there are several styles of “submit” buttons on different pages, it can be frustrating and the app would look messy. Consistency means tidiness and it is certainly a must-have in every app, which is why in the majority of mobile app design companies there’s a rule to create a UI kit for every project. This includes all elements of the app, and it becomes easier to follow the consistency principle with a UI kit.

7. Communication



The core idea here is to let users know your app registered the action and processing it even if it is not possible to give an immediate result. When the user performs a certain action, it can be considered an act of communication with your app, which is why the app should respond with something. Sometimes even the loader sign can be enough. From a psychological perspective, people need to get acknowledgment for their actions, and it is especially important in the banking sphere.

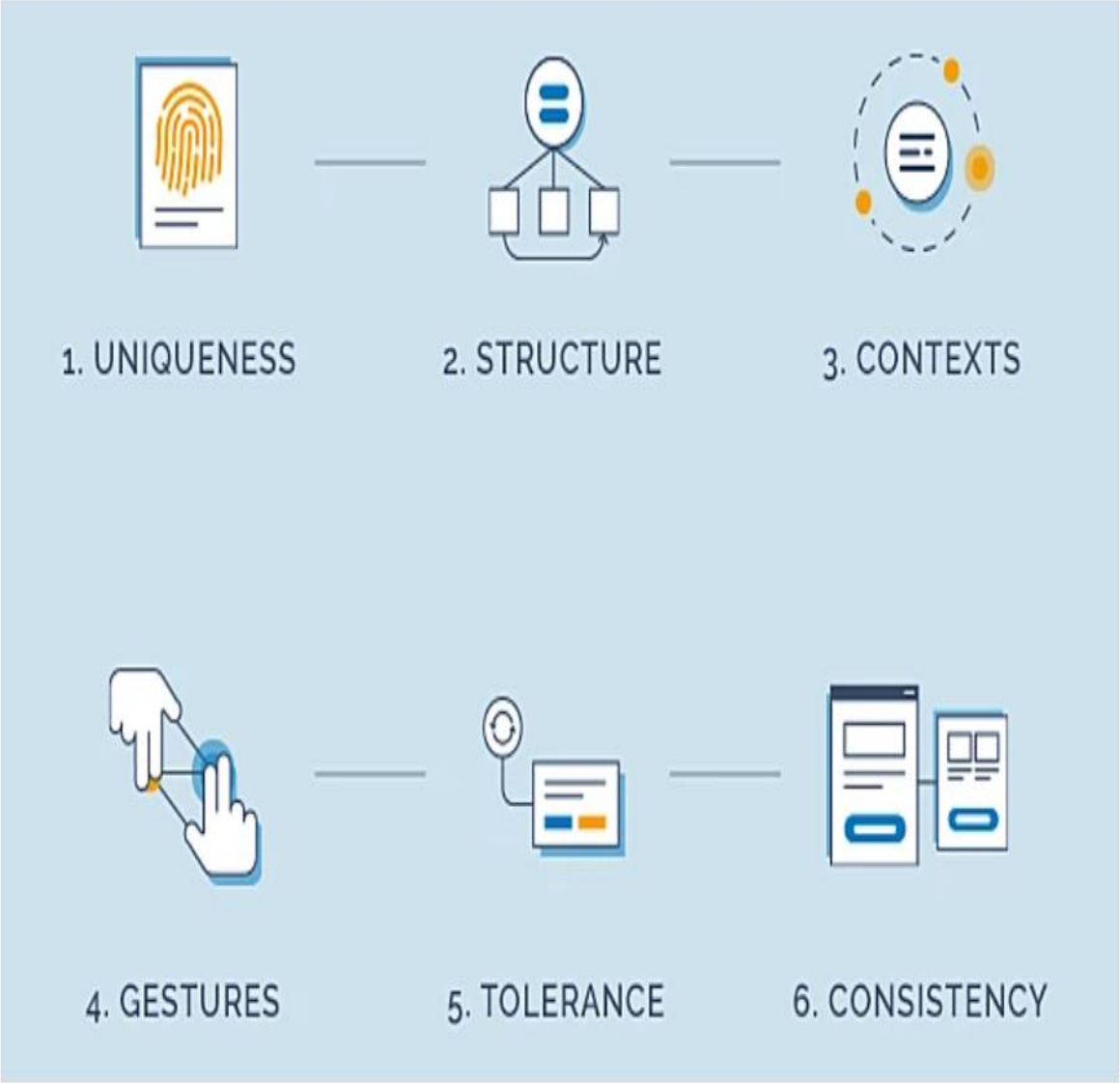


Figure 22:Design of right UIs

2.9.Multichannel and Multimodal UI

One of the breakthroughs users can benefit of is that automatic speech recognition (ASR) improved highly significant over the last years. ASR now works good for dictation tasks. However, dictation is a highly specific use case which does not require the extraction of semantics from the utterances. Some applications use speech input for form filling. However, filling each single slot by speech is often not more efficient then typing. The question arises: What are important challenges in using speech as a "mainstream" modality? While ASR made significant efforts within the last years, e.g. partly driven by the successful application of deep neural networks, the identification of the intended semantic for a further processing by the dialog manager is still a rather difficult process. ASR capabilities are easy to integrate into new

user interfaces by making use of available programming APIs. On the technical side one of the next challenges is therefore to realize conversational speech interaction in many applications. This requires to simplify the usage of NLP methods for information extraction, dialog processing and presentation, so that developers can easily deploy speech interfaces. Since the Internet is mobile nowadays and conversational speech is the most convenient interaction mode of complex applications that require more than simple gestures, this will enable even more services at the hand of the users. In that matter it is important to better understand the specific benefits that emerge for individual users. Information about these benefits can be revealed by observing the users' modality choice behavior. Understanding the factors influencing users' modality choice will enable interface designers to adapt applications to the advantage of the user, and to inform the user about extra possibilities of interaction.

Chapter Three
3. Intents and Services

3.1. Introduction

After Completing this session(chapter), Students must Understand:

- Describe Intents and services
- Explain characteristics of mobile applications
- Discussed on successful mobile development

In our day-to-day life, we all come across a number of mobile applications that runs in the background. Also, in many applications, certain tasks are being performed without using any UI i.e. the task is being performed in the background. For example, the Music app of our mobile device or any other Music application runs in the background and while using the Music app, you can use any other application normally. So, this feature is implemented using the **Service** or **IntentService**.

3.2.Intents and Services

We will look upon the differences between the **Service** and **IntentService**. But before moving forward, we will have a quick revision about **Service** and **IntentService**. So, let's get started.

```
Intent i = new Intent(this, YourIntentService.class);


startService(i);  // For the service.


startActivity(i); // For the activity; ignore this for now.
```

Service

You can think Service as an Android component that is used to perform some long-running operations in the background like that in the Music app, where we run the app in the background and use other applications of the mobile parallelly. The best part is that you don't need to provide some UI for the operations to be performed in the background. By using Service, you can perform some InterProcess Communication (IPC) also. So, with the help of Service, you can perform a number of operations together because any application component can start a Service and can run in background.

There are three ways of using Service:

1. **Foreground:** A foreground service is a Service that will let the user know about what is happening in the background. For example, in the Music application, the user can see the ongoing song on the device as a form of notification. So, here displaying notification is a must.
2. **Background:** Here, the user will never know about what is happening in the background of the application. For example, while sending some images over Whatsapp, Whatsapp compresses the image file to reduce the size. This task is done in background and the user have no idea about what is going in the background. But for the API level 21 or higher, the Android System imposes some restrictions while using the Background Service. So, take care of those restrictions before using the Background Service.
3. **Bound:** The Bound Service is used when one or more than one application component binds the Service by using the **bindService()** method. If the applications unbind the Service, then the Service will be destroyed.

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

There are two types of intents:

- **Explicit intents** specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name. You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.

- **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

  Figure 1 shows how an intent is used when starting an activity. When the Intent object names a specific activity component explicitly, the system immediately starts that component.

  When you use an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the *intent filters* declared in the manifest file of other apps on the device. If the intent matches an intent filter, the system starts that component and delivers it the Intent object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.

  An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive. For instance, by declaring an intent filter for an activity, you make it possible for other apps to directly start your activity with a certain kind of intent. Likewise, if you do *not* declare any intent filters for an activity, then it can be started only with an explicit intent.

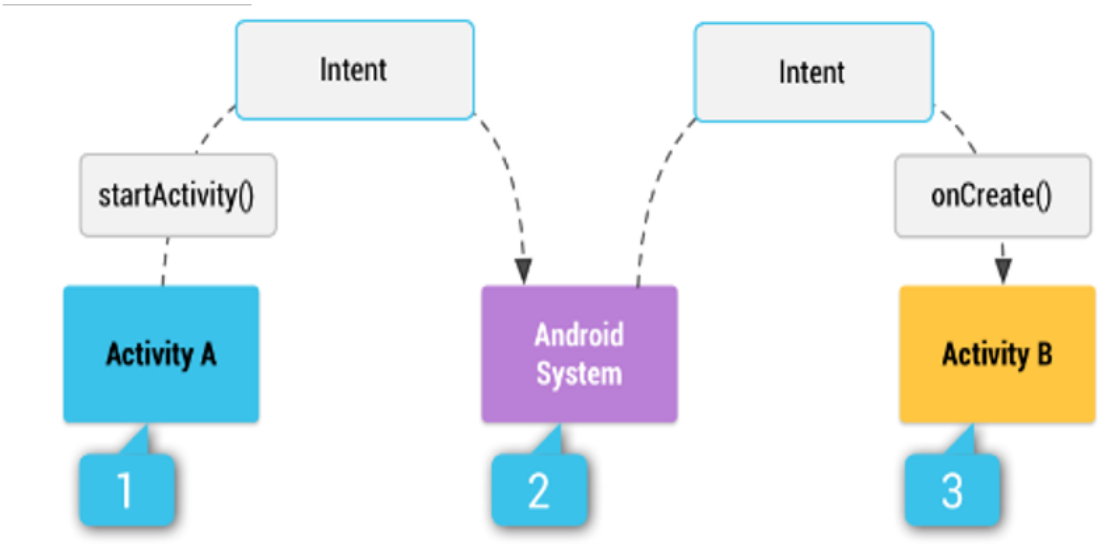


Figure 23:. How an implicit intent is delivered through the system to start another activity:

[1] *Activity A* creates an Intent with an action description and passes it to startActivity(). **[2]**The Android System searches all apps for an intent filter that matches

the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its onCreate() method and passing it the Intent.

- **Starting an activity**

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.

If you want to receive a result from the activity when it finishes, call startActivityForResult(). Your activity receives the result as a separate Intent object in your activity's onActivityResult() callback. For more information, see the Activities guide.

For example, if you have content that you want the user to share with other people, create an intent with the ACTION_SEND action and add extras that specify the content to share. When you call startActivity() with that intent, the user can pick an app through which to share the content.

Example

// Create the text message with a string.

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);

sendIntent.setType("text/plain");

// Try to invoke the intent.

try {

startActivity(sendIntent);

} catch (ActivityNotFoundException e) {

// Define what your app should do if no activity can handle the intent.

}

- **Starting a service**

  A Service is a component that performs operations in the background without a user interface. With Android 5.0 (API level 21) and later, you can start a service with JobScheduler. For more information about JobScheduler.

  For versions earlier than Android 5.0 (API level 21), you can start a service by using methods of the Service class. You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.

  If the service is designed with a client-server interface, you can bind to the service from another component by passing an Intent to bindService(). For more information, see the Services guide.

- **Delivering a broadcast**

  A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().

3.3. Android Intents and Services

Use a PendingIntent instead of a nested intent. That way, when another app unparcels the PendingIntent of its containing Intent, the other app can launch the PendingIntent using the identity of your app. This configuration allows the other app to safely launch any component, including a non-exported component, in your app.

Your app might launch intents to navigate between components inside of your app, or to perform an action on behalf of another app. To improve platform security, Android 12 (API level 31) and higher provide a debugging feature that warns you if your app performs an unsafe

launch of an intent. For example, your app might perform an unsafe launch of a *nested intent*, which is an intent that is passed as an extra in another intent.

If your app performs both of the following actions, the system detects an unsafe intent launch, and a StrictMode violation occurs:

1. Your app unparcels a nested intent from the extras of a delivered intent.
2. Your app immediately starts an app component using that nested intent, such as passing the intent into startActivity(), startService(), or bindService().

The diagram in figure 2 shows how the system passes control from your (client) app to another (service) app, and back to your app:

1. Your app creates an intent that invokes an activity in another app. Within that intent, you add a PendingIntent object as an extra. This pending intent invokes a component in your app; this component isn't exported.
2. Upon receiving your app's intent, the other app extracts the nested PendingIntent object.
3. The other app invokes the send() method on the PendingIntent object.
4. After passing control back to your app, the system invokes the pending intent using your app's context.

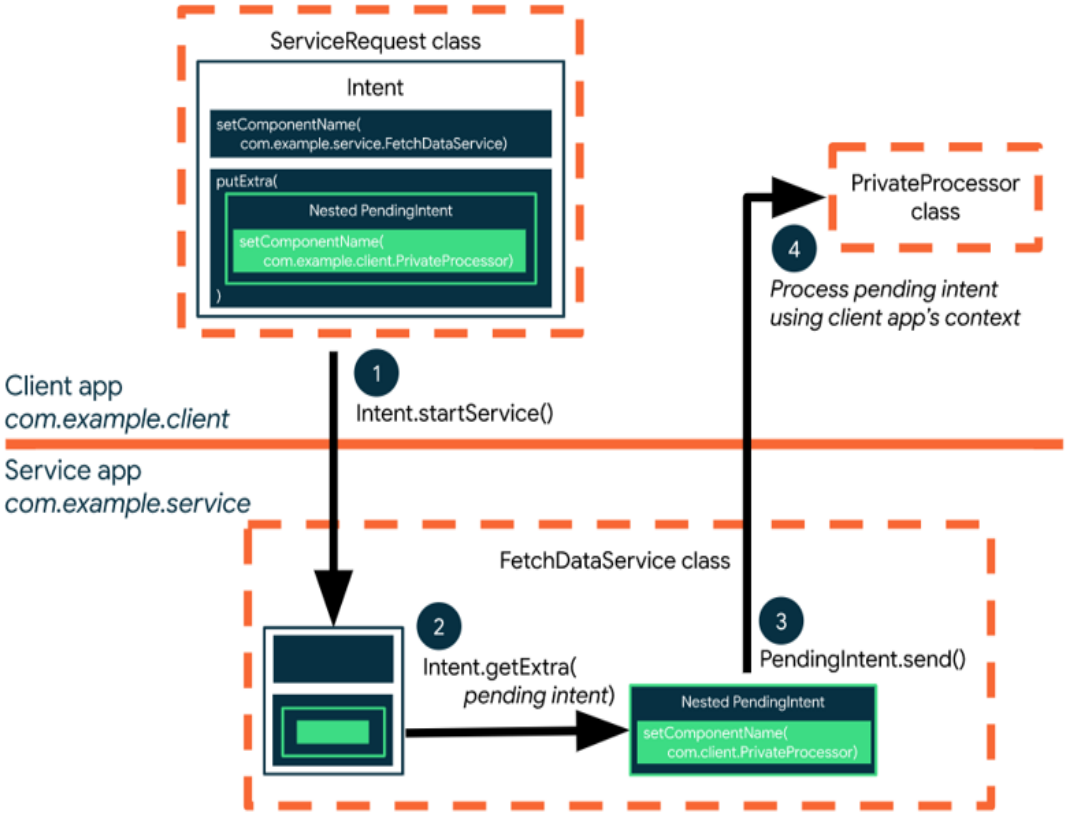


Fig 3.2. Diagram of inter-app communication when using a nested pending intent.

Fig 3.2. Diagram of inter-app communication when using a nested pending intent.

Difference between Service and IntentService

So, we have revised the concept of Service and IntentService. Till now, have you found any difference between these two? If not, don't worry, in this section, we will look upon some of the differences between the Service and IntentService, so that it will be easier for you to find which one to use in which condition. Let's see the difference, point wise:

1. **Usage:** If you want some background task to be performed for a very long period of time, then you should use the IntentService. But at the same time, you should take care that there is no or very less communication with the main thread. If the communication is required then you can use main thread handler or broadcast intents. You can use Service for the tasks that don't require any UI and also it is not a very long running task.

2. **How to start?:** To start a Service you need to call the **onStartService()** method while in order to start a start IntentService you have to use Intent i.e. start the IntentService by calling **Context.startService(Intent).**
3. **Running Thread:** Service always runs on the **Main thread** while the IntentService runs on a separate **Worker thread** that is triggered from the Main thread.
4. **Triggering Thread:** Service can be triggered from any thread while the IntentService can be triggered only from the Main thread i.e. firstly, the Intent is received on the Main thread and after that, the Worker thread will be executed.
5. **Main Thread blocking:** If you are using Service then there are chances that your Main thread will be blocked because Service runs on the Main thread. But, in case of IntentService, there is no involvement of the Main thread. Here, the tasks are performed in the form of Queue i.e. on the First Come First Serve basis.
6. **Stop Service:** If you are using Service, then you have to stop the Service after using it otherwise the Service will be there for an infinite period of time i.e. until your phone is in normal state. So, to stop a Service you have to use **stopService()** or **stopSelf()**. But in the case of IntentService, there is no need of stopping the Service because the Service will be automatically stopped once the work is done.
7. **Interaction with the UI:** If you are using IntentService, then you will find it difficult to interact with the UI of the application. If you want to out some result of the IntentService in your UI, then you have to take help of some Activity.

Service	Intent Service
If the task doesn't require any and also not a very long task you can use service.	If the Background task is to be performed for a long time we can use the intent service.
we use the method **onStartService()** to start the service	we use the method **Context.startService(Intent)** to start the intent service
Service will always run on the main thread.	intent service always runs on the worker thread triggered from the main thread.
There is a chance of blocking the main thread.	tasks will be performed on a queue basis i.e, first come first serve basis.
To stop service we have to use **stopService()** or **stopSelf()**	No need to stop the service, it will stop automatically.
Easy to interact with the UI of the application.	Difficult to interact with the UI of the application.

3.4. Characteristics of Mobile Applications

Mobile is a rapidly growing industry that attracts companies from many industries. Because of the growing popularity of smartphones and tablets, mobile application development is becoming more popular among business owners all over the world. A mobile application is a software program that runs on a mobile device, such as a smartphone or tablet computer. Despite being little software units with limited functionality, apps manage to provide high-quality services and experiences to users. **App Development Services** can be a little expensive for you without making any difference.

3.5. Successful Mobile Development

In today's fast paced world, the mobile app market is expanding by leaps and bounds. Consequently, mobile marketing strategies are becoming more competitive. To ensure the visibility of apps in such a complex scenario, app makers need to be very particular about the

approach being followed for mobile app development (including game apps). To create a successful mobile application, you need to follow a systematic approach to the mobile app development lifecycle. We have summarized 10 steps to create a successful mobile application to help you out in this process.

Finding the right development partner is the most crucial step in the process. Understanding the technology stack, development challenges and other technical aspects can be difficult. It can be troubling for entrepreneurs without tech backgrounds. So, find an app development partner that takes care of your project professionally, without you having to look into everything.

You can choose between:

• **Freelancers:** You can easily find freelancers who fit in your budget. However, freelancers can forge their experience or portfolio, or they may have limited exposure to diverse industries. This option can come with a lot of risks, but if you are on a budget, then it's for you.

• **In-House Development:** You can hire an in-house development team for your project. This option is a major investment, including recruitment costs, employee benefits, operational costs for the team and more. However, you may have more control over in-house development teams in terms of delivery and deadlines.

• **App Development Agencies:** With diverse portfolios and experienced engineers, app development agencies are affordable, experienced, and professional. You don't have to chase them as you might have to with freelancers. Agencies have expert developers, designers, project managers, creative artists and data scientists. However, not all app development agencies are experienced in modern technologies.

• **Silicon Giants:** You can hand over your project to silicon giants like Oracle and IBM. But this elite app development is rarely affordable for startups.

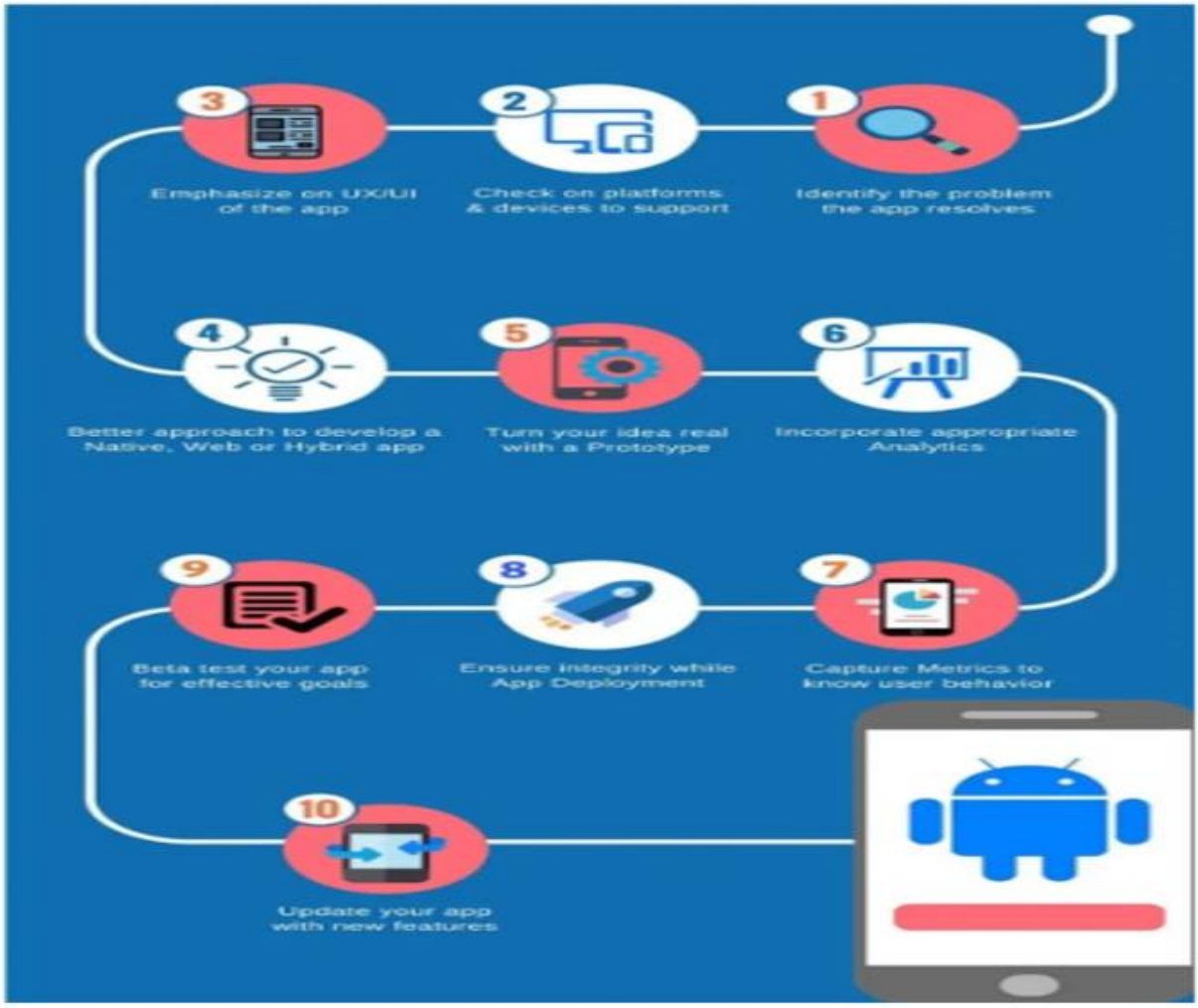


Figure 24:Steps to follow for a successful mobile application

Step 1: **A successful mobile app starts with an app idea**

To create a successful mobile application, the first thing you need to keep in mind is:

- Identify a problem which can be resolved by your app
- Decide the features of your app

The app should provide customer with tangible benefits including reducing costs via productivity enhancements, new revenue or improving the customer experience.

Step 2: Identification / Clarification

To create a successful mobile application, you need to identify or be clear about:

- **Application target users –** An app should always be developed keeping in mind the target users of an application. Having a clear vision regarding the target group, enhance the success ratio of an app.
- **Mobile platforms and devices to be supported –** Mobile platforms and devices should be selected keeping in mind hardware performance, battery life, ruggedness and required peripherals. Certain factors that needs to be considered while selecting mobile platforms and devices includes coverage, device support, performance and other features.
- **Revenue model –** The app market is booming like never before. To ensure this resource and **generate revenue** , app developer need to select appropriate approach in accordance with the app. There are different models of **generating revenue** from **mobile applications** which include paid applications, separate app and in-app freemiums, advertisements, subscription and pay per download.These techniques can be employed to generate revenue. However, the developer's approach has to be in accordance with the application. It is highly essential for the developer to attract the user and spend money on the various aspects of the application.



Figure 25:Designing your app

Designing your app is yet another significant factor responsible for success of an app in the market. Remember, a good UX design and good UI-UX means good discoverability. An app developer should concentrate on the UI design, multi-touch gestures for touch-enabled devices and consider platform design standards as well. Today, emphasis is on the **UI design of an app** as it plays a crucial role in the success of an app. A number of drag and drop app builders

are available in the market. Designing an app is becoming increasingly popular as it create an instant impact on the mind of the user while ensuring usability of an app.

Step 4: Identify approach to develop the app – native, web or hybrid

Selecting the right approach for developing an app is highly important. Ideally, the backend development of the app must be in accordance with the time and budget constraints of a client. A number of app developers prefer to follow the agile methodology.

- **Native:** Native apps enables in delivering the best user experience but require significant time and skill to be developed. These apps are basically platform specific and require expertise along with knowledge. Native apps are costly as well as time taking to be developed and deliver the highest user experience amongst all the approaches.
- **Web:** Web apps are quick and cheap ones to develop and can run on multiple platforms. These are developed using HTML5, CSS and JavaScript code. These web apps are less powerful than native apps.
- **Hybrid:** Hybrid approach is the latest approach to develop any app. This approach combines prebuilt native containers with on-the-fly web coding in order to achieve the best of both worlds. In this approach, the developer augments the web code with native language to create unique features and access native APIs which are not yet available through JavaScript.

Step 5: Make an app prototype

Next stage, after identifying the approach is developing a prototype. It is actually the process of taking your idea and turning it into an application with some basic functionality. A prototype makes it quite easier to sell your idea to potential buyers who can now actually view the tangible benefits instead of just visualizing or reading product description. It is quite helpful in attracting investors and working with manufacturers and finding licensees. You can also share the prototype app with testers for functional testing and get an idea of what needs to change to make it a successful mobile application.

Even while working on a prototype, do ensure you take measures to **secure your app** against unauthorized usage and access to data.

Step 6: Integrate an appropriate analytics tool



There is also a need to incorporate appropriate analytics which gives you a detailed picture of how many visitors use your webs, how they arrived on your site and how can they keep coming back.

Some of the mobile analytics tool which helps in this process:

- **Google Analytics**
- Firebase
- Mixpanel
- Preemptive

With **data sciences**, including **predictive analytics** coming up in mobile apps, it can make your apps highly marketable.

Step 7: Identify your testers: Listen to them and incorporate relevant feedback

Beta testing is the first opportunity to get feedback from your target customers. It is especially important as it enhances your visibility in the app store. It not only reduce product risk but get you that initial push in the app store. To identify beta testers is another important task to have a successful mobile app

Preparing for beta launch:

- **Define target customer –** It is highly important to identify and clearly define your target audience. This will enable you identify the right testers during your beta tester recruiting.

Early **market research** helps in understanding market analysis which eases the process of beta testing.

- **Eliminate bugs –** Before beta testing your app on different platforms you need to take into account majority of the devices which eliminate device specific bugs. Alpha testing with a small number of users enables to clear out maximum bugs. At the same time, device coverage plan is significant for quality assurance of mobile app.
- **Identify goals –** Beta testing is the best opportunity to get real feedback from target customers. It provides a great opportunity to further understand target market and their requirements. Identifying goals for beta testing helps in focusing the efforts. These goals reduce your product launch risk.

Step 8: Release/Deploy the app

Deploying an app requires plan, schedule and control of the movement of releases to test and live environments.

Step 9: Capture the metrics

There has been significant rise in the mobile app users in the present decade. As a result, the need to collect accurate metrics is highly important. As the number of consumers using **mobile applications** steadily rises, the need to collect accurate metrics from them is increasingly important. Unfortunately, many of the methods used to **measure apps** are taken from web analytics.

Major input metrics which should be kept in mind:

- Funnel analysis signifies as to why users are failing to complete desired user actions including in-app purchases or ad clicks
- Measuring social sharing signifies what aspects of your app are capturing the attention of your users
- Correlating demographic data with user behavior
- Tracking time and location gives you insights into the contexts in which your app is used
- Finally, capturing the emergent behavior of your user base is critical

Step 10: Upgrade your app with improvements and new features



After capturing the **metrics** it becomes important to upgrade your app with improvements and innovative features. A mobile app without innovative features loses its usability in long run. Upgrading your app with innovative features enhances its visibility along with downloads of an app. Also ensure you keep **updating your app** to meet new guidelines offered by the various platforms, don't let your apps stagnate.

These are some of the of steps which should be taken into account while developing an app. Using these steps, you can develop an app ensuring success in long run. However, it is nearly impossible to pen down the exact steps which are responsible for success of an app.

Bonus Step 11: Market your app right

While this doesn't fall under the steps to "make" an app, it is definitely important to make your app successful especially for small businesses. If you do not **market your app** well once it is released, there is a high possibility of it being lost in the multitude of apps available on the various android and apple app stores. So, make sure your **market your app** well. **Social media** can be a useful tool here. This a bonus step to create a successful mobile application but it is highly recommended that you use it for your own benefit.

Chapter Four

4. Storing and Retrieving Data

4.1. Introduction

After completing this Session(chapter), Students must able to:

- Explain how to store and retrieving data using mobile application.
- Describe Synchronization and Replication of Mobile Data
- Understand working with a content provide

Android provides several options for you to save persistent application data. The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires. Your data storage options are the following:

Shared Preferences

Store private primitive data in key-value pairs.

Internal Storage

Store private data on the device memory.

External Storage

Store public data on the shared external storage.

SQLite Databases

Store structured data in a private database.

Network Connection

Store data on the web with your own network server.

Android provides a way for you to expose even your private data to other applications — with a content provider. A content provider is an optional component that exposes read/write

access to your application data, subject to whatever restrictions you want to impose. For more information about using content providers.

Using Shared Preferences

The Shared Preferencesclass provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use Shared Preferencesto save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).

User Preferences

Shared preferences are not strictly for saving "user preferences," such as what ringtone a user has chosen. If you're interested in creating user preferences for your application, see Preference Activity, which provides an Activity framework for you to create user preferences, which will be automatically persisted (using shared preferences).

To get a Shared Preferencesobject for your application, use one of two methods:

- getSharedPreferences() – Use this if you need multiple preferences files identified by name, which you specify with the first parameter.
- getPreferences() – Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

To write values:

1. Call edit()to get a SharedPreferences.Editor.
2. Add values with methods such as putBoolean()and putString().
3. Commit the new values with commit()

To read values, use SharedPreferencesmethods such as getBoolean()and getString().

Here is an example that saves a preference for silent keypress mode in a calculator:

```
public class Calc extends Activity {
```

```
public static final String PREFS_NAME = "MyPrefsFile";

@Override

protected void onCreate(Bundle state){

   super.onCreate(state);

   // Restore preferences

   SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);

   boolean silent = settings.getBoolean("silentMode", false);

   setSilent(silent);

}

@Override

protected void onStop(){

   super.onStop();

  // We need an Editor object to make preference changes.

  // All objects are from android.context.Context
```

```
    SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);

    SharedPreferences.Editor editor = settings.edit();

    editor.putBoolean("silentMode", mSilentMode);

    // Commit the edits!

    editor.commit();

  }

}
```

Using the Internal Storage

You can save files directly on the device's internal storage. By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user). When the user uninstalls your application, these files are removed.

To create and write a private file to the internal storage:

1. Call openFileOutput()with the name of the file and the operating mode. This returns a FileOutputStream.
2. Write to the file with write().
3. Close the stream with close().

For example:

```
String FILENAME = "hello_file";
```

```
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);

fos.write(string.getBytes());

fos.close();
```

MODE_PRIVATE will create the file (or replace a file of the same name) and make it private to your application. Other modes available
are: MODE_APPEND, MODE_WORLD_READABLE,
and MODE_WORLD_WRITEABLE.

To read a file from internal storage:

1. Call openFileInput()and pass it the name of the file to read. This returns a FileInputStream.
2. Read bytes from the file with read().
3. Then close the stream with close().

Tip: If you want to save a static file in your application at compile time, save the file in your project res/raw/directory. You can open it with openRawResource(), passing the R.raw.<*filename*>resource ID. This method returns an InputStream that you can use to read the file (but you cannot write to the original file).

Saving cache files

If you'd like to cache some data, rather than store it persistently, you should use getCacheDir()to open a Filethat represents the internal directory where your application should save temporary cache files.

When the device is low on internal storage space, Android may delete these cache files to recover space. However, you should not rely on the system to clean up these files for you.

You should always maintain the cache files yourself and stay within a reasonable limit of space consumed, such as 1MB. When the user uninstalls your application, these files are removed.

Other useful methods

getFilesDir()

Gets the absolute path to the filesystem directory where your internal files are saved.

getDir()

Creates (or opens an existing) directory within your internal storage space.

deleteFile()

Deletes a file saved on the internal storage.

fileList()

Returns an array of files currently saved by your application.

Using the External Storage

Every Android-compatible device supports a shared "external storage" that you can use to save files. This can be a removable storage media (such as an SD card) or an internal (non-removable) storage. Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

Caution: External files can disappear if the user mounts the external storage on a computer or removes the media, and there's no security enforced upon files you save to the external storage. All applications can read and write files placed on the external storage and the user can remove them.

Checking media availability

Before you do any work with the external storage, you should always call getExternalStorageState()to check whether the media is available. The media might be mounted to a computer, missing, read-only, *or in some other state. For example, here's how you can check the availability:*

```
boolean mExternalStorageAvailable = false;

boolean mExternalStorageWriteable = false;

String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {

    // We can read and write the media

    mExternalStorageAvailable = mExternalStorageWriteable = true;

} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {

    // We can only read the media

    mExternalStorageAvailable = true;

    mExternalStorageWriteable = false;
```

```
} else {

   // Something else is wrong. It may be one of many other states, but all we need

   //  to know is we can neither read nor write

   mExternalStorageAvailable = mExternalStorageWriteable = false;

}
```

This example checks whether the external storage is available to read and write.
The getExternalStorageState()method returns other states that you might want to check, such as whether the media is being shared (connected to a computer), is missing entirely, has been removed badly, etc. You can use these to notify the user with more information when your application needs to access the media.

Accessing files on external storage

If you're using API Level 8 or greater, use getExternalFilesDir()to open a Filethat represents the external storage directory where you should save your files. This method takes a `type`parameter that specifies the type of subdirectory you want, such asDIRECTORY_MUSICand DIRECTORY_RINGTONES(pass `null`to receive the root of your application's file directory). This method will create the appropriate directory if necessary. By specifying the type of directory, you ensure that the Android's media scanner will properly categorize your files in the system (for example, ringtones are identified as ringtones and not music). If the user uninstalls your application, this directory and all its contents will be deleted.

If you're using API Level 7 or lower, use getExternalStorageDirectory(), to open a Filerepresenting the root of the external storage. You should then write your data in the following directory:

```
/Android/data/<package_name>/files/
```

The <*package_name*>is your Java-style package name, such as "com.example.android.app". If the user's device is running API Level 8 or greater and they uninstall your application, this directory and all its contents will be deleted.

Hiding your files from the Media Scanner

Include an empty file named .nomediain your external files directory (note the dot prefix in the filename). This will prevent Android's media scanner from reading your media files and including them in apps like Gallery or Music.

Saving files that should be shared

If you want to save files that are not specific to your application and that should *not* be deleted when your application is uninstalled, save them to one of the public directories on the external storage. These directories lay at the root of the external storage, such asMusic/, Pictures/, Ringtones/, and others.

In API Level 8 or greater, use getExternalStoragePublicDirectory(), passing it the type of public directory you want, such as DIRECTORY_MUSIC, DIRECTORY_PICTURES, DIRECTORY_RING ONES, or others. This method will create the appropriate directory if necessary.

If you're using API Level 7 or lower, use getExternalStorageDirectory()to open a Filethat represents the root of the external storage, then save your shared files in one of the following directories:

- Music/ – Media scanner classifies all media found here as user music.

- Podcasts/ – Media scanner classifies all media found here as a podcast.
- Ringtones/ – Media scanner classifies all media found here as a ringtone.
- Alarms/ – Media scanner classifies all media found here as an alarm sound.
- Notifications/ – Media scanner classifies all media found here as a notification sound.
- Pictures/ – All photos (excluding those taken with the camera).
- Movies/ – All movies (excluding those taken with the camcorder).
- Download/ – Miscellaneous downloads.

Saving cache files

If you're using API Level 8 or greater, use getExternalCacheDir()to open a Filethat represents the external storage directory where you should save cache files. If the user uninstalls your application, these files will be automatically deleted. However, during the life of your application, you should manage these cache files and remove those that aren't needed in order to preserve file space.

If you're using API Level 7 or lower, use getExternalStorageDirectory()to open a Filethat represents the root of the external storage, then write your cache data in the following directory:

```
/Android/data/<package_name>/cache/
```

The <*package_name*>is your Java-style package name, such as "com.example.android.app".

Using Databases

Android provides full support for SQLite databases. Any databases you create will be accessible by name to any class in the application, but not outside the application.

The recommended method to create a new SQLite database is to create a subclass of SQLiteOpenHelperand override theonCreate()method, in which you can execute a SQLite command to create tables in the database. For example:

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {

   private static final int DATABASE_VERSION = 2;

   private static final String DICTIONARY_TABLE_NAME = "dictionary";

   private static final String DICTIONARY_TABLE_CREATE =

            "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +

            KEY_WORD + " TEXT, " +

            KEY_DEFINITION + " TEXT);";

   DictionaryOpenHelper(Context context) {

      super(context, DATABASE_NAME, null, DATABASE_VERSION);

   }

   @Override

   public void onCreate(SQLiteDatabase db) {
```

```
    db.execSQL(DICTIONARY_TABLE_CREATE);

  }

}
```

You can then get an instance of your SQLiteOpenHelperimplementation using the constructor you've defined. To write to and read from the database, call getWritableDatabase()and getReadableDatabase(), respectively. These both return a SQLiteDatabaseobject that represents the database and provides methods for SQLite operations.

Android does not impose any limitations beyond the standard SQLite concepts. We do recommend including an autoincrement value key field that can be used as a unique ID to quickly find a record. This is not required for private data, but if you implement acontent provider, you must include a unique ID using the BaseColumns._IDconstant.

You can execute SQLite queries using the SQLiteDatabasequery()methods, which accept various query parameters, such as the table to query, the projection, selection, columns, grouping, and others. For complex queries, such as those that require column aliases, you should use SQLiteQueryBuilder, which provides several convienent methods for building queries.

Every SQLite query will return a Cursorthat points to all the rows found by the query. The Cursoris always the mechanism with which you can navigate results from a database query and read rows and columns.

For sample apps that demonstrate how to use SQLite databases in Android.

Database debugging

The Android SDK includes a sqlite3database tool that allows you to browse table contents, run SQL commands, and perform other useful functions on SQLite databases.

Using a Network Connection

You can use the network (when it's available) to store and retrieve data on your own web-based services. To do network operations, use classes in the following packages:

- java.net.*
- android.net.*

4.2. Synchronization and Replication of Mobile Data

Data synchronization is a method of establishing consistency among data from a data source to the target data storage and vice versa. In data synchronization, we have to keep multiple copies of a dataset in coherence with one another to maintain the data integrity.

Data synchronization provides the continuous harmonization of the data over time. This is the basic fundamental concept used in a wide variety of applications, including file synchronization and mobile device synchronization, e.g., PDAs. It is also used in encryption for synchronizing Public Key Servers.

Data synchronization ensures accurate, secure, compliant data and successful team and customer experiences. It assures congruence between each source of data and its different endpoints. As data comes in, it is cleaned, checked for errors, duplication, and consistency before being put to use. Local synchronization involves devices and computers that are next to each other, while remote synchronization takes place over a mobile network.

Data must always be consistent throughout the data record. If data is modified in any way, changes must upgrade through every system in real-time to avoid mistakes, prevent privacy breaches, and ensure that the most up-to-date data is the only information available. Data synchronization ensures that all records are consistent, all the time.

Data synchronization is important and required in mobile computing because it checks the differences between two data containers or data sources and data receivers to restrict the unnecessary transfer of data that already resides in both data sources. The data synchronization process typically updates both data sources by transferring only additions, changes, and

deletions. The following are the reasons why data synchronization is required in Mobile computing:

- Data synchronization is required between the mobile devices and their service provider.
- It is also required between the device and personal area computer and nearby wireless access points (in Wi-Fi connection) and other nearby devices.
- It is used to establish consistency among data from a data source to the target data storage and vice versa.
- Example of Data Synchronization

Suppose we have added a new popular ringtone to one of the servers of a mobile service provider, so here, data synchronization means that all the service provider servers get identical sets of ringtones. All the devices connected to the server should be updated about the availability of the new data. The ringtone databases available to all the mobile phones include a copy of the title of that tone.

The importance of data synchronization grows in step with increased accessibility to cloud-based data as well as access to mobile devices. Mobile devices use data for basic operation as well as personal information for apps, websites, and email. Updates to information generated by the user as well as the end target must be constant and secure. This synchronization process requires clean, consistent data for product and service competence, but also for data governance issues like security and regulatory compliance.

Data conflicts can result in errors and low data quality, which consequently leads to a lack of trusted data down the line. With data synchronization properly implemented throughout a system, a business will see performance improvement in many areas, including:

- Logistics and transportation
- Sales team productivity
- Order management
- Invoice accuracy
- Business systems
- Cost efficiency
- Reputation management

Data accessibility and error resolution afford time savings, allowing emphasis to be put on important business development processes like marketing, new product development, and strategic decision-making. Virtually everyone benefits from clean, synced data.

- Customers receive product information and service that meets their specific needs
- Business users can interact with all department members using up-to-date information, in real-time, even globally
- Executives receive the latest data when making important strategy decisions
- Stockholders can easily stay on top of their business interests
- Manufacturers access the most recent updates or changes for accurate design and production
- Distributors have access to the most recent product and marketing information

Exploring data synchronization methods

There are a few types of data synchronization methods. Version control and file synchronization tools can make changes to more than one copy of a file at a time. The other two, distributed and mirror, have more specific uses.

File Synchronization: Faster and more error-proof than a manual copy technique, this method is most used for home backups, external hard drives, or updating portable data via flash drive. File synchronization ensures that two or more locations share the same data, occurs automatically, and prevents duplication of identical files.

Version Control: This technique aims to provide synchronizing solutions for files that can be altered by more than one user at the same time.

Distributed File Systems: When multiple file versions must be synced at the same time on different devices, those devices must always be connected for the distributed file system to work. A few of these systems allow devices to disconnect for short periods of time, as long as data reconciliation is implemented before synchronization.

Mirror Computing: Mirror computing is used to provide different sources with an exact copy of a data set. Especially useful for backup, mirror computing provides an exact copy to just one other location — source to target**.**

4.3. Getting the Model Right

Every business wants to provide services to their customers, wherever they are and whatever they are doing. Some do this through their website; others do use a mobile app. The challenge is that there is a multitude of different devices used by customers. So which devices should the app be supported on? Can all be supported? Are there features on some devices that could be used to make the app more engaging? Do we have the right skillsets in-house to develop an app that can be supported on all these different platforms? In addressing these questions, a number of different mobile app development models have sprung up. If you are reviewing what mobile development model will be right for you, this article examines four models and offers pros and cons to each and also an alternative option.

Responsive Web App

In this case, mobile app development devolves down to web app development, but uses techniques to produce a responsive web design that automatically adapts to the profile of the device that it is being displayed on. Web apps that are designed using responsive techniques make use of new HTML5 and CSS3 properties such as media queries:

@media screen and (min-width:500px)

Using these media queries will allow the web app to automatically determine the size of the available screen real estate that the app will run in and adapt to it, by, for example, applying a different set of styles. The JavaScript code that makes up the logic of the app can largely remain the same, regardless of the device the app is running on.

This model is a fantastic way to create mobile apps quickly and easily, giving you faster time to market for your app. Furthermore, the cost of development will be kept low as you are likely to already have the skills in-house to develop apps, or will be able to select from a large and available talent pool. However, the disadvantage of this approach is that the app will not have access to underlying features of the device that it is running on, such as the GPS for geo-location or camera. This means that the depth of functionality will be limited across all devices and the responsive web app is diluted to a 'lowest common denominator' of functionality that will work on any device/browser platform. Moreover, the user is required to type a URL into the mobile browser to access the web app. It is possible to create bookmarks to apps that display

as icons on the home screen (like a native app), but you are still left with a browser-based user experience.

Pros – • Create mobile apps quickly and easily

• Cost of development will be kept low

Cons – • Not have access to underlying features of the device that it is running on, such as the GPS for geo-location or camera

• User still left with a browser-based user experience

Hybrid-Hybrid Web App A different approach is to take a web app writteninHTML5/JavaScript and wrap it in a native container app such as Apache Cordova. This approach has the benefit of removing the browser from the user experience and it means that the web app code can be installed from an app store in exactly the same way as any other mobile app. All the elements of the UI and app logic are handled in the web app's HTML, CSS and JavaScript code, but now some device functions such as geo-location and device storage may be accessible to the web app's JavaScript via the native app container. This means that a more feature-rich app is possible across a variety of device platforms depending on what platforms are supported by the app container. The downsides of this approach are two-fold: 1. The execution environment of the web app JavaScript within the native app container may not be optimal for the device it is running on. This means that a comparable native app would offer better performance over the hybrid web app. This is most often manifested in the responsiveness of the UI. 2.The UI elements rendered by the web app's CSS may not be representative or up-to-date with the native UI elements that a native app would have. This could lead to an inconsistent UX across different devices or different versions of the OS (think iOS 6 vs. iOS 7). However, the advantage to this approach is that mobile app development time remains shorter and predominantly makes use of readily available skills that may already be available in-house. The native app containers can often be used across different device platforms with little or no code changes.

Pros – • removing the browser from the user experience

• web app code can be installed from an app store in exactly the same way as any other mobile app • mobile app development time remains shorter and predominantly makes use of readily available skills that may already be available in-house

Cons – • hybrid app may not be optimal for the device it is running on

• UI elements rendered by the web app's CSS may not be representative or up-to -date with the native UI elements

Hybrid App II – Hybrid Mixed App

A more sophisticated approach is to take the hybrid approach (described above), but infuse it with more platform-specific native code, to take advantage of more platform features. In this hybrid model, the app logic is split between the web app code and the native container. This allows for tighter integration between the app and the device hardware and can also improve the performance of some of the more intensive aspects of the application logic. The UI can still be handled by the web app code using HTML5, CSS3 and JavaScript although this still retains the disadvantage of the earlier hybrid model and the responsive web app. Whilst there are clear advantages to this model in terms of the tighter integration between app and device whilst retaining common app logic in JavaScript, there is an increased need for more platform-specific skills for each platform that the app is intended to be used on. By keeping the platform-specific code within the same code base as the common app code, this hybrid approach remains viable as faster go-to market strategy for development than a fully native app model that requires separate code bases and skills for each platform that is to be supported.

Pros – • tighter integration between the app and the device hardware

• improve the performance of some of the more intensive aspects of the application logic

• hybrid approach remains viable as faster go-to market strategy for development

Cons – • increased need for more platform-specific skills for each platform

• retains the disadvantage of the earlier hybrid model and the responsive web app

Native App The native app development model has the application written in the native language and environment for each platform, i.e. Objective-C on iOS for Apple, Java on

Android, C# on Windows Phone, etc. This route offers the best performance and provides full access to the underlying device hardware. However, it comes at a significant cost in terms of development: a separate app and code base needs to be maintained for each platform and each platform requires different skill sets and may necessitate the need for separate development teams for each. A significant advantage to this approach is that the performance of the mobile app is often superior to its hybrid equivalent. Furthermore, the apps UI elements will be guaranteed to be consistent with the platform look and feel. A development organization that already has separate mobile development teams for each platform supported will already have all the necessary skills to support this development model, without having to invest in additional resources with HTML5 skills.

Pros – • best performance and provides full access to the underlying device hardware

• performance of the mobile app is often superior to its hybrid equivalent

• UI elements will be guaranteed to be consistent with the platform look and feel.

Cons – • comes at a significant cost in terms of development

One Dev Platform to Rule Them All With so many options for developing mobile apps you might be forgiven for thinking that you need to adopt a single approach and invest in the skills and tools required to support that approach. But surprisingly there are tools on the market that can address all of the models described above, which is great because each model has its pros and cons. You may develop your next app as a hybrid app, but the one after that may require the extra performance and features of a native app. What should you look for in a development environment to support mobile app development?

• It should use and support tools that are already in common development usage today. Developers won't need to learn a whole new set of tooling and processes to just get stuff done.

• It should support all the development models that you need to get your apps to market as quickly as possible.

• Getting quality apps to market quickly is crucial so the development environment must support device emulators as well as browser-based simulators and it should provide an easy, scriptable way of cycling through tests across multiple device types.

• All the latest native platform SDKs must be supported from day one of their release.

• Trust and security is paramount, particularly when developing enterprise apps for employees, affiliates or partners. Any mobile development environment must support strong security principles that do not leave data on the device exposed. It should avoid exposing business critical enterprise data to the Internet and does not allow scope for malicious trojan horse apps to masquerade as the real thing.

• Integrating mobile apps to data, either in an enterprise or in the cloud, enables apps to be relevant, personalized and more engaging to the user. A mobile app development environment must make it easy to integrate the app to data.

• And if that data is constantly changing in response to events such as sports, M2M, news or social, then apps need to reflect the new state of the world in near real-time. An app development environment must make it easy to write apps that are "event-driven".

4.4. Android Storing and Retrieving Data

Data has always been the most important part of any mobile application. Developers need to store substantial amounts of data in an organized fashion in order for their apps to be truly valuable to their users. In Android, you can store your data using quite a few different methods, both in active memory and in written files. Many apps also have a remote web service that provide the data for the app in question. Android also supports data storage in a local database, and the operating system provides a good infrastructure for storing and retrieving data. In most cases, the most simple and straightforward way to acquire and preserve user data is via SQLite databases.

SQLite

SQLite is a relational database technology that's used most often when the developer requires an embedded database in a small system. SQLite is included with the Android system and can be easily used in your Android app. For more details on SQLite, you can **visit http://www.sqlite.org and http://en.wikipedia.org/wiki/SQLite**

Creating a Database in Android

In order to write data to a database, we need to create our database first. To create a database in SQLite in Android, you need to create a subclass of the class SQLiteOpenHelper. This class provides the functionality to create a database and calls a function that can be overridden by the derived class. The SQLiteOpenHelper constructor accepts four arguments, which are

Context – This is the context that will be used to create the database. (Here you can pass the Activity object.)

Name – The name of the new database file.

Factory – A cursor factory (you can usually pass this as null)

Version – The version of the database. This number is used to identify if there is an upgrade or downgrade of the database.

Below, we I've created a class (MyDatabaseHelper) that is going to derive from SQLiteOpenHelper:

```
public class MyDatabaseHelper extends SQLiteOpenHelper{

private static final String DATABASE_NAME="MyFriendsDatabase";

public MyDatabaseHelper(Context context) {

super(context, DATABASE_NAME, null, 1);

}

@Override

public void onCreate(SQLiteDatabase database) {

database.execSQL("CREATE TABLE friends (_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, phonenumber INTEGER);");

}

@Override
```

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS friends");

onCreate(db);

}

}

In the constructor above, we pass the name of the database that we want to create called as "MyFriendsDatabase." We also override the following methods:

onCreate

This method is called when we create the database. It is passed the SQLiteDatabase reference, which we can use to perform various operations on the database. In this method, we use the function execSQL to execute an SQL query to create a table called "friends," which has three columns. The first column, (_id) is required to generate a unique id for each column. This is necessary in case you are going to use classes like SimpleCursorAdapter (more on this later in the article). The second and third columns (name and phone number) are the data fields that we actually want to store.

onUpgrade

This method is called whenever the database is upgraded. In this method, SQLiteDatabase and the oldVersion number and newVersion number are passed.

In this function, we simply drop the "friends" table if it exists and create a new one, but there could be more complex logic based on the tables you are creating in the database and what you would want to happen on an upgrade.

Adding Values within the Database

Once the database is created, now let's examine how we can add values to the database. We are going to use the insert function on the SQLiteDatabase class. The insert function takes three

arguments: the name of the table in which to insert values, the name of one of the column of the table, and a reference to a ContentValues

In class MyDatabaseHelper below, we will write a function to insert values into the database.

```
public void addFriend(String name,int phonenumber)

{

ContentValues values=new ContentValues(2);

values.put("name", name);

values.put("phonenumber", phonenumber);

getWritableDatabase().insert("friends", "name", values);

}
```

In this function, we take the input arguments as the name and the phonenumber. Then, we create a new object of ContentValues, we put the values for name and phonenumber within that new object. Once we have done that, we get the SQLiteDatabase reference for writing using the function getWritableDatabase, which is a member of SQLiteOpenHelper. Once we have the reference of the SQLiteDatabase, we call the insert function, which takes the arguments as the table name, one of the columns of the table, and the ContentValues. This will enter a new row in your friends database table.

If I want to use MyDatabaseHelper to create a database and add some data values, the code would be as follows:

```
public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

databaseHelper = new MyDatabaseHelper(this);
```

databaseHelper.addFriend("Abbas", 987);

databaseHelper.addFriend("John", 9877);

databaseHelper.addFriend("Michael", 8334);

}

Here in our activity, we create a new object (MyDatabaseHelper) and then add some values in the table by calling the function addFriend.

Running Raw Queries on a Database

Once we have added the data into the database, we will now see how the data can be retrieved. To retrieve the data, there are two possible methods.

Runing a raw SQL query

We can directly run an SQL query like SELECT to retrieve the records from the database.

To run a raw SQL query, you have the function rawQuery, which takes the SQL statement as a string as its first parameter and the selection arguments as its second parameter. So, if we have to write a function in MyDatabaseHelper to retrieve all records from table friends, it would be as follows:

public Cursor getFriends()

*{ Cursor cursor = getReadableDatabase().rawQuery("select * from friends", null);*

return cursor;

}

To Run Queries on Database

We can also run queries on the database using the function query on the SQLiteDatabase. The query function lets you query on one table and also specify some selection and ordering criteria. The query function takes several arguments: the name of the table, the name of the columns of

the table to retrieve, the selection criteria, the arguments for selection criteria, the group by clause, the having clause, the order by clause, and the limit.

If we have to write the same function above to retrieve all friends using query, it would be as follows:

```
public Cursor getFriends()

{

Cursor cursor = getReadableDatabase().query("friends",

new String[] { "_id", "name", "phonenumber"},

null, null, null, null, null);

return cursor;

}
```

The Database Cursor

The result of a query is returned in form of a Cursor object, which basically helps by caching the result of the query in an efficient way and providing functions to access the data.

If we want to fetch the data of all friends from the Cursor returned by the getFriends function, we would write the following code:

```
Cursor AllFriends = databaseHelper.getFriends();

AllFriends.moveToFirst();

while (!AllFriends.isAfterLast()) {

String Name = AllFriends.getString(1);

int number = AllFriends.getInt(2);

AllFriends.moveToNext();
```

}

The cursor has functions such as moveToFirst, which moves the cursor to the first record, moveToNext, which moves the cursor to the next record, and isAfterLast, which checks if the Cursor has moved passed the last record.

The cursor also has functions like getInt, getString, etc., which take the index of the column and return the value.

Deleting Values from the Database

To delete values from the database, you can either run raw query or use the delete function on SQLiteDatabase. The delete function takes three arguments: the table name, the where clause, and the where clause arguments. The function to delete all values from the database is as follows:

public void deleteAll()

{

getWritableDatabase().delete("friends", null, null);

}

Using Cursor Adapter to Display Data Values.

Android provides us with Cursor adapters, which let us attach a cursor with a ListView. The class SimpleCursorAdapter is one of these adapters, and it's quite useful. If we want to show the list of friends in a list view, the code would be as follows:

Cursor AllFriends = databaseHelper.getFriends();

String[] from = { "name", "phonenumber" };

int[] to = { android.R.id.text1, android.R.id.text2 };

ListAdapter adapter=new SimpleCursorAdapter(this,android.R.layout.simple_list_item_2,

AllFriends

, from,

to);

ListView myList=(ListView)findViewById(android.R.id.list);

myList.setAdapter(adapter);

4.5. Working with a Content Provider

In Android, Content Providers are a very important component that serves the purpose of a relational database to store the data of applications. The role of the content provider in the android system is like a central repository in which data of the applications are stored, and it facilitates other applications to securely access and modifies that data based on the user requirements. Android system allows the content provider to store the application data in several ways. Users can manage to store the application data like images, audio, videos, and personal contact information by storing them in **SQLite Database, in files**, **or even on a network**. In order to share the data, content providers have certain permissions that are used to grant or restrict the rights to other applications to interfere with the data.

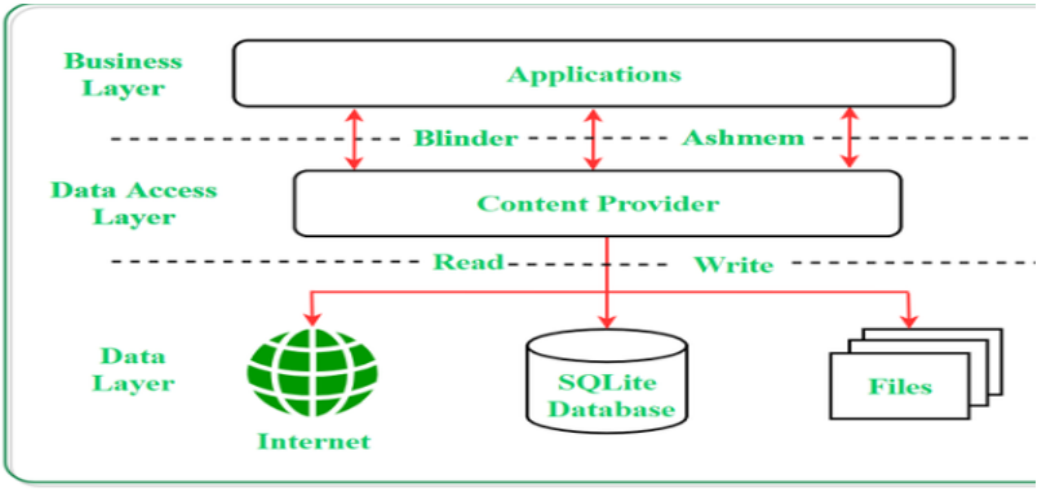


Figure 26: Working with a Content Provider

Content URI

Content URI (Uniform Resource Identifier) is the key concept of Content providers. To access the data from a content provider, URI is used as a query string.

Structure of a Content URI: *content://authority/optionalPath/optionalID*

Details of different parts of Content URI:

- **content:// –** Mandatory part of the URI as it represents that the given URI is a Content URI.
- **authority –** Signifies the name of the content provider like contacts, browser, etc. This part must be unique for every content provider.
- **optionalPath –** Specifies the type of data provided by the content provider. It is essential as this part helps content providers to support different types of data that are not related to each other like audio and video files.
- **optionalID –** It is a numeric value that is used when there is a need to access a particular record.

If an ID is mentioned in a URI then it is an id-based URI otherwise a directory-based URI.

Operations in Content Provider

Four fundamental operations are possible in Content Provider namely **Create**, **Read**, **Update**, and **Delete**. These operations are often termed as **CRUD operations**.

- **Create:** Operation to create data in a content provider.
- **Read:** Used to fetch data from a content provider.
- **Update:** To modify existing data.
- **Delete:** To remove existing data from the storage.

Working of the Content Provider UI components of android applications like Activity and Fragments use an object **CursorLoader** to send query requests to **ContentResolver.** The ContentResolver object sends requests (like create, read, update, and delete) to the **ContentProvider** as a client. After receiving a request, Content Provider process it and returns the desired result. Below is a diagram to represent these processes in pictorial form.

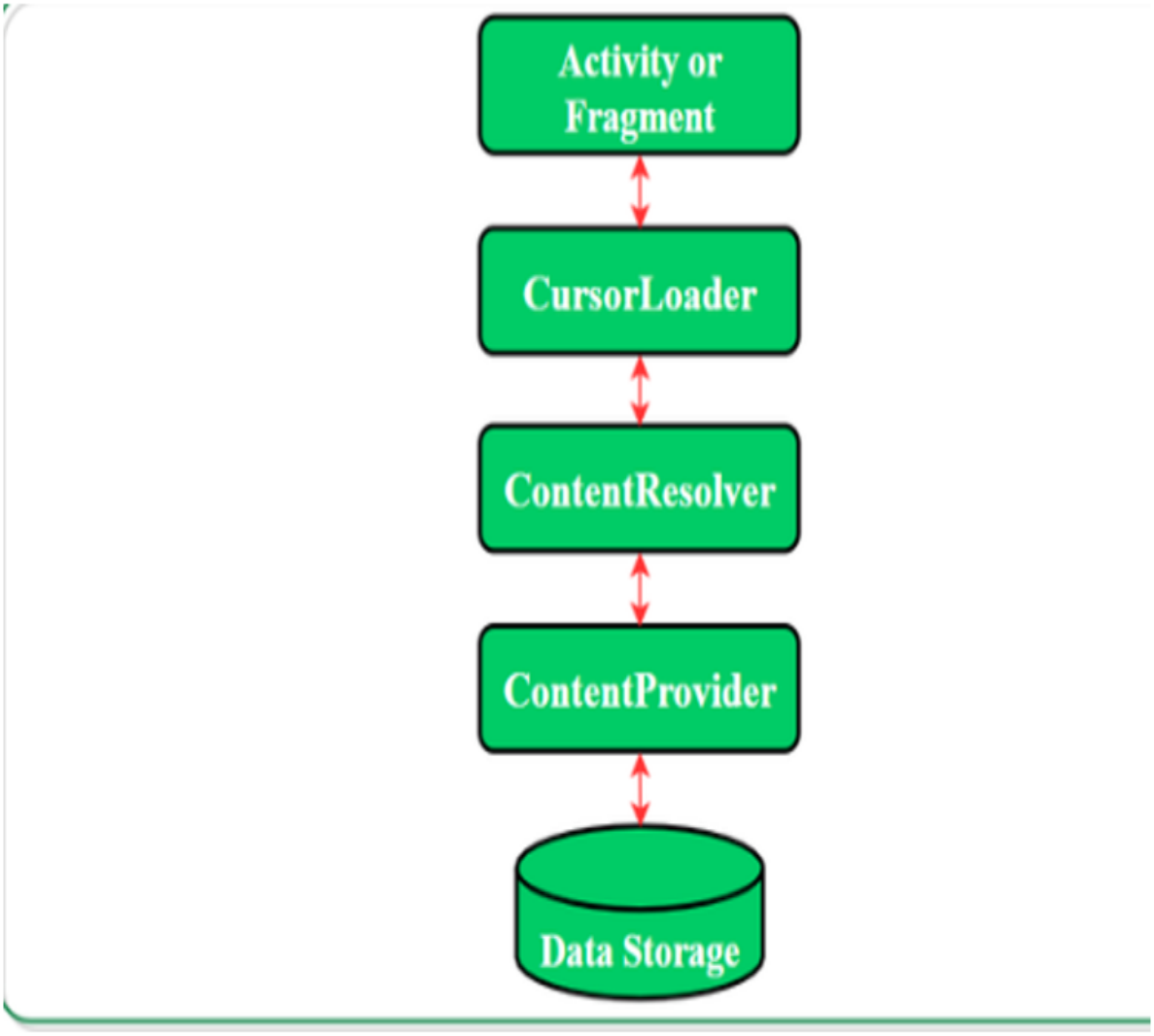


Figure 27:UI components of android applications

Creating a Content Provider

Following are the steps which are essential to follow in order to create a Content Provider:

- Create a class in the same directory where the that **MainActivity** file resides and this class must extend the ContentProvider base class.
- To access the content, define a content provider URI address.
- Create a database to store the application data.
- Implement the **six abstract methods** of ContentProvider class.
- Register the content provider in **AndroidManifest.xml** file using **<provider> tag**.

Following are the six abstract methods and their description which are essential to override as the part of Content Provider class:

Abstract Method	Desription
query()	A method that accepts arguments and fetches the data from the desired table. Data is retired as a cursor object.
insert()	To insert a new row in the database of the content provider. It returns the content URI of the inserted row.
update()	This method is used to update the fields of an existing row. It returns the number of rows updated.
delete()	This method is used to delete the existing rows. It returns the number of rows deleted.
getType()	This method returns the Multipurpose Internet Mail Extension(MIME) type of data to the given Content URI.
onCreate()	As the content provider is created, the android system calls this method immediately to initialise the provider.

Example

The prime purpose of a content provider is to serve as a central repository of data where users can store and can fetch the data. The access of this repository is given to other applications also but in a safe manner in order to serve the different requirements of the user. The following are the steps involved in implementing a content provider. In this content provider, the user can store the name of persons and can fetch the stored data. Moreover, another application can also access the stored data and can display the data.

Note: *Following steps are performed on Android Studio version 4.0*

Creating a Content Provider:

Step 1: Create a new project

1. Click on File, then New => New Project.
2. Select language as Java/Kotlin.
3. Choose empty activity as a template

4. Select the minimum SDK as per your need.

Step 2: Modify the strings.xml file

All the strings used in the activity are stored here.

xml

```
<resources>
    <string name="app_name">Content_Provider_In_Android</string>
    <string name="hintText">Enter User Name</string>
    <string name="heading">Content Provider In Android</string>
    <string name="insertButtontext">Insert Data</string>
    <string name="loadButtonText">Load Data</string>
</resources>
```

Step 3: Creating the Content Provider class

1. Click on File, then New => Other => ContentProvider.
2. Name the ContentProvider
3. Define authority (it can be anything for example **"com.demo.user.provider"**)
4. Select **Exported** and **Enabled** option
5. Choose the language as Java/Kotlin

This class extends the ContentProvider base class and override the six abstract methods. Below is the complete code to define a content provider.

```
package com.example.contentprovidersinandroid;

import android.content.ContentProvider;

import android.content.ContentUris;

import android.content.ContentValues;

import android.content.Context;

import android.content.UriMatcher;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;
```

```
import android.database.sqlite.SQLiteException;

import android.database.sqlite.SQLiteOpenHelper;

import android.database.sqlite.SQLiteQueryBuilder;

import android.net.Uri;

import java.util.HashMap;

 public class MyContentProvider extends ContentProvider {

    public MyContentProvider() {

    }

      // defining authority so that other application can access it

    static final String PROVIDER_NAME = "com.demo.user.provider";

      // defining content URI

    static final String URL = "Error! Hyperlink reference not valid." +
PROVIDER_NAME + "/users";

      // parsing the content URI

    static final Uri CONTENT_URI = Uri.parse(URL);

     static final String id = "id";

    static final String name = "name";

    static final int uriCode = 1;

    static final UriMatcher uriMatcher;

    private static HashMap<String, String> values;

     static {

         // to match the content URI

        // every time user access table under content provider

        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);

         // to access whole table
```

```
        uriMatcher.addURI(PROVIDER_NAME, "users", uriCode);
          // to access a particular row
        // of the table
        uriMatcher.addURI(PROVIDER_NAME, "users/*", uriCode);
    }
    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
            case uriCode:
                return "vnd.android.cursor.dir/users";
            default:
                throw new IllegalArgumentException("Unsupported URI: " +
uri);
        }
    }
      // creating the database
    @Override
    public boolean onCreate() {
        Context context = getContext();
        DatabaseHelper dbHelper = new DatabaseHelper(context);
        db = dbHelper.getWritableDatabase();
        if (db != null) {
            return true;
        }
        return false;
    }
    @Override
```

```
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
        qb.setTables(TABLE_NAME);
        switch (uriMatcher.match(uri)) {
            case uriCode:
                qb.setProjectionMap(values);
                break;
            default:
                throw new IllegalArgumentException("Unknown URI " + uri);
        }
        if (sortOrder == null || sortOrder == "") {
            sortOrder = id;
        }
        Cursor c = qb.query(db, projection, selection,
selectionArgs, null,
                null, sortOrder);
        c.setNotificationUri(getContext().getContentResolver(), uri);
        return c;
    }
      // adding data to the database
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        long rowID = db.insert(TABLE_NAME, "", values);
        if (rowID > 0) {
            Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
```

```
            getContext().getContentResolver().notifyChange(_uri, null);
            return _uri;
        }
        throw new SQLiteException("Failed to add a record into " + uri);
    }
    @Override
    public int update(Uri uri, ContentValues values, String selection,
                      String[] selectionArgs) {
        int count = 0;
        switch (uriMatcher.match(uri)) {
            case uriCode:
                count = db.update(TABLE_NAME, values, selection,
selectionArgs);
                break;
            default:
                throw new IllegalArgumentException("Unknown URI " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        int count = 0;
        switch (uriMatcher.match(uri)) {
            case uriCode:
                count = db.delete(TABLE_NAME, selection, selectionArgs);
```

```
                break;
            default:
                throw new IllegalArgumentException("Unknown URI " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }
    // creating object of database
    // to perform query
    private SQLiteDatabase db;
    // declaring name of the database
    static final String DATABASE_NAME = "UserDB";
    // declaring table name of the database
    static final String TABLE_NAME = "Users";
    // declaring version of the database
    static final int DATABASE_VERSION = 1;
    // sql query to create the table
    static final String CREATE_DB_TABLE = " CREATE TABLE " + TABLE_NAME
            + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
            + " name TEXT NOT NULL);";
    // creating a database
    private static class DatabaseHelper extends SQLiteOpenHelper {
        // defining a constructor
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

```
        }

         // creating a table in the database

        @Override

        public void onCreate(SQLiteDatabase db) {

            db.execSQL(CREATE_DB_TABLE);

        }

        @Override

        public void onUpgrade(SQLiteDatabase
db, int oldVersion, int newVersion) {

            // sql query to drop a table

            // having similar name

            db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);

            onCreate(db);

        }

    }

}
```

Step 4: Design the activity_main.xml layout

One **Textview**, **EditText field, two Buttons**, and a **Textview to display the stored data** will be added in the activity using the below code.

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:background="#168BC34A"
```

```
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        app:layout_constraintBottom_toTopOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.13"
        tools:ignore="MissingConstraints">
        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="40dp"
            android:layout_marginBottom="70dp"
            android:fontFamily="@font/roboto"
            android:text="@string/heading"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="36sp"
            android:textStyle="bold" />
        <EditText
```

```
            android:id="@+id/textName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="40dp"
            android:fontFamily="@font/roboto"
            android:hint="@string/hintText" />
        <Button
            android:id="@+id/insertButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="@font/roboto"
            android:onClick="onClickAddDetails"
            android:text="@string/insertButtontext"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display
1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />
        <Button
            android:id="@+id/loadButton"
            android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:fontFamily="@font/roboto"
            android:onClick="onClickShowDetails"
            android:text="@string/loadButtonText"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display
1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />
        <TextView
            android:id="@+id/res"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="20dp"
            android:layout_marginEnd="20dp"
            android:clickable="false"
            android:ems="10"
            android:fontFamily="@font/roboto"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="18sp"
            android:textStyle="bold" />
    </LinearLayout>
```

```
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/banner" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Chapter Five

5. Communications Via Network and the Web

5.1. Introduction

After completing this Session(chapter), Students able to:

- Explain communication and network
- Understand state machine
- Describe correct communications model
- Explain wireless connectivity and mobile apps
- Discuss notifications and alarms

This section looks at network communication from a very high level and defines some terms used throughout the information. For more detailed information on z/OS® network communication and TCP/IP sockets, see z/OS Communications Server: IP Configuration Guide and z/OS Communications Server: IP Programmer's Guide and Reference. For more detailed information on IPv6 network communication and AF_INET6 sockets, see z/OS Communications Server: IPv6 Network and Appl Design Guide.

Network communication, or *internetworking*, defines a set of protocols (that is, rules and standards) that allow application programs to talk with each other without regard to the hardware and operating systems where they are run. Internetworking allows application programs to communicate independently of their physical network connections.

The internetworking technology called *TCP/IP* is named after its two main protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP). To understand TCP/IP, you should be familiar with the following terms:

client

A process that requests services on the network.

server

A process that responds to a request for service from a client.

The early days of computing, computers were seen as devices for making calculations, storing data, and automating business processes. However, as the devices evolved, it became apparent that many of the functions of telecommunications could be integrated into the computer. During the 1980s, many organizations began combining their once-separate telecommunications and information-systems departments into an information technology, or IT, department. This ability for computers to communicate with one another and, maybe more importantly, to facilitate communication between individuals and groups, has been an important factor in the growth of computing over the past several decades.

Computer networking really began in the 1960s with the birth of the Internet, as we'll see below. However, while the Internet and web were evolving, corporate networking was also taking shape in the form of local area networks and client-server computing. In the 1990s, when the Internet came of age, Internet technologies began to pervade all areas of the organization. Now, with the Internet a global phenomenon, it would be unthinkable to have a computer that did not include communications capabilities. This chapter will review the different technologies that have been put in place to enable this communications revolution.

5.2. Communications Via Network and the Web

Over the next decade, the ARPANET grew and gained popularity. During this time, other networks also came into existence. Different organizations were connected to different networks. This led to a problem: the networks could not talk to each other. Each network used its own proprietary language, or protocol (see sidebar for the definition of *protocol*), to send information back and forth. This problem was solved by the invention of transmission control protocol/Internet protocol (TCP/IP). TCP/IP was designed to allow networks running on different protocols to have an intermediary protocol that would allow them to communicate. So as long as your network supported TCP/IP, you could communicate with all of the other networks running TCP/IP. TCP/IP quickly became the standard protocol and allowed networks to communicate with each other. It is from this breakthrough that we first got the term *Internet*, which simply means "an interconnected network of networks."

working communication is full of some very technical concepts based on some simple principles. Learn the terms below and you'll be able to hold your own in a conversation about the Internet.

- *Packet*: The fundamental unit of data transmitted over the Internet. When a device intends to send a message to another device (for example, your PC sends a request to YouTube to open a video), it breaks the message down into smaller pieces, called packets. Each packet has the sender's address, the destination address, a sequence number, and a piece of the overall message to be sent.
- *Hub*:A simple network device that connects other devices to the network and sends packets to all the devices connected to it.
- *Bridge*: A network device that connects two networks together and only allows packets through that are needed.
- *Switch*: *A* network device that connects multiple devices together and filters packets based on their destination within the connected devices.
- *Router*: A device that receives and analyzes packets and then routes them towards their destination. In some cases, a router will send a packet to another router; in other cases, it will send it directly to its destination.
- *IP Address*: Every device that communicates on the Internet, whether it be a personal computer, a tablet, a smartphone, or anything else, is assigned a unique identifying number called an IP (Internet Protocol) address. Historically, the IP-address standard used has been IPv4 (version 4), which has the format of four numbers between 0 and 255 separated by a period. For example, the domain Saylor.org has the IP address of 107.23.196.166. The IPv4 standard has a limit of 4,294,967,296 possible addresses. As the use of the Internet has proliferated, the number of IP addresses needed has grown to the point where the use of IPv4 addresses will be exhausted. This has led to the new IPv6 standard, which is currently being phased in. The IPv6 standard is formatted as eight groups of four hexadecimal digits, such as 2001:0db8:85a3:0042:1000:8a2e:0370:7334. The IPv6 standard has a limit of 3.4×10^{38} possible addresses. For more detail about the new IPv6 standard,
- *Domain name*: If you had to try to remember the IP address of every web server you wanted to access, the Internet would not be nearly as easy to use. A domain name is a human-friendly name for a device on the Internet. These names generally consist of a descriptive text followed by the top-level domain (TLD). For example, Wikepedia's domain name is wikipedia.org; *wikipedia* describes the organization and *.org* is the top-level domain. In this case, the *.org* TLD is designed for nonprofit organizations. Other well-known TLDs include *.com*, *.net*, and *.gov*. For a complete list and description of domain names,

- *DNS*: DNS stands for "domain name system," which acts as the directory on the Internet. When a request to access a device with a domain name is given, a DNS server is queried. It returns the IP address of the device requested, allowing for proper routing.
- *Packet-switching*: When a packet is sent from one device out over the Internet, it does not follow a straight path to its destination. Instead, it is passed from one router to another across the Internet until it is reaches its destination. In fact, sometimes two packets from the same message will take different routes! Sometimes, packets will arrive at their destination out of order. When this happens, the receiving device restores them to their proper order. For more details on packet-switching.
- *Protocol*:In computer networking, a protocol is the set of rules that allow two (or more) devices to exchange information back and forth across the network.

5.3. Telephony

The word *telephony* comes from the Greek root words for far, *tele*, and speak, *phone*. In 1876, Alexander Graham Bell famously patented the telephone as a way to electronically transmit human speech, building on the success of the telegraph system. A few days later, he spoke the first words ever communicated via telephone to his assistant, Thomas Watson, who was in the next room: "Mr. Watson, come here. I want to see you." The following year, he made the first distanced phone call in history — from Salem, Mass., to Boston — and founded the Bell Telephone Company, which would later become part of AT&T. The earliest telephones came in pairs, with each line directly connecting just two locations. Impractically, users, therefore, needed a separate wired device for each contact they wished to call. Soon, the invention of the telephone exchange enabled users to communicate with any other local party wired for service, with the help of a switchboard operator. Trunk lines connected individual telephone exchanges, enabling calls to take place over increasingly significant distances.

Telephony is technology associated with interactive communication between two or more physically distant parties via the electronic transmission of speech or other data. Long associated with voice communication, telephony has evolved to also include text messaging, video calling, video conferencing, voicemail, call recording and faxing.

A telephonic exchange historically required the use of traditional fixed-line telephones, handheld devices containing both speakers (transmitters) and receivers that connected to local exchange networks via physical wiring. Telephonic communication increasingly happens using

modern computing and cellular technology, thus blurring the line between the fields of telephony and telecommunication. The definition of telephony and its scope have expanded accordingly.

Internet telephony enables users to make calls over Internet Protocol (IP) networks, at much lower cost than over the traditional public switched telephone network (PSTN). Computer applications called *softphones* behave like legacy telephones but without the need for standalone devices. IP telephony software can reside on a variety of computing hosts, including personal computers (PCs), smart desk phones, smart mobile phones and tablets.

How telephony works

- **How traditional telephony works.** Traditional phone systems convert sound waves at one end of a call into electrical signals that travel to their destination via the wires and cables of the PSTN — the circuit-switched telephone network that crisscrosses the globe. The recipient's phone then converts the transmission back into sound signals, enabling a real-time conversation. Also known as the *plain old telephone service* (POTS), the PTSN is operated by international, national, regional and local carriers. Its underlying infrastructure originally featured copper lines but now includes fiber-optic cabling, cellular technology and satellite systems.
- **How mobile telephony works.** Using cellular technology, a mobile phone converts sound into electrical signals that it broadcasts via radio waves to a local cell tower. The cellular network uses radio signaling to forward this information to the recipient's phone, which, in turn, converts the signals back into sound.
- **How internet telephony** Internet telephony software converts sound waves into data that travels over packet-switched computer networks, enabling voice calls to occur online and independent of the PSTN and cellular systems. Voice over IP (VoIP), a Layer 3 protocol and subset of IP telephony, delivers voice and other communication services, such as video conferencing and text messaging, across broadband and private IP networks.
- **How faxing works.** Legacy fax (short for facsimile) technology is similar to traditional voice telephony. Instead of using sound waves as the mechanism for converting data, a fax machine's sensor encodes a printed document, which it interprets as a two-dimensional, fixed bitmap The machine then converts that graphic into electronic signals. Those signals traverse the telephone network and arrive at the receiving fax machine, which reconverts and decodes the signals,

reassembling and printing them to reflect the original document. IP faxing, also known as *fax over IP* (FoIP) or *virtual faxing*, functions much like traditional faxing but via the internet. Software encodes a scanned document and converts it into data that can travel over local or wide area IP networks before being decoded and reconverted on the receiving end.

AT&T began planning a cross-country telephone line in 1908, eventually laying 3,400 miles of copper wiring from coast to coast. Engineers used innovative loading coils and vacuum tubes to amplify electrical signals as they made their way across the country.

Thirty-nine years after filing his patent, Bell inaugurated the transcontinental telephone service with a ceremonial call from New York to San Francisco. He repeated his now-famous words: "Mr. Watson, come here. I want to see you." Watson replied from California: "It will take me five days to get there now." President Woodrow Wilson and AT&T President Theodore Vail joined them on the call. Other major milestones in the history of telephony include the following:

- **1889.** The first automatic telephone switch was invented.
- **1956.** The first submarine transatlantic telephone cable system was launched between Europe and North America.
- **1965.** AT&T deployed the first all-electronic telephone switch, enabling advanced features like call waiting.
- **1969.** The Advanced Research Projects Agency Network (ARPANET), an experimental predecessor of the internet, came to fruition with the successful interconnection of four university computers.
- **1973.** Computer scientist Danny Cohen pioneered Network Voice Protocol (NVP), the forerunner of VoIP, at the University of Southern California (USC).
- **1983.** Motorola debuted the first commercially available mobile phone. Nicknamed "The Brick," the DynaTAC weighed two pounds, took 10 hours to charge and sold for $3,995 each.
- **1994.** Smartphones became available to the public with the launch of IBM's Simon Personal Communicator.
- **1995.** VocalTec released Internet Phone, considered the first commercially available VoIP service.
- **1999.** Internet Engineering Task Force (IETF) standardizes Session Initiation Protocol (SIP), a key network protocol underlying many online telecommunication services.

- **2003.** Skype launched its revolutionary free VoIP service, later adding video functionality.
- **2011.** Google debuted the open-source Web Real-Time Communication (WebRTC) project, which enables voice and video communication across standard web browsers without the use of plugins.

4 types of telephony systems for the enterprise

1. **Legacy, circuit-switched PBX.** In traditional phone systems, on-premises private branch exchange (PBX) boxes act as in-house switching centers, connecting calls between internal devices directly and routing external calls to and from the PSTN via dedicated trunk lines. Human operators managed the earliest PBX systems but were eventually replaced by modern automated attendants. While expensive and complex — and more or less obsolete — legacy PBX systems have historically been most attractive in areas with unreliable network connectivity or electrical power. (Some hard-wired phone lines continue to work even in the event of electrical outages.)
2. **VoIP PBX.** Similar to the traditional PBX, a VoIP or IP PBX essentially creates a private phone network within a business but does so via the internet or a private local area network (LAN) rather than dedicated voice circuits. Because it doesn't require a separately wired network and doesn't rely on the PSTN — with its long-distance charges — internet telephony is significantly less expensive than legacy PBX systems. It also offers more geographic flexibility to users, as they can access softphone applications across their devices, rather than staying tethered to desk handsets. On-premises IP PBX telephony systems consist of software running on servers. An on-site IP PBX typically offers more sophisticated feature sets and greater reliability than a cloud-based VoIP service (see below), but it is also more expensive and complex to deploy, manage and maintain.
3. **Hybrid PBX.** Some organizations deploy a hybrid PBX model that uses both legacy and IP telephony technology — whether for redundancy, cost savings or to enable a staged migration to VoIP. These environments use gateways to bridge VoIP and PSTN functionality. Alternatively, some use the term *hybrid PBX* to refer to a VoIP telephony system with both on-premises and cloud-hosted resources.
4. **Cloud VoIP.** Cloud VoIP, also known as *cloud telephony* or *cloud calling*, is a VoIP service provided by a third party. By outsourcing its system, an enterprise can eliminate the need to provision, manage and maintain an on-site IP PBX, reducing infrastructure costs and increasing

scalability. Features, functionality and controls tend to be less comprehensive than on-premises options, although many analysts expect that difference to shrink with time.

5.4. Notifications and Alarms

Notifications are a handy way to add features and functionality to your application without your users directly interacting with them. For example, they can be used to add playback controls to a music app, provide quick actions to respond to sms or email and alert about breaking news stories among many others.

You will work with the Notification APIs to issue ***Notifications*** with the ***NotificationManager***, schedule ***Alarms*** with the ***AlarmManager*** and enhance them by using ***groups***, ***channels*** and ***actions***.

Getting Started

Start by downloading the project materials by using the ***Download Materials*** button found at the top or bottom of this tutorial. Unzip the contents to a folder and remember the location. Open Android Studio and, at the splash page, choose ***Open an existing Android Studio Project***, navigate to the recently downloaded folder and select ***PetMedicineReminder-Starter***.

Once the starter project finishes loading and building, run the application on a device or emulator.

Once the app is running, load the sample data by selecting the overflow icon in the top right and tapping ***Load Sample Data***. When the data loads, you should see a listing of reminders to administer medications to a few pets.

n order to display a notification, you will have to do some setup work. You will need to:

1. Create a notification channel.
2. Register the notification channel.
3. Create a notification.
4. Send a notification using ***NotificationManager***.

Create a Notification Channel

Notification channels provide a common visual and auditory experience for notifications of a similar type. Since their introduction in API 26, you are now required to set a channel for a notification, otherwise they will not display on newer versions of Android.

Open the ***NotificationHelper.kt*** file under the ***notif*** directory and add the following code to the `createNotificationChannel()` method:

```
val channelId = "${context.packageName}-${context.getString(R.string.app_name)}"

val notificationBuilder = NotificationCompat.Builder(context, channelId).apply {

 setSmallIcon(R.drawable.ic_stat_medicine) // 3

 setContentTitle(title) // 4

 setContentText(message) // 5

 setStyle(NotificationCompat.BigTextStyle().bigText(bigText)) // 6

 priority = NotificationCompat.PRIORITY_DEFAULT // 7

 setAutoCancel(autoCancel) // 8

}}
```

Here you:

1. Safety checked the OS version for API 26 and greater.
2. Created a unique name for the notification channel. The name and description are displayed in the application's Notification settings.
3. Created the channel using the NotificationManager.

Nice work! Now you need to call this method.

Register the Notification Channel

Open the ***PetRx.kt*** application file under the root package and add the following code to the `onCreate()` method:

```
NotificationHelper.createNotificationChannel(this,
    NotificationManagerCompat.IMPORTANCE_DEFAULT, false,
    getString(R.string.app_name), "App notification channel.")
```

Now that the channel is created, you can send a notification to it when the sample data is loaded.

Create a Notification

Open the ***NotificationHelper.kt*** file and navigate to `createSampleDataNotification()`. Add the following code:

```
val channelId = "${context.packageName}-${context.getString(R.string.app_name)}"
```

```
val notificationBuilder = NotificationCompat.Builder(context, channelId).apply {

 setSmallIcon(R.drawable.ic_stat_medicine) // 3

 setContentTitle(title) // 4

 setContentText(message) // 5

 setStyle(NotificationCompat.BigTextStyle().bigText(bigText)) // 6

 priority = NotificationCompat.PRIORITY_DEFAULT // 7

 setAutoCancel(autoCancel) // 8

}
```

1. Create the unique channelId for this app using the package name and app name.
2. Use NotificationCompat.Builder to begin building the notification.
3. Set a small icon to be display in the notification shade. This is the only required attribute.
4. Set a title for the notification.
5. Set content for the notification.
6. Set the style of the notification style to NotificationCompat.BigTextStyle().
7. Set the notifications priority to the default priority. Priority indicates how much of the user's attention the notification should draw. You will see other usages later but acceptable values include:

- PRIORITY_MIN

- PRIORTY_MAX
- PRIORITY_LOW
- PRIORTY_HIGH
- PRIORITY_DEFAULT

8. Set the notification to auto cancel when tapped.

Pending Intents

So far you have created the notification channel and began building a notification but you want to direct the user somewhere when they tap on the notification. This requires adding a `PendingIntent` to the notification calling `setContentIntent` when building the notification.
A ***PendingIntent*** is roughly described as an action to be taken at a later point in time.

Add the following lines to the `createSampleDataNotification()` method inside the `apply` lambda just after `setAutoCancel(autoCancel)`:

```
val intent = Intent(context, MainActivity::class.java)

intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK // 2

val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0) // 3

setContentIntent(pendingIntent)
```

Here you:

1. Created an `Intent` to launch the `MainActivity`.
2. Wrapped the `Intent` in a `PendingIntent`, created through the `getActivity()` method which returns a description of an `Activity` to be launched.

3. Called setContentIntent() to attach it to the NotificationCompat.Builder.

Notifying the Manager

The last piece to issuing a notification is getting a reference to the ***NotificationManagerCompat*** system service and calling notify().

Add the following code after the apply lambda:

val notificationManager = NotificationManagerCompat.from(context)// 2

notificationManager.notify(1001, notificationBuilder.build())

))

Here you:

1. Used the app's Context to get a reference to NotificationManagerCompat.
2. Called notify() on the NotificationManager passing in an identifier and the notification.

Re-run the application, delete the data from the menu ***Delete Data*** and reload it from the menu ***Load sample data*** item. If everything was successful, you should now see a notification icon in the status bar! Pull down the notification shade and you will see the full notification issued from your application indicating your sample data has loaded. Great!

Alarm Time

The sample app contains a model named ReminderData. Each ReminderData object contains both a *time* and a *list of days*. When the user creates the reminder, this information is input and saved to a local database. You'll make use of the ReminderData when working with the AlarmManager to create an alarm. More on this in a bit.

Creating the PendingIntent

Open the AlarmScheduler.kt file under the ***notif*** package and navigate to the createPendingIntent(): PendingIntent? method.

Add the following code:

```
val intent = Intent(context.applicationContext, AlarmReceiver::class.java).apply {// 2

 action = context.getString(R.string.action_notify_administer_medication)// 3

  type = "$day–${reminderData.name}–${reminderData.medicine}–
${reminderData.type.name}"

  putExtra(ReminderDialog.KEY_ID, reminderData.id)

}
```

Return PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)

Here you:

1. Create the Intent with a destination of AlarmReceiver::class.java (this class should already exist, but you'll fill it in in the next section).
2. Set the action for the Intent.
3. Set the type – This has to be unique so you construct it using the day, the pet's name, the medicine and the pet type. If this is not unique it will overwrite any other PendingIntent with this same type. If you are interested you can read more about how two Intent's are considered equal.
4. Add the reminder's ID in the Intent's bundle so you can use it in the AlarmReceiver.
5. Create the PendingIntent using the getBroadcast() method. This is very important because you are creating the Intent with a BroadcastReceiver as a target.

Scheduling the Alarm

Now that the notification channels are set up and you have created a unique PendingIntent for each reminder, it's time to actually schedule the alarms.

Open the ***AlarmScheduler.kt*** file and find the `scheduleAlarm()` method. Add the following code:

```
val datetimeToAlarm = Calendar.getInstance(Locale.getDefault())

datetimeToAlarm.timeInMillis = System.currentTimeMillis()

datetimeToAlarm.set(HOUR_OF_DAY, reminderData.hour)

datetimeToAlarm.set(MINUTE, reminderData.minute)

datetimeToAlarm.set(SECOND, 0)

datetimeToAlarm.set(MILLISECOND, 0)

datetimeToAlarm.set(DAY_OF_WEEK, dayOfWeek)

val today = Calendar.getInstance(Locale.getDefault())

if (shouldNotifyToday(dayOfWeek, today, datetimeToAlarm)) {

  alarmMgr.setRepeating(AlarmManager.RTC_WAKEUP,

      datetimeToAlarm.timeInMillis, (1000 * 60 * 60 * 24 * 7).toLong(), alarmIntent)
```

```
 return

}

datetimeToAlarm.roll(WEEK_OF_YEAR, 1)

alarmMgr.setRepeating(AlarmManager.RTC_WAKEUP,

   datetimeToAlarm.timeInMillis, (1000 * 60 * 60 * 24 * 7).toLong(), alarmIntent)
```

You just did three important things:

1. Set up a Calendar for the alarm's time using the ReminderData.
2. Checked whether that alarm should be scheduled today and scheduled it if so.
3. Else, schedule the alarm to repeat every week at that time.

Create a Broadcast Receiver

There are just a few more things to consider when dealing with alarms, but how do they tie back into the notifications? Well, the instances of PendingIntent you registered through the AlarmManager were created using the PendingIntent.getBroadcast() method. This method allows you to send a broadcast to the system that notifies any listeners of the action you set for the Intent. Open a file named ***AlarmReceiver*** in the ***notif*** package. Make sure it inherits from the BroadcastReceiver() base class and implement the onReceive() method.

```
if (context != null && intent != null && intent.action != null) {
```

```
 if(intent.action!!.equals(context.getString(R.string.action_notify_administer_medication),
ignoreCase = true)) {


   if (intent.extras != null) {


     valreminderData =
DataUtils.getReminderById(intent.extras!!.getInt(ReminderDialog.KEY_ID))


     if (reminderData != null) {


       NotificationHelper.createNotificationForPet(context, reminderData)


     }


   }


 }


}
```

With this code, you:

1. Check that the Intent's ***action*** matches the one from the Intent you created above.
2. Looked up the ReminderData in the database using the extra from the Intent's Bundle.
3. Create the notification using the ReminderData.

Open AndroidManifest.xml and register the ***AlarmReceiver*** by adding this xml inside the ***application*** tag:

```
<receiver android:name=".notif.AlarmReceiver" />
```

All right, you have finally made it to the last step! Now you will issue the notification for the pet's medication.

Create the Notification and Issue It

Open the NotificationHelper class one more time, navigate to the createNotificationForPet()method and add the following code:

```
val groupBuilder = buildGroupNotification(context, reminderData)

val notificationBuilder = buildNotificationForPet(context, reminderData)

val administerPendingIntent = createPendingIntentForAction(context, reminderData)

notificationBuilder.addAction(

 R.drawable.baseline_done_black_24,

 context.getString(R.string.administer),

 administerPendingIntent)
```

```
val notificationManager = NotificationManagerCompat.from(context)

notificationManager.notify(reminderData.type.ordinal, groupBuilder.build())

notificationManager.notify(reminderData.id, notificationBuilder.build())
```

Here you:

1. Create a group notification.
2. Create a notification for the pet.
3. Add an action to the notification for the pet.
4. Called notify using `NotificationManager` for both notifications.

Create a Group Notification

Group notifications, along with channels, help you stay *organized* as more notifications are introduced into your applications. They will help the notifications issued from your app *group together in the notification shade*. Since the sample app supports reminders for cats, dogs and other pets, you already have the groups you need to get started.

Modify the `buildGroupNotification(context: Context, reminderData: ReminderData)`method like below:

private fun buildGroupNotification(context: Context, reminderData: ReminderData): NotificationCompat.Builder {

```
 val channelId = "${context.packageName}-${reminderData.type.name}"

 return NotificationCompat.Builder(context, channelId).apply {

  setSmallIcon(R.drawable.ic_stat_medicine)
```

```
    setContentTitle(reminderData.type.name)

    setContentText(context.getString(R.string.group_notification_for,
reminderData.type.name))

    setStyle(NotificationCompat.BigTextStyle()

        .bigText(context.getString(R.string.group_notification_for, reminderData.type.name)))

    setAutoCancel(true)

    setGroupSummary(true) // 2

    setGroup(reminderData.type.name) // 3

  }

}
```

1. Using the same channelId, create the group notification.
2. Set this notification as the group summary (useful on versions of Android prior to ***API 24***).
3. Set the group to the pet type name (i.e. Cat, Dog, or Other).

Create a Notification

Next, you need to create the notification using the same `channelId`. This time, you will also add a large icon to the notification to make it distinguishable from the other pet types.

Navigate to the `buildNotificationForPet(context: Context, reminderData: ReminderData)` method and modify it like below:

```
private fun buildNotificationForPet(context: Context, reminderData: ReminderData):

NotificationCompat.Builder {// 1

 val channelId = "${context.packageName}-${reminderData.type.name}"

 return NotificationCompat.Builder(context, channelId).apply {

  setSmallIcon(R.drawable.ic_stat_medicine)

  setContentTitle(reminderData.name)

  setAutoCancel(true)// 2

  val drawable = when (reminderData.type) {

   ReminderData.PetType.Dog -> R.drawable.dog

   ReminderData.PetType.Cat -> R.drawable.cat
```

```
  else -> R.drawable.other

}// 3

setLargeIcon(BitmapFactory.decodeResource(context.resources, drawable))

setContentText("${reminderData.medicine}, ${reminderData.desc}")// 4

setGroup(reminderData.type.name)

if (reminderData.note != null) {

  setStyle(NotificationCompat.BigTextStyle().bigText(reminderData.note))

}

val intent = Intent(context, MainActivity::class.java).apply {

  flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK

  putExtra(ReminderDialog.KEY_ID, reminderData.id)

}
```

```
    val pendingIntent = PendingIntent.getActivity(context, 0, intent, 0)

    setContentIntent(pendingIntent)

  }

}
```

1. Build the Notification using the same channelId as the group notification.
2. Using the ReminderData get a drawable reference for the large icon.
3. Apply the large icon to the notification.
4. Set the notification's group.

5.5. State Machine

A state machine is a concept used in designing computer programs or digital logic. There are two types of state machines: finite and infinite state machines. The former is comprised of a finite number of states, transitions, and actions that can be modeled with flow graphs, where the path of logic can be detected when conditions are met. The latter is not practically used.

A state machine is any device storing the status of something at a given time. The status changes based on inputs, providing the resulting output for the implemented changes. A finite state machine has finite internal memory. Input symbols are read in a sequence producing an output feature in the form of a user interface.

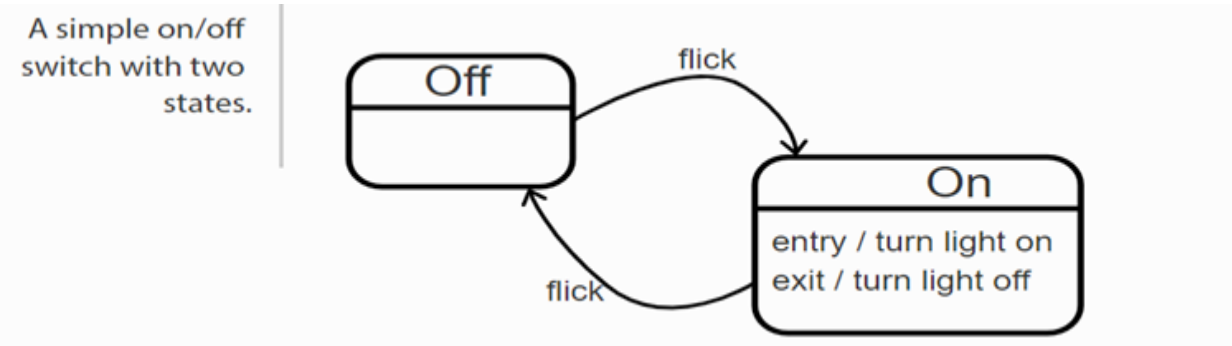
State machines are represented using state diagrams. The output of a state machine is a function of the input and the current state. State machines play a significant role in areas such as electrical engineering, linguistics, computer science, philosophy, biology, mathematics, and logic. They are best used in the modeling of application behavior, software engineering, design of hardware digital systems, network protocols, compilers, and the study of computation and languages.

The operation of a state machine begins from a start state. On a successful transition it ends up in an accept state. The transition takes place based on the inputs provided. The current state depends on the past state of the system. The number of states formed depends on the available states of memory. A transition is enabled based on certain conditions and indicates a state change. An action describes an activity performed at the given moment. The different types of actions are transition action, input action, entry action, and exit action.

Deterministic automata have exactly one transition in every state for each possible input. In non-deterministic automata, a state input leads to one, many, or no transitions. A state machine with only one state is called a combinatorial state machine and uses only input actions.

The two different groups of state machines are acceptors and transducers. Acceptors produce a binary output, based on whether the input is accepted or rejected by the machine. While processing the input, if the current state is accepting, the input is accepted. Otherwise, it is rejected. Languages accepted by state machines are called regular languages. Start states are represented by an arrow pointing on it from anywhere, while accepted states are represented using double circles. Transducers cater output based on a given input, using actions. Moore and Mealy machines are examples of transducers. Unmodified modeling language state machines are also widely used as they have both the Moore and Mealy machine characteristics within them. They include additional concepts such as orthogonal regions and hierarchically-nested states.

Here is a visual depiction of a simple state machine. It is a model of a simple on/off switch.



Some observations over this machine.

- It consists of two states, “on” and “off”. This machine can therefore be in exactly one of the two states at any point in time. In other words, the transitions between states are instantaneous.

- The “flick” event causes it to transition between states.
- When the machine *enters* the “on” state, a side effect occurs. A light is turned on.
- When the machine *exits* the “on” state, another side effect occurs. A light is turned off.

5.6. Correct Communications Model

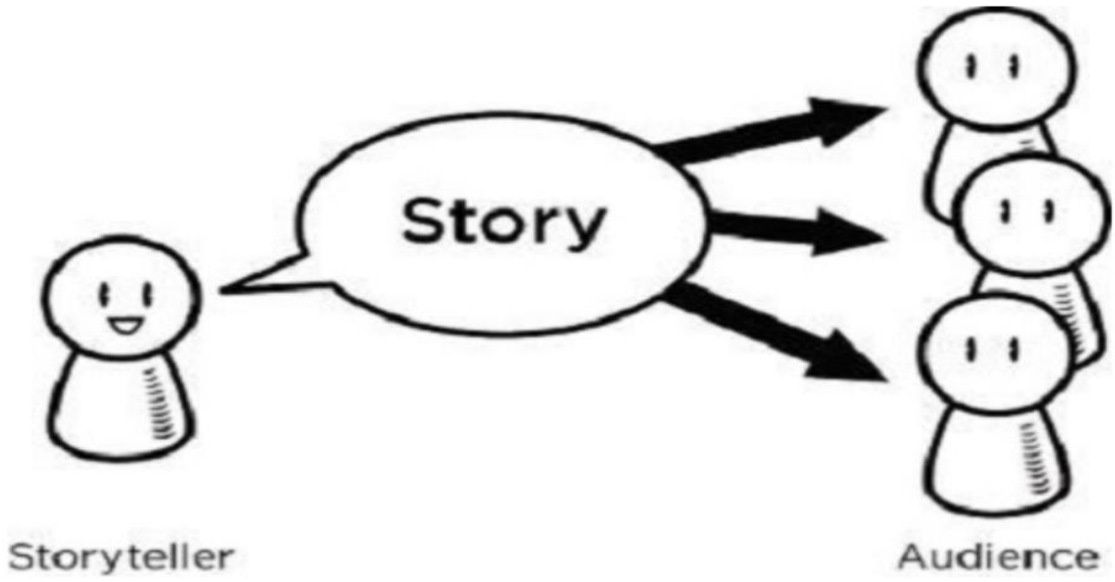
A model is a thing used to follow or imitate. Models of communication provide inputs for setting a standard communication system and ensuring effective communication in a business organization. They are examined from organizational perspectives. Each organization sets up its own communication system mostly deriving inspiration from popular communication models.

- Models of communication are conceptual models that are used to explain the human communication process.
- Models are visual representation of communicating process.
- The first formal communication model was enunciated by Claude Elwood Shannon in 1948 and was published with an introduction by Warren Weaver. This model is popularly known as the Shannon and Weaver Model of Communication.

In the following sections, let us discuss the basic models of communication for easy understanding of theirs which will help us to prepare a model of effective communication in businesses and other areas.

One-Way Communication Model

It is also known as the linear model of Communication as it occurs in a straight line from sender to receiver and serves to inform, persuade or command.



In this type of communication, information is only transmitted from one point to another or to many points at a single point of time.

There is no transmission of information from the receiving point. Here, the recipient is passive.

One-way communication is mostly used to transmit current information like weather information, information about looming calamites, programs and public observations at press conferences and through radio and TV broadcasting. The accurate time and intensity of weather phenomena are transmitted or communicated through the medium of one way communication.

Like in two-way communication, in one-way communication the sender and recipient of the information may be not only people but also computers or automatic devices.

It consists of the sender encoding a message and transmitting it though appropriate channel to the receiver in the presence of noise.

It assumes that there is a clear-cut beginning and end to communication.

The **Aristotle Model of Communication** can be said to be a one-way model of communication in which the sender sends the information or a message to the receiver to influence them and make them respond accordingly. This model is considered to be the golden rule to excel in public speaking, seminars, lectures where the sender makes his point clear by designing an impressive content, passing on the message to the second part and they simply respond accordingly.

Aristotle was the first to take an initiative and design the communication model.

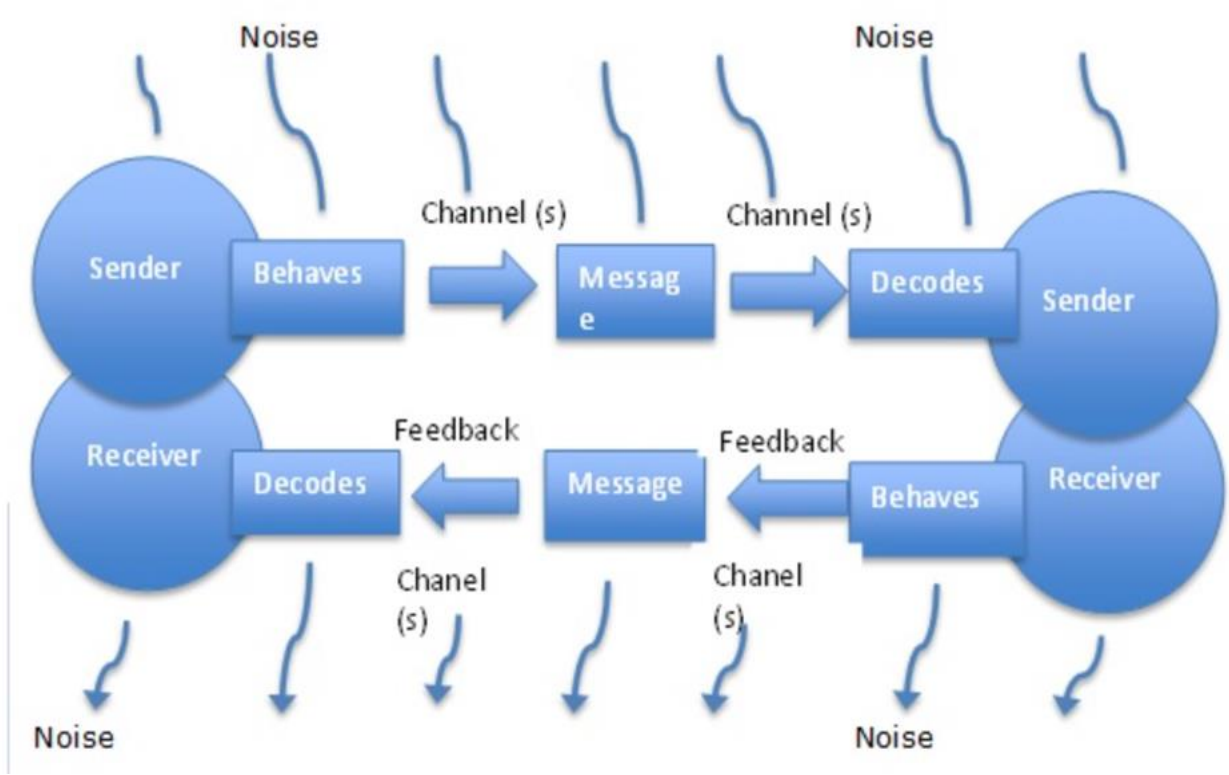
The lack of feedback from the receiver is one major drawback of this model.

In this type of model, the sender is the active member and the receiver is the passive one.

Transactional Model

'Transactional' simply means the exchange or interaction between people. It is based on the basic premises of 'exchange' or 'give and take'.

Hence, the transactional model of communication refers to the exchange of messages or information between the sender and the receiver where each take turns to send or receive messages.



This model is the most general model of communication as even our daily interactions are instances of transactional model communication.

Transactional model becomes more efficient and effective when the participants are of similar environment, know each other and share same social system. Each person in the process reacts

depending upon factors such as their background, experience, attitudes, cultural beliefs and self-esteem.

Proper choice of channel enhances the efficiency and reliability of the communicated message.

Interactive Communication Model

Interactive communication model is also known as the convergence model.

According to this model, exchange of ideas and messages takes place both ways form sender to receiver and vice versa.

The communication process occurs between humans or machines in both verbal and non-verbal ways.

The sender channels a message to the receiver and the receiver then becomes the sender and channels a message to the original sender.

It emphasizes feedback indicating that communication is not a one-way but a two-way process.

Instead of information being sent one way, from the sender to the receiver, both participants send information back and forth.

Wilbur Schramm, a scholar and an authority on communication, also proposed that both the sender and receiver interpret the message, rather than accurately assessing the message's meaning.

In this model, however, feedback is not simultaneous and usually takes time as this model is mostly used for Internet.

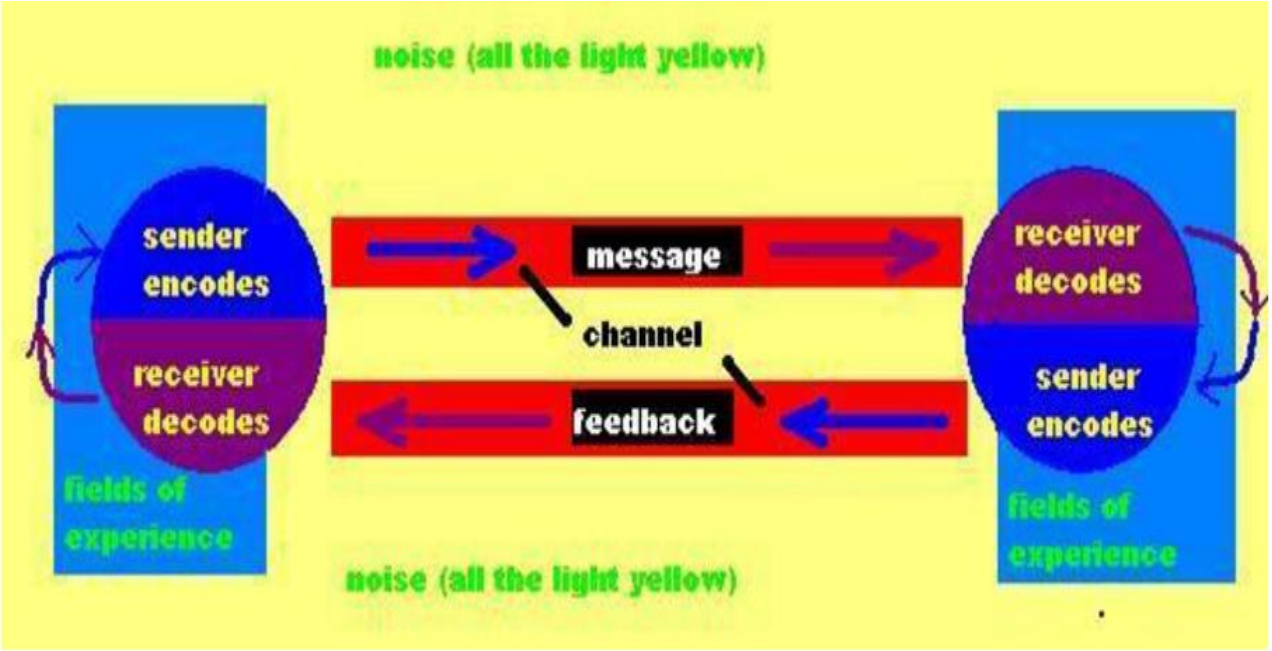


Figure 28:Interactive Communication Model

Components of Interactive Model

In this section, we will consider the components related to the interactive model of communication. This is one of the major models of communication −

- The sender and receiver act as both encoder and decoder and the sources of the message.
- Message is the information to be communicated.
- Feedback, that is, when the decoder forms a second message after receiving the first message.
- Field experience is the experience and knowledge which affect the message formation and interpretation. Such knowledge and experience are often shaped by cultural, psychological, social and situational setting in which the message is formed and is interpreted.
- Interactive model also espouses the concept of noise and barriers to communication like language, network problems, improper channel selection which affects the communication model.
- Internet, social media, interactive marketing. ATMs, online shopping, chat room are examples of interactive communication model.

Effectiveness of Communication Models

Communication models are authentic outcomes of systematic study of various aspects of communication. A model of communication gives a wholesome understanding of a system or structure by which people can understand similar systems or structure.

Consider the following important points related to the importance of communication models −

- Communication models help identify and understand the components and relationship of the communication process being studied.
- Models represent new ideas and thought on various aspects of communication which helps us to plan for effective communication system. They act as stimulant for the researchers and students of communication.
- Models are pictorial representation of communication process.
- They are used for research and investigation in the field of communication.
- As communicating is a rigid process to understand, communication model can be an essential source to understand such rigidity.
- Models predict for more effective communication in future. They also predict the success or failure of a particular communication process.
- Models help explain a phenomenon by providing information in a simplified way which otherwise be complicated or ambiguous.
- We can use the models of communication to analyze real problems and prevent the occurrence of future problems.

5.7. Android Networking and Web

Android lets your application connect to the internet or any other local network and allows you to perform network operations.

A device can have various types of network connections. This chapter focuses on using either a Wi-Fi or a mobile network connection.

Checking Network Connection

Before you perform any network operations, you must first check that are you connected to that network or internet e.t.c. For this android provides **ConnectivityManager** class. You need to instantiate an object of this class by calling **getSystemService()** method. Its syntax is given below −

```
ConnectivityManager check = (ConnectivityManager)

this.context.getSystemService(Context.CONNECTIVITY_SERVICE);
```

Once you instantiate the object of ConnectivityManager class, you can use **getAllNetworkInfo** method to get the information of all the networks. This method returns an array of **NetworkInfo**. So you have to receive it like this.

```
NetworkInfo[] info = check.getAllNetworkInfo();
```

The last thing you need to do is to check **Connected State** of the network. Its syntax is given below −

```
for (int i = 0; i<info.length; i++){

if (info[i].getState() == NetworkInfo.State.CONNECTED){

    Toast.makeText(context, "Internet is connected

    Toast.LENGTH_SHORT).show();

  }

}
```

Apart from these connected states, there are other states a network can achieve.

Sr.No	State
1	Connecting
2	Disconnected
3	Disconnecting
4	Suspended
5	Unknown

content of **AndroidManifest.xml**

```
<?xml version=”1.0″ encoding=”utf-8″?>

<manifest xmlns:android=”http://schemas.android.com/apk/res/android”

  package=”com.tutorialspoint.myapplication” >

  <uses-permission android:name=”android.permission.INTERNET”></uses-permission>

  <uses-permission
android:name=”android.permission.ACCESS_NETWORK_STATE”></uses-permission>
```

5.8. Deciding Scope of an App

The scope of an app refers to how long it will take the app development team to design, build and deliver an application. It’s effectively a timeline that is set by developers with input from the client or sales team and project managers.

There are five key steps that are followed during the scope defining process in app development. First: define the goals of the app

- Second: identify potential issues or hurdles to overcome
- Third: decide which resources are needed to achieve the set goals
- Fourth: set up a milestone schedule to stay on track
- Fifth: list all the important stakeholders that have say in the project

Writing a project scope statement that includes information on the project deliverables is a first step in project planning. The benefits a project scope statement provides to any organization undertaking a new initiative include the following:

- articulates what the project entails so that all stakeholders can understand what's involved;
- provides a roadmap that managers can use to assign tasks, schedule work and budget appropriately;
- helps focus team members on common objectives; and
- prevents projects, particularly complex ones, from expanding beyond the established vision.

Establishing project scope ensures that projects are focused and executed to expectations. The scope provides a strong foundation for managing a project as it moves forward and helps ensure that resources aren't diverted or wasted on out-of-scope elements.

5.9. Wireless Connectivity and Mobile Apps

The definition of mobility and how it differs from wireless connectivity. By definition wireless connectivity between a mobile device and another device requires a physical layer networking technology. To cover all of the subjects that will allow one to comprehensively understand wireless connectivity and its currently existing permutations is impossible, not just in this text, but even in a dedicated text. One may require an entire library to gather all of the various topics on wireless connectivity and related inventions. So, the first thing that we might do is to reduce the scope of what wireless connectivity may mean in our discussion of mobile computing.

we will first look at a basic introduction to wireless communication that will include examples of techniques and technologies from the various abstraction layers in wireless communications.

We can qualify two types of connectivity to the network: *strong* and *weak*. We use the concept of strong and weak connections in various places in this text. Essentially, a weak connection refers to lower QOS than strong connectivity but higher than disconnection. This way of distinguishing weak and strong connections is fairly subjective and relative to the

application. Unfortunately, because of the changing landscape of communication and computing, we cannot quantify the QOS properties of strong and weak connectivity.

Besides enabling communication with the cloud, Android's wireless APIs also enable communication with other devices on the same local network, and even devices which are not on a network, but are physically nearby. The addition of Network Service Discovery (NSD) takes this further by allowing an application to seek out a nearby device running services with which it can communicate. Integrating this functionality into your application helps you provide a wide range of features, such as playing games with users in the same room, pulling images from a networked NSD-enabled webcam, or remotely logging into other machines on the same network.

This class describes the key APIs for finding and connecting to other devices from your application. Specifically, it describes the NSD API for discovering available services and the Wi-Fi Peer-to-Peer (P2P) API for doing peer-to-peer wireless connections. This class also shows you how to use NSD and Wi-Fi P2P in combination to detect the services offered by a device and connect to the device when neither device is connected to a network.

If you're looking for a higher-level API for your Android application to transfer data reliably and securely between devices using a combination of Wi-Fi and Bluetooth, consider using the Nearby Connections API

5.1.Android Telephony

A telephony application in Android is responsible for voice communication through the telephone network. It also provides information about the device and its details, including the IMEI and network type. In addition, Android telephony provides information about SMS. This article will focus on a few of the most important features of an Android telephony application. Let's look at each of them in more detail.

The telephony app is an integral part of the Android system, and it's used by many developers. Android developers often want to use their device as a telephone. Using built-in intents, they can dial outbound calls, intercept, and modify calls. Android also has several other features to

help developers build mobile applications. If you've got an Android device, you're already halfway there.

Telephony is the technology that lets us communicate with others over long distances. Advances in wireless networking and affordable computing power have radically changed this process. Telephony used to mean speaking on an analog telephone, but with the advent of digital phones, it expanded to include voicemail and video conferencing, as well as video conferencing with multiple people.

The Android telephony API provides access to important information about a phone, including IMEI, network state, and SMS. It allows developers to dial outbound calls and modify existing ones. Unlike the iPhone, Android's telephony API does not require calling a specific phone number. Instead, the developer can invoke the built-in dialer and populate it with the number provided. The application then makes a call.

To implement a listener for a call, an application must implement the Phone State Listener interface. This interface provides a handle to telephony-related information on a phone, and allows an application to register a Phone State Listener to receive notifications when a telephony-related state changes. The methods in this class provide access to a phone's data, including a caller's ID and the status of the call.

The Telephony Manager class provides information on the phone's state, including its serial number and network provider code. It also provides methods for application developers to attach event listeners to their applications. The Phone State Listener is one such event listener. This can be used to notify applications when phone service is available, or when a call is started or already in progress. There are several other methods that can be called from within the Telephony Manager class.

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

You can use **ACTION_DIAL** action instead of ACTION_CALL, in that case you will have option to modify hardcoded phone number before making a call instead of making a direct call.

Intent Object – Data/Type to make Phone Call

To make a phone call at a given number 91-000-000-0000, you need to specify **tel:** as URI using setData() method as follows −

```
phoneIntent.setData(Uri.parse("tel:91-000-000-0000"));
```

The interesting point is that, to make a phone call, you do not need to specify any extra data or data type.

Example

Following example shows you in practical how to use Android Intent to make phone call to the given mobile number.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

he content of the modified main activity file **src/MainActivity.java**.

```
package com.example.saira_000.myapplication;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.support.v4.app.ActivityCompat;

import android.support.v7.app.AppCompatActivity;
```

```
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
  private Button button;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    button = (Button) findViewById(R.id.buttonCall);
    button.setOnClickListener(new View.OnClickListener() {
      public void onClick(View arg0) {
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:0377778888"));
        if (ActivityCompat.checkSelfPermission(MainActivity.this,
          Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
            return;
          }
        startActivity(callIntent);
      }
```

```
    });
  }
}
```

Following will be the content of ***res/layout/activity_main.xml*** *file −*

```
<?xml version=”1.0" encoding=”utf-8"?>
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
  android:layout_width=”fill_parent”
  android:layout_height=”fill_parent”
  android:orientation=”vertical” >
  <Button
    android:id=”@+id/buttonCall”
    android:layout_width=”wrap_content”
    android:layout_height=”wrap_content”
    android:text=”call 0377778888" />
</LinearLayout>
```

Following will be the content of ***res/values/strings.xml*** *to define two new constants −*

```
<?xml version=”1.0" encoding=”utf-8"?>
<resources>
  <string name=”app_name”>My Application</string>
```

```
</resources>
```

Following is the default content of ***AndroidManifest.xml*** −

```
<?xml version=”1.0" encoding=”utf-8"?>
<manifest xmlns:android=”http://schemas.android.com/apk/res/android”
  package=”com.example.saira_000.myapplication” >
    <uses-permission android:name=”android.permission.CALL_PHONE” />
    <application
    android:allowBackup=”true”
    android:icon=”@drawable/ic_launcher”
    android:label=”@string/app_name”
    android:theme=”@style/AppTheme” >
      <activity
      android:name=”com.example.saira_000.myapplication.MainActivity”
      android:label=”@string/app_name” >
        <intent-filter>
        <action android:name=”android.intent.action.MAIN” />
        <category android:name=”android.intent.category.LAUNCHER” />
      </intent-filter>
       </activity>
```

</application>

</manifest>

5.10. Android Telephony

A telephony application in Android is responsible for voice communication through the telephone network. It also provides information about the device and its details, including the IMEI and network type. In addition, Android telephony provides information about SMS. This article will focus on a few of the most important features of an Android telephony application. Let's look at each of them in more detail.

The telephony app is an integral part of the Android system, and it's used by many developers. Android developers often want to use their device as a telephone. Using built-in intents, they can dial outbound calls, intercept, and modify calls. Android also has several other features to help developers build mobile applications. If you've got an Android device, you're already halfway there.

Telephony is the technology that lets us communicate with others over long distances. Advances in wireless networking and affordable computing power have radically changed this process. Telephony used to mean speaking on an analog telephone, but with the advent of digital phones, it expanded to include voicemail and video conferencing, as well as video conferencing with multiple people.

The Android telephony API provides access to important information about a phone, including IMEI, network state, and SMS. It allows developers to dial outbound calls and modify existing ones. Unlike the iPhone, Android's telephony API does not require calling a specific phone number. Instead, the developer can invoke the built-in dialer and populate it with the number provided. The application then makes a call.

To implement a listener for a call, an application must implement the Phone State Listener interface. This interface provides a handle to telephony-related information on a phone, and allows an application to register a Phone State Listener to receive notifications when a telephony-related state changes. The methods in this class provide access to a phone's data, including a caller's ID and the status of the call.

The Telephony Manager class provides information on the phone's state, including its serial number and network provider code. It also provides methods for application developers to attach event listeners to their applications. The Phone State Listener is one such event listener. This can be used to notify applications when phone service is available, or when a call is started or already in progress. There are several other methods that can be called from within the Telephony Manager class.

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

You can use **ACTION_DIAL** action instead of ACTION_CALL, in that case you will have option to modify hardcoded phone number before making a call instead of making a direct call.

Intent Object – Data/Type to make Phone Call

To make a phone call at a given number 91-000-000-0000, you need to specify **tel:** as URI using setData() method as follows −

```
phoneIntent.setData(Uri.parse("tel:91-000-000-0000"));
```

The interesting point is that, to make a phone call, you do not need to specify any extra data or data type.

Example

Following example shows you in practical how to use Android Intent to make phone call to the given mobile number.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

he content of the modified main activity file **src/MainActivity.java**.

```
package com.example.saira_000.myapplication;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.support.v4.app.ActivityCompat;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

public class MainActivity extends AppCompatActivity {

  private Button button;

  @Override

  protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    button = (Button) findViewById(R.id.buttonCall);

    button.setOnClickListener(new View.OnClickListener() {

      public void onClick(View arg0) {
```

```
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:0377778888"));
        if (ActivityCompat.checkSelfPermission(MainActivity.this,
          Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
            return;
          }
          startActivity(callIntent);
      }
    });
  }
}
```

Following will be the content of ***res/layout/activity_main.xml*** *file −*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <Button
    android:id="@+id/buttonCall"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="call 0377778888" />
</LinearLayout>
```

Following will be the content of ***res/values/strings.xml*** *to define two new constants −*

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">My Application</string>
</resources>
```

Following is the default content of ***AndroidManifest.xml*** *−*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.saira_000.myapplication" >
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

```
 <activity
android:name="com.example.saira_000.myapplication.MainActivity"
android:label="@string/app_name" >
   <intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
  </activity>
</application>
</manifest>
```

5.11. Performance Memory Management

Mobile Application Performance is the user's perception of how well an application should perform. It can be related to how responsive it is, how quickly it starts up, how well it uses device memory, how well it uses device power, and so on.

Memory management is the hardest facet of mobile development. Mobile/tablet devices, from cheaper ones to the most expensive ones, have a limited amount of dynamic memory compared to our personal computers. While you are developing your application, you need to consider this constraint by freeing the resources you do not need and minimize them not to hinder the user experience. If you fail to manage your application's memory resources which are provided by the mobile platform, you will end up with frozen UI, abrupt crashes, and dissatisfied users. Also, your application's poor handling will affect other idle background applications; they will get closed by the system to provide you with more memory.

Performance testing checks the total effectiveness of an application in relation to its individual throughput, individual response time and the availability.

Perceived performance such as

- The time taken for the startup form to get rendered.
  - The time it takes for a network call to get executed.
  - The number of network calls made to get the required data.
  - How easily are the events and gestures captured in the application?

Memory management requires that the programmer provides ways to dynamically allocate portions of memory to programs when requested, and free it for reuse when it is no longer needed. In any advanced computer system, where more than a single process might be running at any given point in time, this is critical.

5.12. Android Notifications and Alarms

Every mobile phone is associated with an alarm app. We will create this app using android studio. Android Studio provides a great unified environment to build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto because it provides a very large number of app building features and it is also very easy to use.

the code for the **activity_main.xml** file.

XML

```
<?xml version="1.0" encoding="utf-
8"?> <LinearLayout     xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"    android:layout_height="m
atch_parent"    android:orientation="vertical">     <!--Added Time
picker just to pick the alarm time-->    <!--gravity is aligned to
center--
>    <TimePicker        android:id="@+id/timePicker"        android
:layout_width="wrap_content"        android:layout_height="wrap_con
tent"        android:layout_gravity="center" />    <!--Added Toggle
Button to set the alarm on or off-->    <!--ByDefault toggleButton
is set to false--
>    <ToggleButton        android:id="@+id/toggleButton"        and
roid:layout_width="wrap_content"        android:layout_height="wrap
_content"        android:layout_gravity="center"        android:lay
out_margin="20dp"        android:checked="false"        android:onC
lick="OnToggleClicked" />    <!--"OnToggleClicked" method will be
implemented in MainActivity.java --></LinearLayout>
```

```
import android.app.AlarmManager;

import android.app.PendingIntent;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;
import android.widget.ToggleButton;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Calendar;
public class MainActivity extends AppCompatActivity {
    TimePicker alarmTimePicker;
    PendingIntent pendingIntent;
    AlarmManager alarmManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        alarmTimePicker = (TimePicker) findViewById(R.id.timePicker);
        alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
    }
    // OnToggleClicked() method is implemented the time functionality
    public void OnToggleClicked(View view) {
        long time;
        if (((ToggleButton) view).isChecked()) {
            Toast.makeText(MainActivity.this, "ALARM ON", Toast.LENGTH_SHORT).show();
            Calendar calendar = Calendar.getInstance();
```

```
// calendar is called to get current time in hour and minute
calendar.set(Calendar.HOUR_OF_DAY, alarmTimePicker.getCurrentHour());
calendar.set(Calendar.MINUTE, alarmTimePicker.getCurrentMinute());
// using intent i have class AlarmReceiver class which inherits
// BroadcastReceiver
Intent intent = new Intent(this, AlarmReceiver.class);
// we call broadcast using pendingIntent
pendingIntent = PendingIntent.getBroadcast(this, 0, intent, 0);
time = (calendar.getTimeInMillis() – (calendar.getTimeInMillis() % 60000));
if (System.currentTimeMillis() > time) {
   // setting time as AM and PM
   if (calendar.AM_PM == 0)
      time = time + (1000 * 60 * 60 * 12);
   else
      time = time + (1000 * 60 * 60 * 24);
}
// Alarm rings continuously until toggle button is turned off
alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,time,10000,      pending
Intent);
// alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() +
(time * 1000), pendingIntent);
} else {
alarmManager.cancel(pendingIntent);
Toast.makeText(MainActivity.this," ALAR MOFF",
Toast.LENGTH_SHORT).show();
}}}
```

Chapter Six

6. Graphics and multimedia Graphics

6. 1. Introduction

After completing this chapter, Students must able to:

- Describe performance and multithreading
- Explain multimedia
- Understand mobile agents and peer-to-peer architecture

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In Many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the late 1980s, when scientists and engineers realized that they could not interpret the data and prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

Creating and reproducing pictures, however, presented technical problems that stood in the way of their widespread use. Thus, the ancient Chinese proverb "a picture is worth ten thousand words" became a cliché in our society only after the advent of inexpensive and simple technology for producing pictures—first the printing press, then Photography. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television; it has the added advantage that, with the computer, we can make pictures not only of concrete, "real-world" objects but also of abstract, synthetic objects, such as mathematical surfaces in 4D and of data that have no inherent geometry, such as survey results. Furthermore, we are not confined to static images. Although static pictures are a good means of communicating information, dynamically varying pictures are frequently even better–to time-varying phenomena, both real (e.g., growth trends, such as nuclear energy use in the United States or population movement form cities to suburbs and back to the cities). Thus, a movie can show changes over time more graphically than can a sequence of slides. Thus, a sequence of frames displayed on a screen at more than 15 frames per second can convey smooth motion or changing form better than can a jerky sequence, with several seconds between individual frames. The use of dynamics is especially effective when the user can

control the animation by adjusting the speed, the portion of the total scene in view, the amount of detail shown, the geometric relationship of the objects in the another, and so on. Much of interactive graphics technology therefore contains hardware and software for user-controlled motion dynamics and update dynamics. With motion dynamics, objects can be moved and tumbled with respect to a stationary observer. The objects can also remain stationary and the viewer can move around them. Pan to select the portion in view, and zoom in or out for more or less detail, as though looking through the viewfinder of a rapidly moving video camera. In many cases, both the objects and the camera are moving. A typical example is the flight simulator, which combines a mechanical platform supporting a mock cockpit with display screens for windows.

Computers control platform motion, gauges, and the simulated world of both stationary and moving objects through which the pilot navigates. These multimillion-dollar systems train pilots by letting the pilots maneuver a simulated craft over a simulated 3D landscape and around simulated vehicles. Much simpler flight simulators are among the most popular games on personal computers and workstations. Amusement parks also offer "motion-simulator" rides through simulated terrestrial and extraterrestrial landscapes. Video arcades offer graphics-based dexterity games and racecar-driving simulators, Video Games exploiting interactive motion dynamics: The player can change speed and direction with the "gas pedal" and "steering wheel," as trees, buildings, and other cars go whizzing by. Similarly, motion dynamics lets the user fly around the through buildings, molecules, and 3D or 4D mathematical space. In another type of motion dynamics, the "camera" is held fixed, and the objects in the scene are moved relative to it. For example, a complex mechanical linkage, such as the linkage on a stream engine, can be animated by moving or rotating all the pieces appropriately. Update dynamics is the actual change of the shape, color, or other properties of the objects being viewed. For instance, a system can display the deformations of an airplane structure in flight or the state changes in a block diagram of a nuclear reactor in response to the operator's manipulation of graphical representations of the many control mechanisms. The smoother the change, the more realistic and meaningful the result. Dynamic interactive graphics offers a large number of user-controllable modes with which to encode and communicate information: the 2D or 3D shape of objects in a picture, their gray scale or color, and the time variations of these properties. With the recent development of digital signal processing (DSP) and audio synthesis chips, audio feedback can now be provided to augment the graphical feedback and to make the simulated environment even more realistic. Interactive computer graphics thus permits

extensive, high-bandwidth user-computer interaction. This significantly enhances our ability to understand data, to perceive trends, and to visualize real or imaginary objects–indeed, to create "virtual worlds" that we can explore from arbitrary points of view. By making communication more efficient, graphics make possible higher-quality and more precise results or products, greater productivity, and lower analysis and design costs.

6.2. Graphics and multimedia Graphics

The use of computer graphics is wide spread. It is used in various areas such as industry, business, government organizations, education, entertainment and most recently the home. Let us discuss representative uses of computer graphics in brief. User friendliness is one of the main factors underlying the success and popularity of any system. It is now a well-established fact that graphical interfaces provide in attractive and easy interaction between users and computers. The built-in graphics provided with user interfaces use visual control items such as buttons, menus, icons, scroll bar etc, which allows user to interact with computer only by mouse-click. Typing is necessary only to input text to be stored and manipulated. In industry, business, government and educational organizations, computer graphics is most commonly used to create 2D and 3D graphics of mathematical, physical and economic functions in form of histograms, bars, and pie-chats. These graphs and charts are very useful for decision making. The desktop publishing on personal computers allows the use of graphics for the creation and dissemination of information. Many organizations do the in-house creation and dissemination of documents. The desktop publishing allows user to create documents which contain text, tables, graphs, and other forms of drawn or scanned images or pictures. This is one approach towards the office automation. The computer-aided drafting uses graphics to design components and systems electrical, mechanical, electromechanical and electronic devices such as automobile bodies, structures of building, airplane, slips, very large-scale integrated chips, optical systems and computer networks. Use of graphics in simulation makes mathematic models and mechanical systems more realistic and easier to study. The interactive graphics supported by animation software proved their use in production of animated movies and cartoons films. There is lot of development in the tools provided by computer graphics. This allows user to create artistic pictures which express messages and attract attentions. Such pictures are very useful in advertising. By the use of computer now it is possible to control various processes in the industry from a remote-control room. In such cases, process systems

and processing parameters are shown on the computer with graphic symbols and identification. This makes it easy for operator to monitor and control various processing parameters at a time.

- **Interactive Computer Graphics**

  In interactive computer graphics, users have some control over the picture i.e. user can make any changes or alterations in the produced image. One example of it is the ping pong game. At the hardware level, a computer receives input from interaction devices and outputs images to a display device.

  The software has three components. The first is the application program, it creates, stores into, and retrieves from the second component, the application model, which represents the graphic primitive to be shown on the screen. The application program also handles user input. It produces views by sending to the third component, the graphics system, a series of graphics output commands that contain both a detailed geometric description of what is to be viewed and the attributes describing how the objects should appear.

- **Passivecomputer Graphics–**

  Passive computer graphics is an operation on computer graphics that transfers automatically and without operator intervention. Non-interactive computer graphics or passive computer graphics involves one-way communication between the computer and the user. The picture is produced on the monitor and the user does not have any control over the produced picture.

Applications Of Computer Graphics –

Computer graphics has numerous of diverse applications listed in the previous section which differ in a variety of ways, and a number of classification is by type (dimensionality) of the object to be represented and the kind of picture to be produced. Furthermore, computer graphics is used today in many different areas of industry, business, government, education, entertainment, and most recently, the home. The list of applications is enormous and is growing rapidly as computers with graphics capabilities become commodity products. Let`s have a look at a representative sample of these areas:

1. Cartography

2. User interfaces
3. Plotting in Business, Science, and Technology
4. Office automation and electronic publishing
5. Computer-aided drafting and design
6. Art and Commerce
7. Process control

Multimedia Applications –

As the name suggests, multimedia is a set of more than one media element used to produce a concrete and more structured way of communication. Multimedia is nothing but the processing and presentation of information in a more structured and understandable manner using more than one media such as text, graphics, animation, audio, and video.

Multimedia Software :

Multimedia software is sometimes broadly grouped as:

1. a)Multimedia Authoring Tools
2. b)Multimedia Tools for the Web
3. c)Multimedia Presentation Software

- **Multimedia Hardware :**

For producing multimedia you need hardware, software, and creativity. Thus, following are the specific multimedia hardware:

1. Central Processing Unit
2. Monitor
3. Video Grabbing Card
4. Sound Card
5. CD-ROM Drive
6. Scanner
7. Touchscreen

Components Of Multimedia –

1. Text
2. Graphics
3. Animation
4. Video

6.3. Performance and Multithreading

Developers' multithread Android applications in order to improve their performance and usability. By spinning off processor- or resource-intensive tasks into their own threads, the rest of the program can continue to operate while these processor intensive tasks finish working. This can include separating the thread for the graphical user interface from the threads that fetch network data. The JVM automatically manages these multiple tasks, so Android developers do not have to concern themselves with optimizing thread performance for a particular set of hardware.

Android can use multiple CPU cores for multithreading, but the kernel and JVM handle that process, not the developer himself. An internal multithreading design will improve the program's basic performance, but the device upon which it actually runs will determine its speed.

Working on multiple tasks at the same time is **Multitasking**. In the same way, multiple threads running at the same time in a machine is called **Multi-Threading**. Technically, a thread is a unit of a process. Multiple such threads combine to form a process. This means when a process is broken, the equivalent number of threads are available.
For example, *Autocorrect* is the process where the software looks for the mistakes in the current word being typed. Endlessly checking for the mistake and providing suggestions at the same time is an example of a Multi-Threaded process.

Sample Android App:

Let's try to visualize Multi-Threading with the help of an Android App. In the below example, 3 Threads start at the same time on a button click and work concurrently.

Approach:

Step 1: Add the below code in **activity_main.xml**. Here we add three TextViews and a button.

XML

```
<?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" tools:context=".MainActivity"> <TextView android:id="@+id/tv1" android:layout_width="75sp" android:layout_height="75sp" android:background="@color/colorPrimary" android:textColor="@color/colorAccent" android:gravity="center" android:textSize="10sp" android:layout_centerVertical="true" android:layout_toLeftOf="@id/tv2" /> <TextView android:id="@+id/tv2" android:layout_width="75sp" android:layout_height="75sp" android:background="@color/colorPrimary" android:textColor="@color/colorAccent" android:gravity="center" android:textSize="10sp" android:layout_centerInParent="true" /> <TextView android:id="@+id/tv3" android:layout_width="75sp" android:layout_height="75sp" android:background="@color/colorPrimary" android:textColor="@color/colorAccent" android:gravity="center" android:textSize="10sp" android:layout_centerVertical="true" android:layout_toRightOf="@id/tv2" /> <Button android:id="@+id/btnStart" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Start" android:layout_centerHorizontal="true" android:layout_below="@id/tv2" /> </RelativeLayout>
```

Step 2: Add the below code in **MainActivity**. Here, three threads are made, each thread keeps updating the respective TextView every second (declared in the code) when the button is clicked. These Threads keep running until the button is clicked again (i.e."Stop").

- Java

```
class MainActivity : AppCompatActivity()
{    @SuppressLint("SetTextI18n")    override fun
onCreate(savedInstanceState:
```

```
Bundle?) { super.onCreate(savedInstanceState) setCon
tentView(R.layout.activity_main) // Assigning Layout
elements val tv1 =
findViewById<TextView>(R.id.tv1) val tv2 =
findViewById<TextView>(R.id.tv2) val tv3 =
findViewById<TextView>(R.id.tv3) val btn =
findViewById<Button>(R.id.btnStart) // Boolean for Button
(initially False) var
boolbtn = false // Button onClick
action btn.setOnClickListener { // Button
(True) boolbtn = !boolbtn // Case where
Button is
False if (!boolbtn) { tv1.t
ext = "Stopped1" tv2.text =
"Stopped2" tv3.text = "Stopped3" btn.text =
"Start" } // Case where Threads are
running else { // Setting the button
text as "Stop" btn.text =
"Stop" // Thread
1 Thread(Runnable
{ // Runs only when Button is
True while (boolbtn)
{ runOnUiThread
{ tv1.text =
"Started1" }
Thread.sleep(1000) runOnUiThr
ead {
tv1.text =
"Activity1" }
Thread.sleep(1000) }
}).start() // Thread
2 Thread(Runnable
{ // Runs only when Button
is //
True while (boolbtn)
{ runOnUiThread
```

```
            {                                                tv2.text =
"Started2"                                          }
          Thread.sleep(1000)                                                runO
nUiThread                                          {
            tv2.text =
"Activity2"                                          }
           Thread.sleep(1000)                                      }
                  }).start()                          // Thread
3                           Thread(Runnable
{                                      // Runs only when Button
is                                     //
True                                    while (boolbtn)
{                                       runOnUiThread
                  {                                                 tv3.text =
"Started3"                                          }
              Thread.sleep(1000)
  runOnUiThread                                          {
                        tv3.text =
"Activity3"                                          }
               Thread.sleep(1000)                                      }
                            })                                .start()
      }         }     } }
```

6.4. Graphics and UI Performance

The user interface of your app is likely influenced by designers, developers, usability studies, and testing—just about anyone is happy to add input/feedback to how your app looks. As the UI of your app is your connection to your customers, it defines your brand and it requires careful planning. However simple (or complicated) the UI of your app is, it's important that your UI design is built to be performant.

As a developer, your task is to work with the UI/UX team and build an app that follows its design parameters on every Android device. We've already (briefly) discussed the pitfalls of the many screen sizes in the Android ecosystem and the challenges that exist there. But how about UI performance? How does the UI that your designers designed (and you built) run? Do

the pages load quickly? Do they respond in a fast and smooth way? In this chapter, we'll discuss how to optimize your UI for fast rendering and scrolling/animations, and the tools you can use to profile your screen and UI performance.

UI Performance Benchmarks

Like all performance goals, it is important to understand the performance goals associated with UI. Saying "my app needs to load faster" is great, but what are the expectations of the end user, and are there concrete numbers you can apply to those expectations? In general, we can fall back on studies of the psychology of human interactions. These studies have shown that users perceive delays of 0 – 100 ms as instantaneous and delays of 100 – 300 ms as sluggish; delays between 300 – 1,000 ms indicate to users that "the machine is working," whereas delays of 1,000+ ms lead users to feel a context switch.

As this is basic human psychology, it seems to be a good metric to start with for page/view/app loading times. Ilya Grigorik has a great presentation about building mobile websites to take just "1,000 ms to Glass." If your web page can load in 1 second, you win the human perception battle, and now you must wow your customers with great content. Additional research has shown that >50% of users begin abandoning websites if no content has loaded in 3–4 s. Applying the same argument to apps tells us that the faster you can get your app to start, the better. In this chapter, we'll focus just on the UI loading. There may be tasks that must run in the background, files to be downloaded from the Internet, and so on, we'll cover optimizing these tasks (or ways to keep these tasks from blocking the rendering) in future chapters.

Jank

In addition to getting content on the screen as quickly as possible, it has to render in a smooth way. The Android team refers to jerky, unsmooth motion as *jank*, and this is caused by missing a screen frame refresh. Most Android devices refresh the screen 60 times a second (there are undoubtedly exceptions—earlier Android devices were sometimes in the 50 or less fps range). Because the screen is refreshed every 16 ms (1 s/60 fps = 16 ms per frame), it is crucial to ensure that all of your rendering can occur in less than 16 ms. If a frame is skipped, users experience a jump or skip in the animation, which can be jarring. In order to keep your

animations smooth, we'll look at ways to ensure the entire screen renders in 16 ms. In this chapter, we'll diagnose common issues and illustrate how to remove jank from your UI.

UI and Rendering Performance Updates in Android

One of the major complaints of early Android releases was that the UI—especially touch interactions and animations—were laggy. As a result, as Android has matured, the developers have invested a great deal of time and effort to make the user interaction as fast and seamless as possible. Let's walk through a few of the improvements that have been added in various releases of Android to improve the user interaction:

- On devices running Gingerbread or earlier, the screen was drawn completely in software (there was no GPU requirement). However, device screens were getting larger and pixel density was increasing, placing strain on the ability of the software to render the screen in a timely manner.
- Honeycomb added tablets, further increasing screen sizes. To account for this, GPU chips were added, and apps had the option to run the rendering using full GPU hardware acceleration.
- For apps targeting Ice Cream Sandwich and higher, GPU hardware acceleration is on by default; pushing most rendering out of the software and onto dedicated hardware sped up rendering significantly.
- Jelly Bean 4.1 (and 4.2) "Project Butter" made further improvements to avoid jank and jitter, in order to make your apps "buttery smooth." By improving timing with VSYNC (better scheduling frame creation) and adding additional frame buffering, Jelly Bean devices skip frames less often. When building these improvements, the Android team built a number of great tools to measure screen drawing, the new VSYNC buffering and jank, and released these tools to developers.

We'll review all of these changes, the tools introduced, and what they mean to the average Android developer. As you might imagine, the goals from these updates were as follows:

- Lower the latency of screen draws
- Create fast, consistent frame rates to avoid jank

When the Android team was working on all of the improvements to screen rendering and UI performance, they needed tools to quantify the improvements that they made to the OS. To their credit, they have included these tools in the Android SDK so that developers can test their

apps for rendering performance issues. As we walk through the different ways to improve app performance, we'll use these tools with example apps to explain how they work.

Building Views

I assume that you are familiar with the XML layout builder in Android Studio, and how to build views with the different tools in Android Studio (Eclipse) to look at those views. In Figure 4-1, you can see a simple app with a series of nested views. When building your views, it is important to look at the Component Tree in the upper-right of the screen. The more nested your views become, the more complicated the View Tree becomes, and the longer it will take to render.

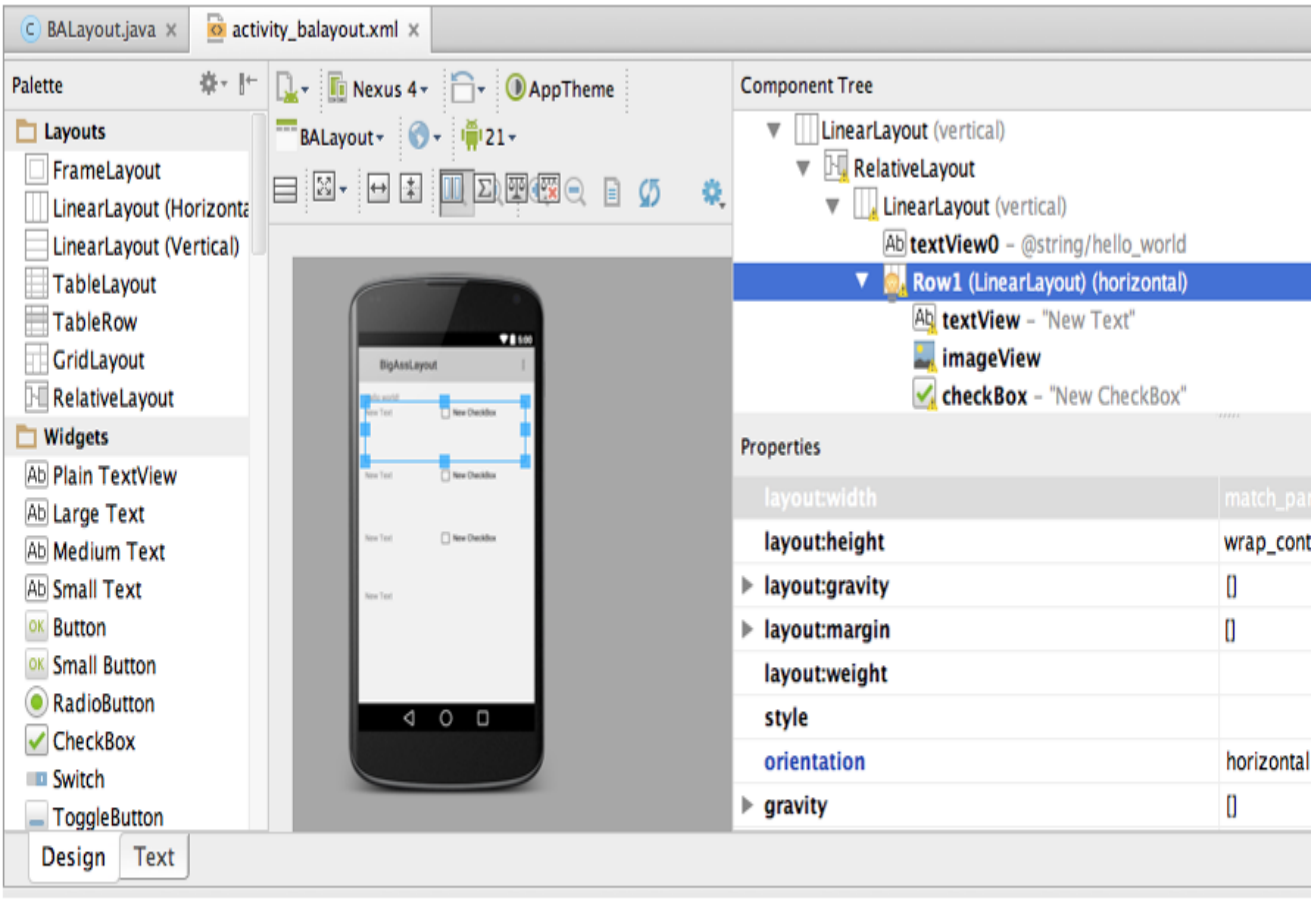


Figure 29:Design view of an app layout

For each view in your app, Android goes through three steps to render on the screen: measure, layout, and draw. If you imagine your XML layout hierarchy in your app, the measure starts at the top node and walks the render tree of the layout: measuring the dimensions of each view to

be displayed on the screen (in : LinearLayout; RelativeLayout; LinearLayout; then branching for textView0 and the LinearLayout Row1—which has three further children). Each view will provide dimensions to the parent for positioning. If a parent view discovers an issue in the measurements of its dimensions (or that of its children), it can force every child (grandchild, great-grandchild, etc.) to remeasure in order to resolve the issue (potentially doubling or tripling the measurement time). This is the reason a flat (less nested) view tree is valuable. The deeper the nodes for the tree, the more nested the measurement, and the calculation times are lengthened (especially on remeasurements). We'll examine some examples of how remeasurement can really hurt rendering as we look through the views.

Remeasuring Views

There does not have to be an error for a remeasure to occur. RelativeLayouts often have to measure their children twice to ensure that all child views are laid out properly. LinearLayouts that have children with layout weights also have to measure twice to get the exact dimensions for the children. If there are nested LinearLayouts or RelativeLayouts, the measure time can grow in an exponential fashion (four remeasures with two nested views, eight remeasures with three nested views, etc.).

Once the views are measured, each view will layout its children, and pass the view up to its parent—all the way back up to the root view. Once the layout is completed, each view will be drawn on the screen. Note that all views are drawn, not just the ones that are seen by your customers. We'll talk about that issue in "Overdrawing the Screen". The more views your app has, the more time it will take to measure, layout, and draw. To minimize the time this takes, it is important to keep the render tree as flat as possible, and remove all views that not essential to rendering. Removing layers of the layout tree will go a long way in speeding up the painting of your screen. Ideally the total measure, layout, and draw should be well below the 16 ms threshold—ensuring smooth scrolling of your UI on the screen.

While it is possible to look at the node view of your layout as XML (like in Figure 4-1), it can be difficult to find redundant views. In order to find these redundant views (and views that add delay to screen rendering), the Hierarchy Viewer tool in Android Studio Monitor can greatly

help you visualize the views in your Android app to resolve these issues (Monitor is a standalone app that is downloaded as a part of Android Studio).

Hierarchy Viewer

The Hierarchy Viewer is a handy way to visualize the nesting behavior of your various views on a screen. The Hierarchy Viewer is a great tool to investigate the construction of your view XML. It is available in Android Studio Monitor, and requires a device with a developer build of Android on it. See "Rooted Devices/Engineering/Developer Builds" for details on what this entails. There is also a class from Googler Romain Guy that allows you to test a debug version of your app. All of the views and screenshots using the Hierarchy View in the subsequent sections are taken from a Samsung Note II running 4.1.2 Jelly Bean. By testing screen rendering on an older device (with a slower processor), you can be sure that if you meet rendering thresholds on this device, your app will likely render well on all Android devices.

6.5. Android Graphics

Android provides a huge set of 2D-drawing APIs that allow you to create graphics.

Android has got visually appealing graphics and mind blowing animations.

The Android framework provides a rich set of powerful APIS for applying animation to UI elements and graphics as well as drawing custom 2D and 3D graphics.

Following are the three animation systems used in Android applications:

1. Property Animation

2. View Animation

3. Drawable Animation

1. Property Animation

- Property animation is the preferred method of animation in Android.

- This animation is the robust framework which lets you animate any properties of any objects, view or non-view objects.
- The **android.animation** provides classes which handle property animation.

2. View Animation

- View Animation is also called as **Tween Animation.**
- The **android.view.animation** provides classes which handle view animation.
- This animation can be used to animate the content of a view.
- It is limited to simple transformation such as moving, re-sizing and rotation, but not its background color.

3. Drawable Animation

- Drawable animation is implemented using the AnimationDrawable class.
- This animation works by displaying a running sequence of 'Drawable' resources that is images, frame by frame inside a view object.
- **Canvas**
- Android graphics provides low level graphics tools such as canvases, color, filters, points and rectangles which handle drawing to the screen directly.
- The Android framework provides a set of 2D-DRAWING APIs which allows user to provide own custom graphics onto a canvas or to modify existing views to customize their look and feel.

There are two ways to draw 2D graphics,

1. Draw your animation into a View object from your layout.
1. Draw your animation directly to a Canvas.

Some of the important methods of Canvas Class are as follows

i. *drawText() drawRoundRect() drawCircle() drawRect() drawBitmap()drawARGB()*

ii. You can use these methods in onDraw() method to create your own custom user interface.

iii. Drawing an animation with a View is the best option to draw simple graphics that do not need to change dynamically and are not a part of a performance-intensive game. It is used when user wants to display a static graphic or predefined animation.

iv. Drawing an animation with a Canvas is better option when your application needs to re-draw itself regularly. For example video games should be drawing to the Canvas on its own.

Example

What is it about?

Create a new Java class that should extend from View class. Override the onDraw() method. In this method, you can use Canvas class to draw the different shapes.

File name: MyView.java

```
public class MyView extends View

{
    public MyView(Context context)

{
        super(context);

 // TODO Auto-generated constructor stub

}
     @Override
     protected void onDraw(Canvas canvas)

{

  // TODO Auto-generated method stub

super.onDraw(canvas);
          int radius;
```

```
radius = 50;

        Paint paint = new Paint();

        paint.setStyle(Paint.Style.FILL);

        paint.setColor(Color.parseColor("#CD5C5C"));

        canvas.drawCircle(150,200, radius, paint);

        canvas.drawRoundRect(newRectF(20,20,100,100), 20, 20, paint);

        canvas.rotate(-45);

        canvas.drawText("TutorialRide", 40, 180, paint);

        canvas.restore();

    }
}
```

6.6. Multimedia Mobile Agents and Peer-to-Peer Architecture

Peer-to-peer (P2P) content-distribution applications are widely spread on stationary computers in fixed networks today. But also mobile devices (MDs) in cellular radio networks, e.g. phones or personal digital assistants, are potential candidates for participating in these popular P2P networks. Their integration into popular P2P content-distribution networks (CDNs) with large user communities is a special instance of the mobile peer-to-peer (mobile P2P) research field. Although current MDs mostly have enough hardware resources to join such P2P networks, they are barely able to benefit from them. Their participation together with a large number of stationary computers with high-bandwidth links, increases heterogeneity within P2P networks, which causes performance problems, especially for MDs. In contrast to this heterogeneity, P2P systems apply homogeneous logical structures (the overlay), in which computers and links are considered to be equal, having similar properties and capabilities.

Mobile P2P approaches that focus on cellular radio networks are partitioned in three categories with respect to the support they provide to MDs: 1) no additional support, 2) support from the

P2P protocol, and 3) other support not being part of the P2P protocol. No support: Solutions of this category are reshaping P2P client software to the requirements of MDs, considering the MDs limitations, whereas the P2P protocol remains unchanged. MDs join P2P networks as ordinary peers without getting further support. In this paper, this approach is called the straightforward approach or no support solution. Symella (http://symella.aut.bme.hu), e.g., is such a P2P client for the Gnutella [2] P2P file-sharing network.

Components

We have to consider not only the distribution of learning contents but also the distribution of functions to provide above services. Our system consists of mobile agents and user interface program. In our system, functions are provided as the part of mobile agent. A mobile agent migrates from one node to another node with a function it provides. These mobile agents are implemented in the mobile agent framework called Maglog [4]. Figure 2 shows an interaction between agents.

Node Agent (NA): Each node has one node agent. It manages the zone information of a CAN and forwards messages to the Category Agents in the node.

Exercise Agent (EA): Each Exercise Agent has questions and functions to score user's answers, to tell the correct answers, and to show some related information about the exercise. **Category Agent (CA)**: One category agent is installed in one learning category. A category agent manages exercise agents related on its category and dispatches the learning request to them.

User Agent (UA): Each user has its own User Agent. A User Agent manages user's learning information that includes login name, password, IP address of the user's computer, online/offline status, and log of learning and/or a list of created exercises.

Group Agent (GA): Each group agent manages user agents.

User Interface: One user interface is on each node which a user logs in as a student. It provides a user interface program for learning.

6.7. Android Multimedia

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called **MediaPlayer**.

Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, we have to call a static Method **create()** of this class. This method returns an instance of MediaPlayer class. Its syntax is as follows −

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
```

The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name **raw** and place the music file into it.

Once you have created the Mediaplayer object you can call some methods to start or stop the music. These methods are listed below.

```
mediaPlayer.start();

mediaPlayer.pause();
```

On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, you have to call **reset()** method. Its syntax is given below.

```
mediaPlayer.reset();
```

Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files.

Android framework to implement multimedia capture and playback.

Media Playback

How to play audio and video in your application.

JetPlayer

How to play interactive audio and video in your application using content created with JetCreator.

Camera

How to use a device camera to take pictures or video in your application.

Audio Capture How to record sound in your application

Chapter Seven
7. Location

7.1. Introduction

After completing this chapter, Students must able to:

- Describe mobility and location-based services

Location, location, location. The changing location of the user and the device used by the user make mobile applications fundamentally different from their stationary counterparts. Yet, most software developers, even those who have some experience with mobile applications, have little experience and understanding in how location-based information is gathered and distributed and how this information may be utilized by mobile applications. Although we will not be able to cover all aspects of location-based information in mobile applications, we will try to tackle the basic problems in this chapter.

If you have looked into developing mobile applications, you have certainly heard of "location-based services." The UTMS Forum defines location-based services as follows:

This definition is somewhat narrow as it limits location-based services to 3G services. There are many location-based services that do not have any relationship with 3G services. So, location-based services are those things that provide the mobile device, the mobile application, and the mobile user with location information about themselves or other devices, applications, and users.

7.2. Location Mobility

Using the Google Play services location APIs, your app can request the last known location of the user's device. In most cases, you are interested in the user's current location, which is usually equivalent to the last known location of the device.

Specifically, use the fused location provider to retrieve the device's last known location. The fused location provider is one of the location APIs in Google Play services. It manages the underlying location technology and provides a simple API so that you can specify

requirements at a high level, like high accuracy or low power. It also optimizes the device's use of battery power.

Create location services client

In your activity's onCreate() method, create an instance of the Fused Location Provider Client as the following code snippet shows.

Java

```
private FusedLocationProviderClient fusedLocationClient;

@Override

protected void onCreate(Bundle savedInstanceState) {

    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

}
```

The `getLastLocation()` method returns a `Task` that you can use to get a `Location` object with the latitude and longitude coordinates of a geographic location. The location object may be `null` in the following situations:

- Location is turned off in the device settings. The result could be `null` even if the last location was previously retrieved because disabling location also clears the cache.
- The device never recorded its location, which could be the case of a new device or a device that has been restored to factory settings.

- Google Play services on the device has restarted, and there is no active Fused Location Provider client that has requested location after the services restarted. To avoid this situation, you can create a new client and request location updates yourself. For more information, see Receiving Location Updates.

7.3. Location Based Service and Android

Google has a list of location-based services, including:

Google Location Accuracy for your Android device (a.k.a. Google Location Services): To get a more accurate location for your phone, learn how to manage Location Accuracy.

Emergency Location Service for your Android device: Learn how to manage Android Emergency Location Service.

Earthquake alerts for your Android device: To get updates for nearby earthquakes on your phone, learn how to manage Earthquake alerts.

Use location for time zone on your Android device: To get time zone updates based on location, learn how to manage location for time zone.

Location History for your Google Account: To find and manage the places your phone has been, learn how to turn on Location History.

Location Sharing for Google Maps: To let others know where your phone is, learn how to share your real-time location via Google Maps.

Location in Search: To get more helpful results when you search on Google, learn how to manage location permissions for websites and apps.

Wi-Fi scanning and Bluetooth scanning: To help apps get better location info, learn how to scan for network or bluetooth devices.

Chapter Eight

8. Putting It All Together (as time allows) Packaging and Deploying

8.1. Packaging and Deploying

After completing this unit, Student must able to:

- Explain android packaging and deploying mobile application
- Identify performance best practices

From the **Packaged Components** page, use the Create Packaged Components wizard to create and immediately deploy packaged components to a runtime environment.

Procedure

1. From the **Deploy** menu, select **Packaged Components**.

   The Packaged Component landing page opens.

- Click the **Create Packaged Component** button.

  The Packaged Components wizard opens. The Component Explorer displays only the list of components eligible for deployment.

- From the Component Explorer, select the check box next to one or more components that you wish to create in the packaged component.

  You can use the search bar in the components list to search for a component by name or component ID, or use the filter icon to narrow results by deployed status, component type, packaged component status, or packaged component ID.

  If the component was created as a packaged component previously, a version history of the component appears in the right-most pane.

- **Optional:** Use the **Action** menu to view details about your selected component and its dependent components, view historical details of all packaged component versions created for the selected component, or deselect the component as a candidate for packaging.
- Click **Next: Add Details**.

The Add Details screen of the wizard opens.

- **Optional:** Click the **View Included Components** action to see the primary component and all dependent components included with your selected component.
- **Optional:** Provide the following information:
  - **Version For All**— Enter the unique version ID that you want to apply to the selected component(s). The version must be unique to previously packaged components and can contain alphanumeric characters. If you do not populate the field, Boomi Integration automatically generates a version number for each component individually, which increments based on the latest revision number respectively.

    Remember that though you may have several components selected at this point for packaging, each component results in its own individual package at the completion of the wizard; only the version number, notes, and sharing options are applied in bulk to each selected packaged component.

  - **Package Notes for All**— include a note about the version of the packaged component you are about to create.
  - **Sharing**— (Disabled by default) enable the Sharing option to allow Process or API component types to be shared to the Process Library or Integration Packs once they are created as a packaged component.
- Once you are finished, click the **Create Packaged Component** button. The button shows the number of components that are selected for creation.

Your selected component(s) is created as a new version of the packaged component, and is added to the version history table on the Packaged Component Screen. Additionally, you are presented with a dialog in which you can choose to immediately deploy the new version of the packaged component.

- Click **Deploy** to deploy the new version of your packaged component.

  The Deploy wizard opens.

1. In the Deployment Environment field, select an environment from the list of available options. Optionally add notes about the deployment that you may want to reference later.
2. Click **Next: Select Versions**.

   The next screen of the Deployments wizard opens.

1. From the Component Explorer, select the check box next to one or more packaged components that you wish to deploy to the environment that you selected in **Step 10**.
2. **Optional:** Use the **Action** menu to compare two versions of a packaged component, or to deselect the component for deployment.
3. **Optional:** Click the **View Details** link in the Selected Version column to view details about the version of the packaged component that is currently selected. If applicable, click the **View Details** link in the Deployed Version column to view details about another version that was most recently deployed.

   Click **Close** on the Package Component Details dialogs to return to the Deployment wizard.

1. **Optional:** Click the **Select different package** link to select a different version of the packaged component in which to deploy. Select a different version of the packaged component to mark it instead as the candidate for deployment and click **Select Package**, or click **Cancel** to return to the Deployments wizard.
2. Click the **Next: Review** button.

   The next screen of the Deployments wizard opens.

1. Review and confirm that all selections and details are correct, then click the **Deploy** button.

   The Deployment Successful dialog appears and informs you that your recent deployment was successful. Click **View Deployments** to see only your recent deployment in the Deployments table, or click **Close** to see all deployments

8.2. Performance Best Practices

Performance – the most significant aspect of a mobile app. The first thing users will notice over and over again is your app's startup time. And they are likely to give a thumbs down to the app if it's slow. As per Nimbledroid, poor app performance has made 86% of users (counting myself in) uninstall their apps.

So, if your app needs to show some content to a user, it has less than 11 seconds to load. But here's the sob story – only 40 out of 100 Android apps start under 2 seconds while 70 take at least 3 seconds. This answers your question of which part of the app development process you should focus the most on: it's Android app performance.

Yes, android devices have multiple cores, so building smooth android apps is really difficult – right? Actually, it's not. You can do a lot of things on Android in different ways, and the only challenge you'll face is picking the best option. If you want to optimize android app performance, you must know what's happening under the hood. Fortunately, you can find bottlenecks, show and measure what's happening with the help of a lot of tools.

Well then, it's time to stop relying on your gut feeling or hoping for the best and know how to improve the performance of your android app that will win the hearts of users.

Tips and Tricks to Improve Android App Performance

- Reduce the App Size
- Optimize Networking
- Optimize Images and Videos
- Optimize Screen Size
- Optimize Offline Model
- Improve Memory usage
- Scale Down the App Launch Time
- Hit a Smooth Frame Rate
- Work Efficiently with Threads
- Identify Latency Killers
- Use StringBuilder instead of String

The smoother your app performs, the harder it will hit the market. Personally, I would not mind spending a couple of extra bucks for exceptionally improved performance, and I've seen clients do so too. Truly the wise said – time is money!

Anyway, please keep in mind that premature optimization during the mobile app development process can spoil the whole game for you. Spending time on micro-optimization won't work for you as well. Try to optimize code that runs frequently. The onDraw() function is such an example that runs every frame with an ideal 60 times/second pace. As drawing goes slower than any other operation, redraw only when you absolutely need to.

Okay, enough theory! Let me cut to the chase and look at some actions you should consider if android app performance matters to you:

Reduce the App Size

Smartphones aren't powerful GPUs yet. And here's a jaw-dropping fact – it's modestly configured mobile phones that consume most of the global Android market share. Even though some of them don't support 2G/3G, users generally avoid downloading huge apps, especially when there's a hurry. So, the less space it consumes on your phone, the better it is for users.

Android App Bundles

Android App Bundle is the latest upload format that consists of all the resources and compiled code. It delays APK building and signing to Play Store. Such features make the job exceptionally easy as app bundles create APKs that are optimized as per the device.

Though the compressed download size of your app should be 150mb or less to publish in the Play Store, minimizing the app size as small as possible is still a great idea to improve your android app performance.

Android Size Analyzer

This is an efficient Android app performance tool that can help you choose and implement a lot of solutions to shrink the size of your app. You can find Android size analyzer as an Android Studio plugin and a standalone JAR (as a command line).

As a Plugin > Install the Android Size Analyzer from the plugin marketplace in Android Studio and use it as a plugin. Run an analysis on your Android app with it, and you will find a tool window with all the suggestions on how to enhance your app performance.

As a Command-Line > Github has the latest version of Android Size Analyzer as a ZIP or TAR file. Extract the ZIP file and run the size-analyzer script so that you can see the size of your files. Additionally, the size analyzer.bat script (windows) is there to run these two commands:

./analyzer check-bundle

./analyzer check project

Handle network requests effectively as your app will use the internet to upload or download data. One of these network requests is cellular data, and another is the battery. Heavy usage of both mobile network and WiFi consumes a lot of battery.

Suppose you have to download a 1 MB file. So you need to stimulate the mobile or WiFi radio to make a network request and download the file. However, the radio won't sleep instantly after the task. For the next 20-40 seconds, it will be in an adequately active state based on your device.

So, can you do something about it? Yes, batching. Prefetch things that users may use and download the data they may need in the next 3-4 minutes so that the radio doesn't wake up every couple of seconds.

You can also improve app performance by editing the batch parameters based on the user's charging condition or internet type. For example, you can't prefetch much data if the user is on cellular internet with a low battery, unless it's WiFi while charging.

Efficient API Design > Android app performance heavily relies on API design. Workarounds on APIs generally bring too much pressure on network restrictions. Hence, ensure to have completeness and consistency while designing the API payloads of your android app.

Start with UI and go all the way to data to design a complete and consistent API payload that will improve android app performance.

Apart from APIs, your Android app loads quite a number of images and videos. Usually, other properties load those images as well, like the website of your app. Images and videos on desktop websites are different (in terms of resolution, screen size, and device type) than on mobile apps.

Hence, you must optimize images and videos in your app for mobile devices with measured and limited resources. When your Android app requests images, ensure they are served with the best size and resolution. You can also use adaptive video buffering and progressive images so that your app can successively download images rather than waiting for the complete image to download. This way, users don't view a blank image due to an improvement in perceived latency.

In the same way, adaptive video streaming techniques effectively modify the video resolution as per the network setup. When bandwidth is low, you can lower the video quality so that users can at least watch the video and not a loading screen.

Additionally, did you know that you can use the WebP format besides JPEG or PNG images as well? Yes, WebP is technically more streamlined, although you need to put extra effort into creating WebP images. And for local network traffic, SVG (Scalable Vector Graphics) format is another good choice that can save your disk space as well.

Lastly, you can use image caching, vector images, libraries like Fresco or Picasso and apply color filters instead of varied sets of colored images. All these are effective ways to improve android app performance.

8.3. Android Field Service App

The Field Service mobile app for Android and iOS is an all-in-one tool for mobile workers on the go. This enterprise-class mobile experience leverages Salesforce in a lightweight design optimized for a modern mobile workforce. Offline capability means that users can keep working without internet connectivity and know that all their changes are saved. And the app is highly customizable, allowing you to tailor it to your unique field service needs.

Service Appointments

- The Recently Viewed service appointments list view shows all service appointments assigned to a user even if the user hasn't viewed them.
- Service appointment list views of more than 2000 don't display labels as dropdown values on the mobile app's Schedule tab.
- The schedule list isn't refreshed unless the ScheduleUpdateFreq timeout value is reached.

Service Reports

- Users can't select a template when creating a service report in the mobile app. Instead, a template is selected based on this order:

1. The work order template
2. The work type template
3. The standard template

- Service report previews have these limitations:

1. Although users can make an offline request to generate a service report, the final PDF is created only when the user's device is online.
2. The app doesn't automatically get the latest data before generating a service report preview, so data in service report previews can be out of date. This limitation applies to online and offline service report previews.
3. If a reference field exceeds the maximum priming depth of 2, it isn't populated.
4. Formula field values can be inaccurate in a preview because they aren't dynamically calculated.
5. Roll-up summary fields are blank.
6. Related lists behave differently in mobile previews.

- The order of records in related lists can differ between the preview and the printed report.
- If a service report template includes a related list, the list populates only if it's on the page layout of the record that the service report is being generated for.
- On the printed report, all related lists are printed. In the mobile preview, a related list must be on the preview page layout to appear on the preview.

7. These elements aren't visible:

- Cases related list
- Headers and footers
- Organization fields

- Certain data fields
- Images in rich text fields on a service report template or service appointment, work order, or work order line item page layout
- The Digital Signature object **Place** field isn't supported.
- Text fields and rich text area fields are truncated to 255 characters. This limitation occurs on rich text area fields, other types of text fields, standard fields, and custom fields.

Inventory Management

- A maximum of 1,000 products are shown in inventory lists. This number is also the maximum number of inventory items that can be primed.
- If you use the setting Enable Multiple Locations, the **Add** and **Add All** buttons are removed from the Product Required screen. Instead, manually add the required products for multiple inventory locations.

Chatter

- The Feed tab in iOS is supported only on the work order, work order line item, service appointment, and case objects. The Feed tab in Android is supported only on the work order, work order line item, and service appointment objects.
- (Android only) Chatter is disabled when the app doesn't have internet connectivity.

Knowledge

- (iOS only) If images in knowledge articles are hosted outside of Salesforce, they display normally. Images uploaded to Salesforce don't load correctly. As an alternative, we recommend using an image link that users can open themselves. You can avoid this issue by selecting the option to use POST requests for cross-domain sessions. Find this option on the **Session Settings** page in Setup.
- (iOS only) Some knowledge articles of supported UI languages, such as Japanese, don't show in the Field Service mobile app.
- (Android only) The Knowledge Article action isn't visible on the Work Order Overview screen. View knowledge articles using the related list.

- (Android only) In Knowledge Article details, field names such as Title don't display on Android devices.
- If a device is running in any of Salesforce's 26 supported languages, the app automatically translates knowledge articles to that language. If the device isn't running in a supported language, no knowledge articles are shown.

Barcode Scanner

- The barcode scanner is compatible only with 1D serial barcodes and 2D QR codes.
- In barcode scanning, the Android app uses the Zebra Crossing library. The iOS app uses Apple's built-in libraries, except for AVMetadataObjectTypeFace. For details, see AVMetadataObjectType.

Briefcase Builder

- Related lists don't work offline when the records are primed with Briefcase Builder.
- The Product object in the Inventory tab isn't supported.

Flows

- Select formula functions and operators are available offline.
- Backslash \ characters and single quotes (') aren't supported in flow formula fields.
- Screen flows launched using quick actions or app extensions aren't supported.
- Email alert actions aren't supported in flows.
- Paused flows aren't supported. If you leave a flow in progress and/or exit the Field Service mobile app, you can't resume the flow where you left off.
- The wasSelected operator can't be used in decisions.
- For eval() functions, the boolean values true and false are case-sensitive.
- Hardcoded ID values must be 18 digits long. 15-digit values aren't supported.
- Apex classes aren't supported in flows.
- Fault connectors aren't supported in flows.
- When uploading images in flows, the content document IDs parameter isn't supported. Admins must create a flow confirmation screen to show which images were uploaded. Videos and other file types aren't supported.

- Flows don’t support the following elements. Adding these elements to a flow causes it to display an error.
  - Lightning components (except for file upload)
  - Output parameters on file upload except for “Uploaded File Names”
  - Choice resources with Display Text Input enabled
  - Actions with output parameters

* * *