ATTENDANCE MANAGER

JIS COLLEGE OF ENGINEERING
Kalyani

ANDROID APPLICATION DEVELOPMENT

Webskitters Technology Solutions Private Ltd.

Session: 2018 –2019

A Industrial Training Report Submitted To

JIS College of Engineering

Bachelor of Technology

In

Computer Science and Engineering

Submitted To:

Mr Swarup Kumar Saha
(Senior Technical Analyst)

Submitted By:

Swarnava Dutta
Tiyasha Giri
Amisha Singh
Sneha Swati
Arghya Bhowmick

Certification

This is to certify that the Industrial Training Report entitled "*Android Application Development*" submitted by ***SWARNAVA DUTTA, TIYASHA GIRI, AMISHA SINGH, SNEHA SWATI, ARGHYA BHOWMICK*** *of* 3rd year BTECH in the year 2018-2019 of Computer Science & Engineering Department of this Institute is a satisfactory account of his industrial training work based on syllabus which is approved for the award of degree of Bachelor of Technology.

....

Date   Signature

ACKNOWLEDGEMENT

I would never have been able to finish my dissertation without the guidance of my teachers and support from my parents.

I would like to express my deepest gratitude to my project guider Mr Swarup Kumar Saha (Senior Technical Analyst) for his excellent guidance, caring, patience and providing me with an excellence atmosphere for completing my project.

I would not forget to show my gratitude to my parents, friends and last but not least to God All Mighty for His blessings on me.

CONTENTS

➢ WHAT IS ANDROID?

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets .

➢ HISTORY:

- ✓ Andy Rubin founded Android Incorporation in Palo Alto, California on October, 2003.
- ✓ Google acquired Android on 17th August 2005. Since then it is a subsidiary of Google Incorporation.
- ✓ Google formed Open Handset alliance (OHA) on 5th November 2007.

➢ VERSION :

Sl No.	Version	Year of foundation	API Level
1	Android 1.0	September 2008	1
2	Android 1.5 Cupcake	April 2009	2
3	Android 1.6 Donut	September 2009	3
4	Android 2.0-2.1 Éclair	October 2009	4
5	Android 2.2 Froyo	May 2010	5-7
6	Android 2.3 Gingerbread	February 2011	8
7	Android 4.0 Ice Cream Sandwich	October 2011	9-10
8	Android 4.1-4.3 Jelly bean	July 2012	11-13
9	Android Kitkat 4.4	October 2013	14-15
10	5.0-5.1 Lollipop	November 2014	16-18
11	Android 6.0 Marshmallow	October 2015	19-20
12	Android 7.0-7.1 Nougat	August 2016	21-22
13	Android 8.0-8.1 Oreo	August 2017	23
14	Android 9 P	2018	24-25

Android Cupcake

Android Donut

Android   Eclairs

Android Foroyo

GingerBread

Ice Cream Sandwich

Jellybean

Kitkat

Lollipop

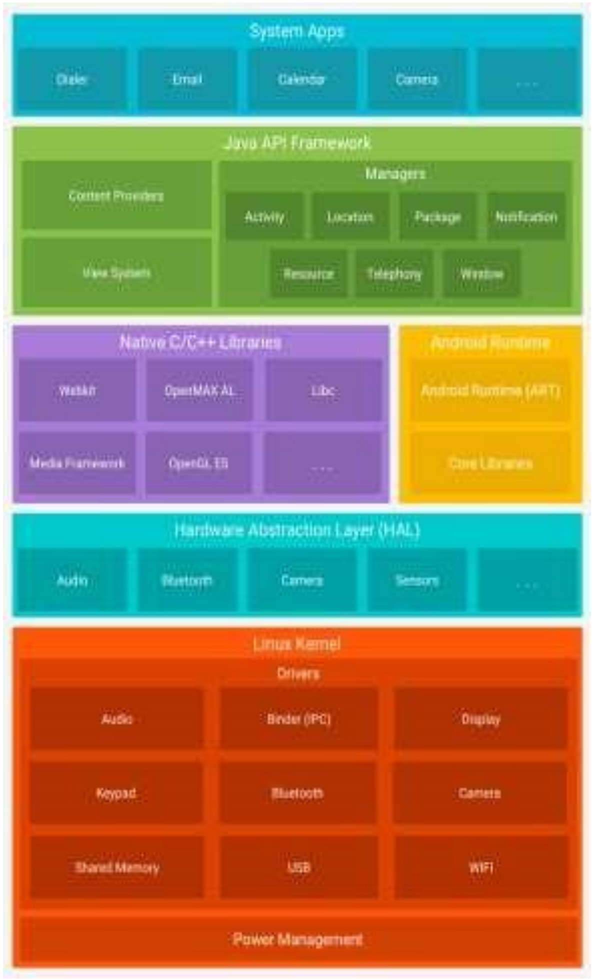
Marshmallow

Naugat

Oreo

➢ <u>ANDROID ARCHITECHTURE :</u>

It is an open source, Linux-based software stack created for a wide array of devices and form factors. The diagram shows the major components of the Android platform.



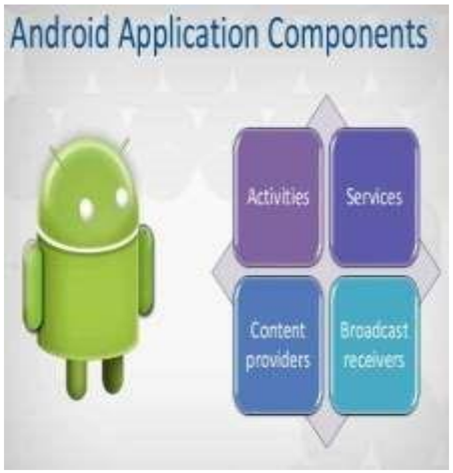
- **<u>Linux Kernal</u>**: Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.
- **<u>Hardware Abstraction Layer</u>**: The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.
- **<u>Android Runtime</u>**: For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a byte code format designed especially for Android that's optimized for minimal memory footprint.

- **Native C/C++ Libraries**: Many core Android system components and services, such as ART and HAL, are built from native code that requires native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps.
- **Java API Framework**: The entire feature-set of the Android OS is available to you through APIs written in the Java language.
  - I) A view system that provides the user interface (UI) including TextView, ListView, EditText etc. The user can see the interface.
  - II) A Resource Manager that provides accessing of the non-code resource such as localized string, graphics, layout files etc.
  - III) A Notification Manager that enables all app to display custom alerts in the status bar.
  - IV) An Activity manager that manages the life cycle of the app.

➢ ANDROID APPLICATION COMPONENTS:

1. Activity :

It is a predefined class present in every android application which will be responsible for creating window. It is basically a class comprising of user interface in regard to which some event need to be fired. An Android application can contain several activities means many different screens that can interact with each other.



- I) UI part will be handled by a markup language like XML(Extended Markup Language)
- II) Code part will be handled by OOPS like Java, Kotlin.

Activity has few other subclasses as follows:

I) AppcompactActivity

II) FragmentActivity

III)BasefragmentActivity API16

IV) BasefragmentActivity API14

V) SupportActivity

VI) Activity

VII) ContextThemeWrapper

VIII) ContextWrapper

IX) Context

✓ **Activity Life cycle :**

Android activity has its own life cycle during its life in Android App. Activities are stored and managed in a stack called activity stack.

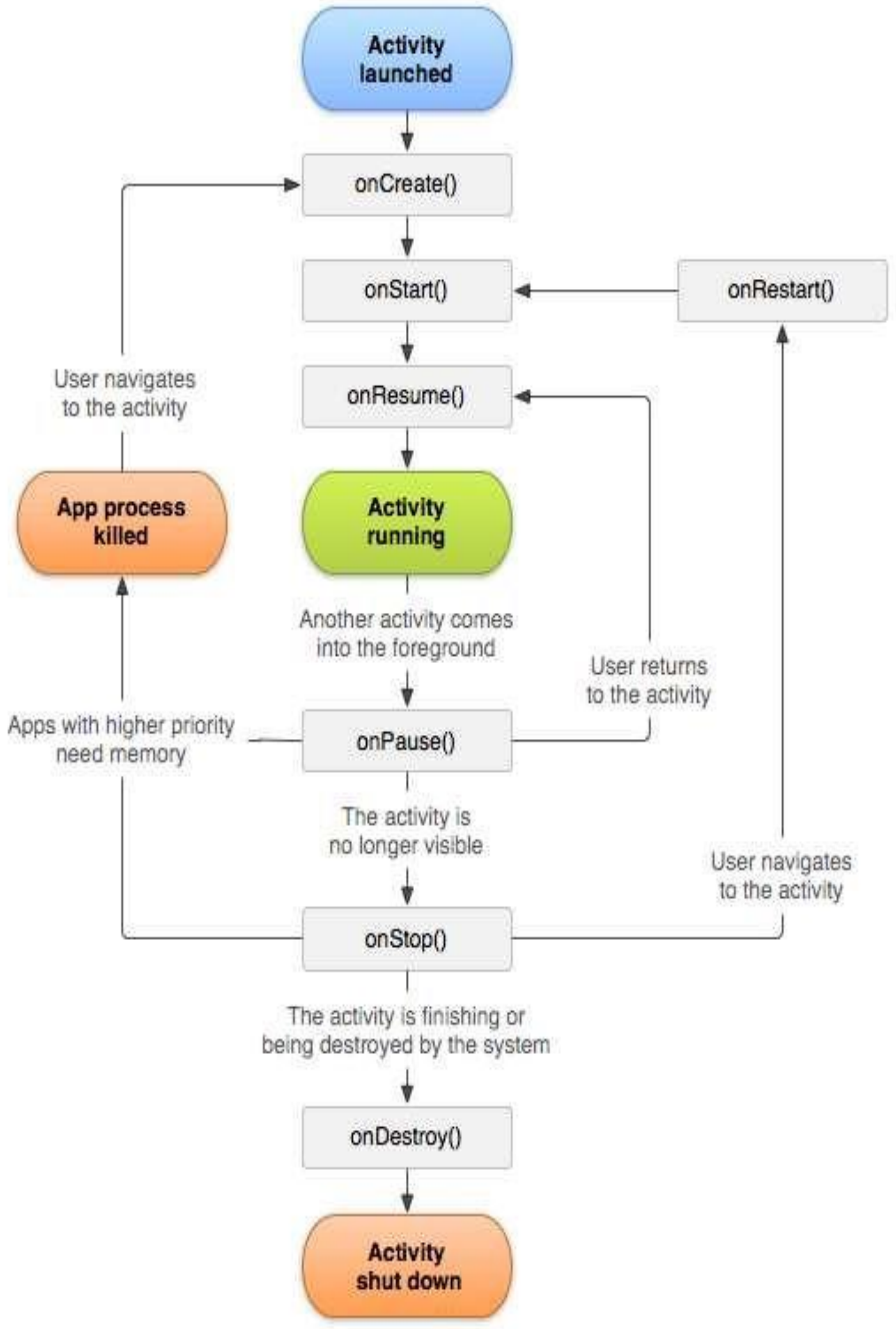
The 7 lifecycle method of Activity describes how activity will behave at different states.

Method	Description
onCreate()	Called when an Activity is first created.
onStart()	Called when an Activity is becoming visible to the user.
onResume()	Called when an Activity will start interacting with the user
onPause()	Called when an Activity is not visible to the user
onStop()	Called when an Activity is no longer visible to the user.
onRestart()	Called when an Activity is stopped, prior to restart
onDestroy()	Called when an Activity is destroyed.

Along these 7 methods there are two optional methods as follows:

I) onnRestoreSaveInstance()

II) onSaveInstance()


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

2. **Broadcast Receivers :** It simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. A broadcast receiver is implemented as a subclass of *BroadcastReceiver* class and each message is broadcaster as an Intent object.

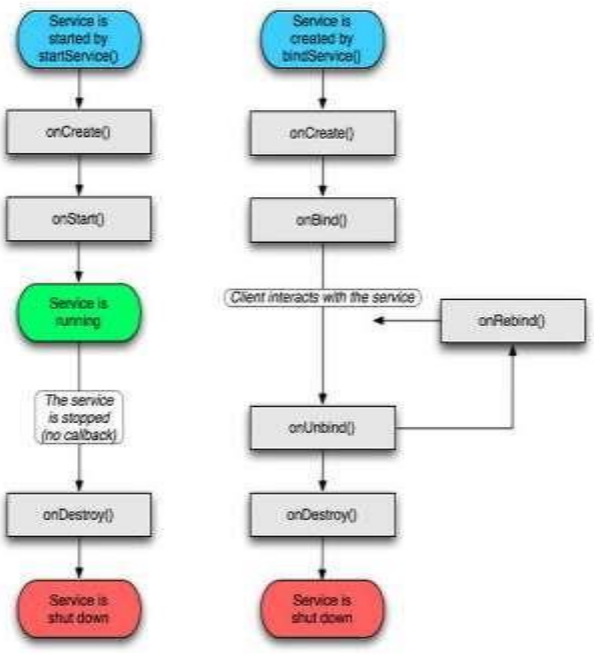
```
public class MyReceiver  extends  BroadcastReceiver
  {    public void onReceive(context,intent){} }
```

3. **Content Provider:** A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver*class. The data may be stored in the file system, the database or somewhere else entirely. A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends  ContentProvider {

                public void onCreate(){}

                            }
```

4. **Services:** A service is a component which runs in the background without direct interaction with the user. As the service has no user interface, it is not bound to the lifecycle of an activity. Services are used for repetitive and potentially long running operations, i.e., Intent downloads, checking for new data, data processing, updating content providers and the like. A service life cycle has call back methods.

I) onStartCommand() :The system calls this method when another component, such as an activity, requests that the service be started, by calling *startService()*. If you implement this method, it is your responsibility to stop the service when its work is done, by calling *stopSelf()* or *stopService()* methods.


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

II) onBind() :The system calls this method when another component wants to bind with the service by calling *bindService()*. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return *null*.

III) onUnbind() :The system calls this method when all clients have disconnected from a particular interface published by the service.

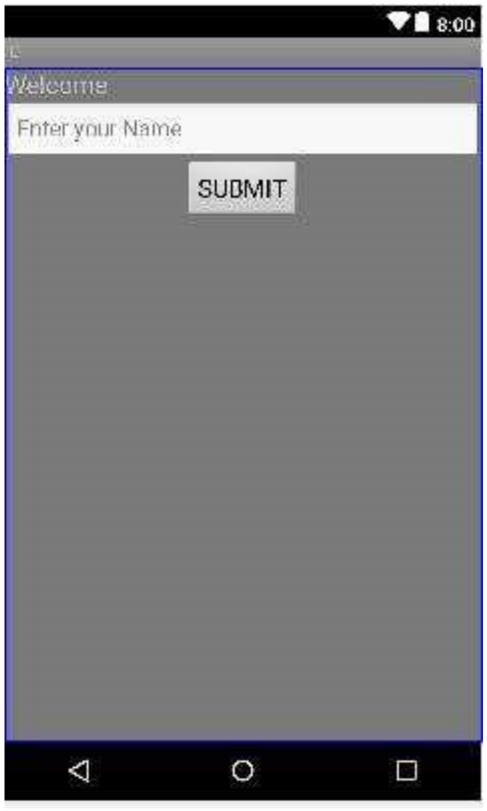
IV) onRebind() :The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*.

V) onCreate() :The system calls this method when the service is first created using *onStartCommand()* or *onBind()*. This call is required to perform one-time set-up.

VI) onDestroy() : The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.

➢ VIEW:

The UI consists of a hierarchy of objects called views i .e. every element of the screen is a view. View objects are used specifically for drawing content onto the screen of an Android device. While you can instantiate a in your Java code, the easiest way to use them is through an XML layout file. An example of this can be seen when you create a simple "Hello World" application in Android Studio. The layout file is the one named *activity_main.xml*,looking something like this.



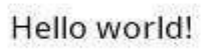
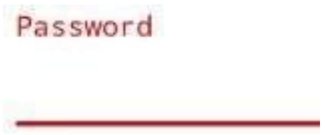
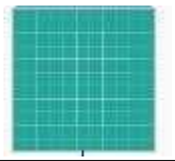
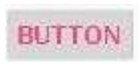
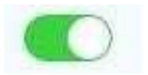
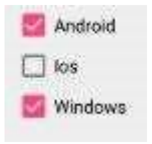
```
      <TextView
android:id="@+id/hello_world"

android:layout_width="wrap_content"
      android:layout_height="wrap_content"
            android:text="Hello World!" />
```

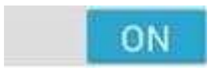
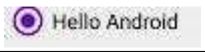
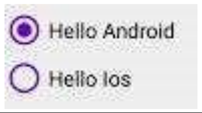
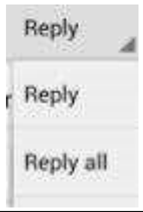
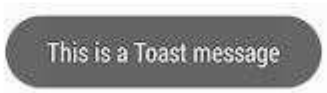
The above example shows a type of View that will be displayed on the screen. The layout_width and layout_height attributes state that View should only take up as much room on the screen as needed to display the text "Hello World!". The id attribute is how that View can be referenced in your Java code like this:

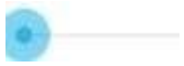
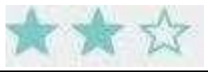
```
      setContentView(R.layout.activity_main);
      TextView textView = (TextView) findViewById(R.id.hello_world);
```

While you can set attributes for a View in XML you can also change attributes in your Java code, such as changing the text of the above TextView.

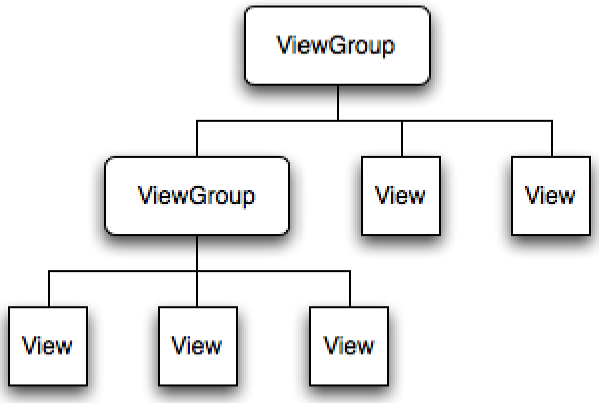
- **WIDGETS :** Widgets are pre-built control through which user interacts with the application. Android contains the following commonly used View subclasses :

	Text View: It is used to display text on screen.
	Edit Text: It is used to get text input from the user
	Image View: It is used for an image display.
	Button view: A button is used to perform event handling by clicking.
	Toggle button: It has two states ON/OFF used for toggling response.
	Check Box: A check box is a simple widget to get user response on specific options/events.

	Switch: It’s like a simple switch you can place in your UI according to your need for event handling.
	Radio Button: Radio Button is just another form of event handling buttons with a radio interface.
	Radio Group: Radio Group is a set of Radio Button’s to get specific response from different options given by the developer.
	Spinner: Spinner displays different options in form of a list from which one can be selected as a response.
	Toast: Displays some information on the screen for some time.
	Progress Bar: It displays progress state for a task performed by the user.

	Seek bar: It has a dragable thumbs that you can touch and drag left or right to set the current progress level.
	Rating Bar: It is used to get a rating response/feedback from the user.

➢ VIEWGROUP:

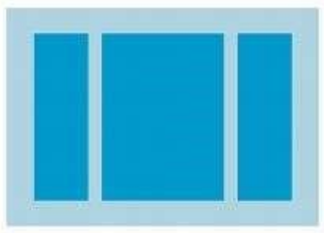


View can be grouped together inside a view group(ViewGroup), which acts as a container of views. A ViewGroup is an invisible object used to contain other View and ViewGroup objects in order to organize and control the layout of a screen. The view group is the base class for layouts and views containers. The relationship is parent-child, in which the parent is a view group within the group. The following are common view group:

- Linear Layout
- Relative Layout
- ListView
- GridView

- **LAYOUTS:** The views for a screen are organized in a hierarchy. At the root of this hierarchy is a ViewGroup that contains the layout of the entire screen.

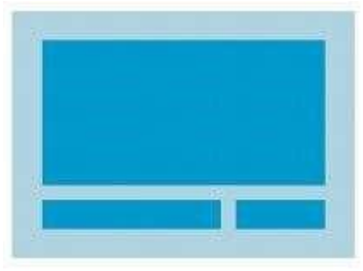
   Linear Layout: Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally. Layout direction is specified by android:orientation attribute.



```
<?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="match_parent"
android:orientation="vertical">
<TextView
android:id="@+id/tv"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="20dp"         android:text="Welcome"
/>
<EditText
android:id="@+id/et"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Enter your Name" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:textSize="20dp"
android:text="SUBMIT"/></LinearLayout>
```

- **Relative Layout :** Relative Layout lays out elements based on their relationships with one another, and with the parent container.

  a) `android:layout_alignParentBottom="true"` : Places the bottom of the element on the bottom of the container



b) `android:layout_alignParentLeft="true"` : Places the left of the element on the left side of the container

c) `android:layout_alignParentRight="true"`: Places the right of the element on the right side of the container

d) `android:layout_alignParentTop="true"` :Places the element at the top of the top of the container

e) `android:layout_centerHorizontal="true"` : Centers the element horizontally within its parent container.

f) `android:layout_centerInParent="true"` : Centers the element both horizontally and vertically its parent container.

g) `android:layout_centerVertical="true"` : Centers the element both vertically within its parent container.

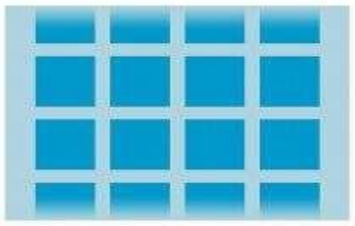
h) `android:layout_above` : Places the element above the specified element.

i) `android:layout_below:` Places the element below the specified element

j) `android:layout_toLeftOf` : Places the element left to the specified element

k) `android:layout_toRightOf:` Places the element right to the specified element

- **Constraint Layout:** A group of child views to control how views are positioned relative to other elements in the layout. Constraint Layout was designed to make it the layout editor.
- **Grid Layout:** Layout places its child screens in a rectangular grid that can be scrolled.



- **VALUE RESOURCE FILES:** Keeping values such as strings, colors in separate resource files makes it easier to manage them, especially if you use them more than once in your layouts.
  1. **Strings:** The strings.xml file is used to store the string values to be used in your app.
  2. **Colors:** The strings.xml file is used to define and store the color values to be used in the app.
  3. **Dimensions:** The dimens.xml file is used to store the dimension values to be used in the app for padding, margins, size etc for different views.
  4. **Styles:** A style is a resource that specifies common attributes such as height, padding, font color, font size, background color. Styles are meant for attributes that modify the look of the view.

- **INTENT :**

  Intents are asynchronous message which allow application components to request functionality from other Android components. Intents allow user to interact with components from the same application as well as with components contributed by other applications. It's like a message that Android listens for and react accordingly by identifying and invoking the targeted app's appropriate component (like an Activity, Service, Content Provider). To start an activity, we use the method startActiviity(intent). Intents are also used to start the services via intents, use the startService(Intent) method. It can be used for:
  - Starting an Activity
  - Starting a service
  - Delivering a broadcast

Android supports explicit and implicit intents. An application can define the target component directly in the intent (*explicit intent*) or ask the Android system to evaluate registered components based on the intent data(*implicit intents*).

I) Explicit intents explicitly define the component which should be called by the Android system, by using the Java class as identifier. Explicit intents are typically used within an application as the classes in an application are controlled by the application developer. The following shows how to create an explicit intent and send it to the Android system to start an activity.

We can send data from one Activity to another Activity by adding it to our intent in key-value pairs that can be retrieved by the receiving Activity.

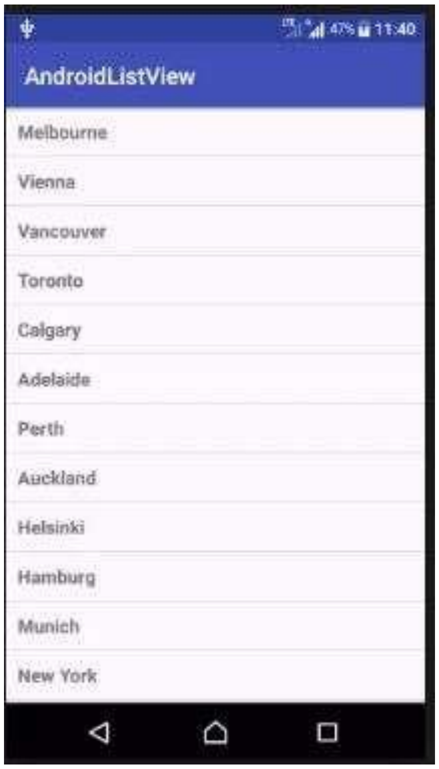
```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
Bundle(extras !=null){
String Value1=extras.getString(“Value1”);
int Value2=extras.getString(“Value2”);
}
```

Android Bundle is used to pass data between activities. The values that are to be passed are mapped to String keys which are later used in the next activity to retrieve the values. We just transferred two pieces of data. The key has to be type of String. Calling *getExtras()* on the intent object gives us a bundle object on which we can call various methods like *getString()*, *getInt()*, *gteChar().* These all are doing with Bundle.

II) Implicit intents specify the action which should be performed and optionally data which provides content for the action. If an implicit intent is sent to the Android system, it searches for all components which are registered for the specific action and the fitting data type. If only one component is found, Android starts this component directly.

```
Intent i=new Intent
(Intent.ACTION_VIEW,Uri.parse("http://www.google.com"));
```

➢ ANDROID CONTAINERS :



1. **ListView** **:** Android ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

   - The source of list item can be anything ranging from simple Array to database queries.
   - The Adapter act as a bridge between the listview and the source. Thus the Adapter provides access to data items by fetching the data from the source.
   - The adapter is also responsible for making a View for each item in data set.

✓ *LIST VIEW ATTRIBUTES*

Attributes	Description
`android:id`	This is the ID which uniquely identifies the layout.
`android:divider`	This is drawable or color to draw between list items.

`android:dividerHeight`	This specifies height of the divider. This could be in px, dp, sp, in, or mm.
`android:entries`	Specifies the reference to an array resource that will populate the ListView.
`android:footerDividersEnabled`	When set to false, the ListView will not draw the divider before each footer view. The default value is true.
`android:headerDividersEnabled`	When set to false, the ListView will not draw the divider after each header view. The default value is true.

Steps of implementing ListView lists to display lists in Android using ArrayAdapter

I) Define the Array data source

```
String[] days={"Sunday", "Monday", "Tuasday"}
```

II) Define the adapter and provide the layout of a single view to the adapter.

```
ArrayAdapter adapter=new ArrayAdapter
(this,android.R.layout.simple_list_item_1,days)
```

III) Set the adapter to the listview

```
ListView lv=(ListView)findViewById(R.id.idListView)
```

IV) Attach an onItemClickListener to define what action should taKe place on click of the list item.

```
lv.setOnItemClickListener(new
AdapterView.OnItemClickListener())
     {
```

```
public  void  OnItemClickListener(AdapterView  adapterView,
View
view, int I, long l)
   {

         }
   }
   )
   ;
```

Steps of implementing ListView using BaseAdapter:

I) Define the Array data source

```
String[] days={"Sunday", "Monday", "Tuasday"}
```

II) Define the adapter and provide the layout of a single view to the adapter.

III) The Class BaseAdapter is an abstract class that extends four methods.

-
  ```
  @Override public int getCount()
  {     return names.length;
  }
  ```

-
  ```
  @Override     public     Object
  getItem(int pos)  {       return
  pos;
  }
  ```

-
  ```
  @Override       public       long
  getItemId(int pos) {      return
  pos;
  }
  ```

-
  ```
  @Override      public      View
  getView(final int position, View
  convertView,
  ViewGroup parent){
  }
  ```

2. GridView :

Android GridView shows item in two dimensional scrolling (rows and columns) grid and grid items are not necessarily predetermined but they inserted automatically in the layout by using ListAdapter.
The ListView and GridView are subclasses of AdapterView and they can be populated by binding them to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.

How to create your own Android application using GridView?

a) You will use Android studio IDE to create an Android application and name it as *HelloWorld* under a package *com .example. helloworld* .
b) Modify the detault content of *res/layout/activity_main.xml* file to include GridView content with the self explanatory attributes.
c) No need to change string.xml, Android studio takes care of defaults strings which are placed at string.xml
d) Let's put few pictures in *res/drawable-hdpi* folder. Like sample0.jpg, sample1.jpg, sample2.jpg, sample3.jpg, sample4.jpg, sample5.jpg, sample6.jpg and sample7.jpg.
e) Create a new class called **ImageAdapter** under a package com.example.helloworld that extends BaseAdapter. This class will implement functionality of an adapter to be used to fill the view.

➢ TOOLBAR:

Toolbar was introduced in Android Lollipop, API 21 release and is the spiritual successor of the ActionBar. It's a ViewGroup that can be placed anywhere in your XML layouts. Toolbar's appearance and behavior can be more easily customized than the ActionBar. In AppCompat, Toolbar is implemented *in android:support.v7.widget.Toolbar* class. There are two ways to implement Toolbar :

I) Use a Toolbar as an Action Bar when you want to use the existing ActionBar facilities (such as menu inflation and selection.

II) Use a standalone Toolbar when you want to use the pattern in your app for situations that an Action Bar would not support; for example, showing multiple toolbars on the screen, spanning only part of the width, and so on.

Using Toolbar as ActionBar:

I) first ensure the *AppCompat-v7 support* library is added to your application *build.gradle (Module:app) file*:

```
dependencies {
………..                                  implementation
'com.android.support:design:28.0.0alpha3'
}
```

II) Second, let's disable the theme-provided ActionBar. The easiest way is to have your theme extend from *Theme.AppCompat.NoActionBar (or the light variant) within the res/values/styles.xml file*:

```
<resources>
 <!-- Base application theme. -->   <style
 name="AppTheme"
 parent="Theme.AppCompat.Light.NoActionBar">
 </style>
 </resources>
```

III) Now you need to add a Toolbar to your Activity layout file. One of the biggest advantages of using the Toolbar widget is that you can place the view anywhere within your layout. Below we place the toolbar at the top of a LinearLayout like the standard ActionBar:

```
<LinearLayout
```

```
      xmlns:android=http://schemas.android.com/apk/res/android
      xmlns:app="http://schemas.android.com/apk/res-auto"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:fitsSystemWindows="true"
      android:orientation="vertical">
      <android.support.v7.widget.Toolbar android:id="@+id/toolbar"
            android:minHeight="?attr/actionBarSize"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:titleTextColor="@android:color/white"
            android:background="?attr/colorPrimary">
      </android.support.v7.widget.Toolbar>
</LinearLayout>
```

IV) When using the support library, make sure that you are importing *android.support.v7.widget*.Toolbar and not android.widget.Toolbar.

V) Find the toolbar view inside the activity layout

```
Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
```

Sets the Toolbar to act as the ActionBar for this Activity window. Make

sure the toolbar exists in the activity and is not null

```
setSupportActionBar(toolbar);
```

- **MENUS:**

Menus are a common user interface component in many types of applications. To provide a familiar and consistent user experience, you should use the MENU APIs to present user actions and other options in your activities.

- Beginning with Android 3.0 (API level 11), Android-powered devices are no longer required to provide a dedicated Menu button. With this change, Android apps should migrate away from a dependence on the traditional 6-item menu panel and instead provide an app bar to present common user actions.

- Although the design and user experience for some menu items have changed, the semantics to define a set of actions and options is still based on the MENU APIs.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/about"
  android:title="About The App"
        android:icon="@drawable/about_the_app"
        app:showAsAction="ifRoom" >
</item>
  </menu>
```

JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater menuInflater=getMenuInflater();
menuInflater.inflate(R.menu.drawer_menu,menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
int id=item.getItemId();
switch (id) {
    case R.id.about:
        showDialog();
        break;
}
return super.onOptionsItemSelected(item);
}
```

➢ INFLATER:

Inflater keyword is used in the java class of the corresponding layout(xml) file. This keyword is used in the onCreate() function of the class. Inflater is of two types.

I) **Layout Inflator**: LayoutInflater is a class used to instantiate layout XML file into its corresponding view objects which can be used in java programs.

In simple terms there are two ways to create UI in android. One is static way and another is dynamic or programmatically. Suppose we have a simple layout main.xml having one textview and one edittext as follow.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:id="@+id/layout1"
> <TextView
android:id="@+id/namelabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Enter your name"

android:textAppearance="?android:attr/textAppearanceLarge" >
</TextView>     <EditText
android:id="@+id/name"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_marginTop="14dp"
android:ems="10">
</EditText>
</LinearLayout>
```

We can display this layout in static way by

```
public    void    onCreate(Bundle    savedInstanceState)    {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
```

Dynamic way of creating a view means the view is not mentioned in our main.xml but we want to show with this in run time. For example We have another xml in layout folder as *footer.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/TextView1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="center_horizontal"
android:text="Add your record"     android:textSize="24sp"
>
</TextView>
```

We want to show this textbox in run time within our main UI. So here we will inflate *text.xml* . See how

```
public      void      onCreate(Bundle      savedInstanceState)      {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
final LayoutInflater  inflater =
(LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE);
TextView   t   =   (TextView)inflater.inflate(R.layout.footer,null);
lLayout         =         (LinearLayout)findViewById(R.id.layout1);
lLayout.addView(t);
```

Here I have used *getSystemService (String)* to retrieve a LayoutInflater instance. I can use *getLayoutInflator()* too to inflate instead of using *getSystemService* (String) like below

```
LayoutInflator inflater = getLayoutInflater();
TextView  t  =  (TextView)  inflater.inflate(R.layout.footer,  null);
lLayout.addView(t);
```

We have inflated the view footer using inflater and converted it Textview object. In this example we have inflated a simple textview.

➢ SHARED PREFERENCES :

Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of key, value pair.

- In order to use shared preferences, you have to call a method *getSharedPreferences()* that returns a SharedPreference instance pointing to the file that contains the values of preferences.

```
SharedPreferences                 sharedpreferences                 =
getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);
```

The first parameter is the key and the second parameter is the MODE.

- You can save something in the sharedpreferences by using *SharedPreferences.Editor class*. You will call the edit method of SharedPreference instance and will receive it in an editor object. Its syntax is :

```
Editor    editor    =    sharedpreferences.edit();
editor.putString("key",                  "value");
editor.commit();
```

➢ NAVIGATION DRAWER:

The Android Navigation Drawer is a sliding panel mostly found on the left edge of the Main Screen where you have your app’s main navigation menu or options. To reveal it user swipes a finger from the left edge of the screen or, while at the top level of the App, the user touches the App icon in the action bar. It is known by different names such as Android navigation drawer, Android sliding menu, Android swipe menu among others.

- The drawer must be the top element and the main content(Linear Layout) must be the first child in the DrawerLayout.
- The main content view must match the parent view's width and height,since it represents the full screen contents when the drawer is closed.

How to apply Navigation Drawer?

a) first ensure the *android.support.v4.widget.DrawerLayout support* library is added to your application *build.gradle (Module:app) file*:

```
dependencies {
………..
implementation'com.android.support:design:28.0.0-
alpha3'
}
```

b) We need to add the *NavigationView* in Main Layout XML file

```
<android.support.design.widget.NavigationView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/navigation_view"
android:layout_gravity="start"
app:headerLayout="@layout/header_layout"
app:menu="@menu/menu_layout">
</android.support.design.widget.NavigationView>
```

c) Navigation Drawer always has a header part separately. So a separate XML file is needed to be created.

d) Navigation Drawer is always set in the toolbar with Hamberger Icon so to do that we need the code as follows :

```
toolbar=(Toolbar)findViewById(R.id.toolBar);
setSupportActionBar(toolbar);
drawerLayout=(DrawerLayout)
findViewById(R.id.drawerLayout);
```

e) To implement Hamburg Icon

```
ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(this, drawerLayout, toolbar,
R.string.open, R.string.close);
```

f) To get response from the Hamberger Icon

g) To verify whether the drawer is open or not

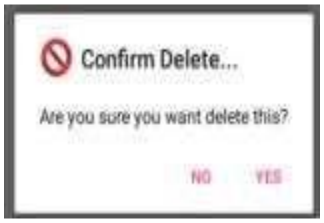
```
drawerLayout.addDrawerListener(toggle);
toggle.syncState();
```

h) We also need to modify the *Strings.xml* file as :

```
<resources>
<string name="app_name">Navigation</string>
<string name="open">open</string>
<string name="close">close</string>
</resources>
```

➢ ALERT DIALOGUE :

A Dialog is small window that prompts the user to a decision or enter additional information. Some times in your application, if you wanted to ask the user about taking a decision between yes or no in response of any particular action taken by the user, by remaining in the same activity and without changing the screen, you can use Alert Dialog.



Are you sure you want delete this?

1) In order to make an alert dialog, you need to make an object of *AlertDialogBuilder* which an inner class of AlertDialog. Its syntax is given below:

```
AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
```

2) Now you have to set the positive (yes) or negative (no) button using the object of the *AlertDialogBuilder class*. Its syntax is

```
alertDialogBuilder.setPositiveButton(CharSequence  text,
DialogInterface.OnClickListener listener)
alertDialogBuilder.setNegativeButton(CharSequence text,
DialogInterface.OnClickListener listener)
```

3) After creating and setting the dialog builder, you will create an alert dialog by calling the *create()* method of the builder class. Its syntax is :

```
AlertDialog  alertDialog  =  alertDialogBuilder.create();
alertDialog.show();
```

➢ ANDROID SQLITE *:*

SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. It is embedded in android by default. So, there is no need to perform any database setup or administration task.

Here, we are going to see the example of sqlite to store and fetch the data. Data is displayed in the logcat. SQLiteOpenHelper class provides the functionality to use the SQLite database.

- **SQLiteOpenHelper class :**

The *android.database.sqlite.SQLiteOpenHelper* class is used for database creation and version management. For performing any database operation, you have to provide the implementation of *onCreate()* and *onUpgrade()* methods of *SQLiteOpenHelper class*.

✓ ***Constructors of SQLiteOpenHelper class***

There are two constructors of SQLiteOpenHelper class.

Constructor	Description
`SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)`	creates an object for creating, opening and managing the database.
`SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)`	creates an object for creating, opening and managing the database. It specifies the error handler.

✓ ***Methods of SQLiteOpenHelper class***

Method	Description
`public abstract void onCreate(SQLiteDatabase db)`	called only once when database is created for the first time.
`public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`	called when database needs to be upgraded.

- **SQLiteDatabase class**

It contains methods to be performed on sqlite database such as create, update, delete, select etc.

✓ ***Methods of SQLiteDatabase class***

There are many methods in *SQLiteDatabase class*. Some of them are as follows:

Method	*Description*
`void execSQL(String sql)`	executes the sql query not select query.
`long insert(String table, String nullColumnHack, ContentValues values)`	inserts a record on the database. The table specifies the table name, *nullColumnHack* doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
`int update(String table, ContentValues values, String whereClause, String[] whereArgs)`	updates a row.
`Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)`	returns a cursor over the resultset.

✓ ***Steps of creating SQLite Database and Table***

1) Create a new java class called *DataBaseHelper.java*

2) Extend the class with *SQLiteOpenHelper*

```
public DataBaseHelper extends SQLiteOpenHelper
{
}
```

3) Implements the methods:

```
public DataBaseHelper extends SQLiteOpenHelper
{
@overide
public void onCreate(SQliteDatabase db)
{
}
@overide
public void onUpgrade(SQliteDatabase db,int oldversion,int newversion)
{
}
}
```

4) Create the default constructor

```
public DataBaseHelper extends SQLiteOpenHelper
{
public DataBaseHelper(Context context)
{
Super(context, name, factory, version);
}
@overide
public void onCreate(SQliteDatabase db)
{
}
@overide
public void onUpgrade(SQliteDatabase db,int oldversion,int newversion)
{
}
}
```

5) Define name of the database and table.

```
public  static  final  String  DATABASE_NAME=”Student.db”;
public static final String TABLE_NAME=”Student_table”;
```

6) Define name of the columns.

```
public  static  final  String  COL_1=”id”;
public      static      final      String
COL_2=”name”;  public  static  final  String
COL_3=”marks”;
```

7) In *onCreate()* method write a Query to create a Table.

```
public void onCreate(SQliteDatabase db)
{      db.execSQL("CREATE"   +   DATABASE_NAME   +   "(ID
INTEGER
PRIMARY KEY AUTOINCREMENT, NAME TEXT, MARKS INTEGER)");
}
```

8) In *onUpgrade()* method write a Query to upgrade a Table.

```
public      void      onUpgrade(SQliteDatabase      db,int
oldversion,int newversion)   {
db.execSQL("DROP TABLE IF EXISTS " +TABLE_NAME);
}
```

9) To create database use getWritableDatabase() method in constructor.

```
public DataBaseHelper(Context context)
{         super(context,  name,  factory,  version);
   SQLiteDatabase db=this.getWritableDatabase();
}
```

10) Create an object of DataBaseHelper class in MainActivity and pass Context to it.

```
DataBaseHelper myDb=new DataBaseHelper(this);
```

✓ ***Steps of reading data from SQLite***

1) In *DataBaseHelper* class create a method to read data with return type as Cursor.

```
public Cursor getAllData()
{
}
```

2) Open database for reading data in the table using *getWritableDatabase()*

```
public Cursor getAllData()
{
SQLiteDatabase db=this.getWritableDatabase();

}
```

3) Write your query for reading values in the *rawQuery()* method. It runs the provided SQL and returns a Cursor over the result set.

```
Cursor res=db.rawQuery("Select * from " + TABLE_NAME,null); return
res;
```

4) Now go to *MainActivity.class* call the method for reading data from

```
DataBaseHelper.class DataBaseHelper
myDb; myDb.getAllData();
```

Since return type is *Cursor*. We need to store the data in an object of Class *Cursor*.

```
Cursor res-=myDb.getAllData();
```

Now we can get result by moving cursor to the next row and appending the result in *StringBuffer*.
To get the result from the resultset we pass the parameter *ColumnIndex* in *getString()* method.

```
while(res.moveToNext())
{
```

```
stringBuffer.append("Id: " +res.getString(0)+"\n")
}
```

✓ ***Steps of updating data of a table created in SQLite***

1) In DataBaseHelper Class create a method with return type as Boolean to update data in database.

   ```
   public Boolean updateData()
   {
   }
   ```

2) Pass all arguments which need to be updated in the database as parameters to the function.

   ```
   public Boolean updateData(String id, String name,
   String marks)
   {
   }
   ```

3) Open database for updating the data in the table using getWritableDatabase().

   ```
   SQLiteDatabase db=this.getWritableDatabase();
   ```

4) Create an instance of a class *ContentValues* and put method to store values in the object

   ```
   ContentValues   contentValues=new   Contentvalues();
   contentValues.put(COL_2,name);
   contentValues.put(COL_3,marks);
   ```

5) Update data in table using update() method on the SQliteDatabase istance.

   ```
   int update(String table, ContentValues  contentValues,
   String whereClause, String[] whereArgs)
   ```

- **<u>VOLLEY:</u>**

Android volley is a networking library was introduced to make networking calls much easier, faster without writing tons of code. By default all the volley network calls works asynchronously, so we don't have to worry about using a sync task anymore.

- Network Requests in a Volley are added to a RequestQueue Instead of creating a new instance of RequestQueue every time, we follow the singleton pattern by creating a single instance of RequestQueue throughout our application.

- In the application below, we'll be implementing GET and POST StringQueue and JsonObjectRequest We'll use Json library to parse the response and display the results in a RecyclerView.

```
        stringRequest=new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
        showMsg("Server Response",response);
        }},
        new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
        showMsg("Server Response","Page not reachable");
        }}){
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String,String> params=new HashMap<String,String>();
            params.put("uname",s1);
            params.put("pwd",s2);
            return params;
}};
 requestQueue.add(stringRequest);
}};
```

PROJECT DISCUSSION

❑ INTRODUCTION & OBJECTIVES OF THE PROJECT

INTRODUCTION:

Attendance manager is an app which reduces the efforts of the teachers of taking attendance by pen and paper i.e. attendance register .this app maintains an offline data base which stores the roll no and name of the students which can be accessed by the teacher whenever he/she wants it until the database is cleared manually by the teacher.

OBJECTIVE:

As the world is becoming digital day by day, this app is a small step towards making India more digital and make my country to be recognized as a digital nation across the glob. The current scenario of various educational institutions to manage and maintaining student information is very tough task for any one. The whole academic record of the student information consist of monitoring their performance and progress changes periodically which is very huge workload on lecturers to handle and keep on 0updating the progress report of each and every student in against of their respective scheduled classes.

To developed and design the android-based mobile attendance application for the management of attendance records in the educational organization.

Sometimes we come across some common issues like someone missing their attendance that occurs while teacher takes attendance in the register. But If the teacher uses this app ,students insert their roll and name themselves from

❑ <u>TOOLS/PLATFORM</u> & HARDWARE-SOFTWARE REQUIREMENT SPECIFICATION

Hardware Requirement -

<u>Web Server</u>

Processor	:	Intel® Core™ i3-380M (3.06 GHz,3MB L2 Cache, 1066MHz FSB)
	:	
Standard Memory		4 GB DDR3 RAM
Network Controller	:	Embedded NC105i Gigabit Adapter
Secondary Storage	:	SATA Hard disk (500GB)
Optical drive type	:	Optional 12.7mm DVD-ROM/DVD-RW Drive

<u>Desktop PCs</u>

Processor	:	Intel® Pentium™ 4, Core2Duo, Dual Core, Quad Core, AMD
Processor cache	:	1MB or 2MB L2 cache memory
Standard Memory	:	1 GB DDR2RAM or above
Network Controller	:	Integrated 10/100 Base T network interface
Secondary Storage	:	Hard disk (500 GB)
Optical drive type	:	Optional 12.7mm DVD-ROM/DVD-RW Drive

Software Requirement

Operating System	Operating System Windows XP/7 or higher version will be used for client platform.
Front-End Tool	Extended Markup Language (XML) will be used as main front-end tools.
Back-End Tool	OOPS Java will be used for coding purpose.
Database	Android SQLite will be used to design the database and tables.
Database Editor	Android SQLite Tools.
Database Connection	Android SQLite open source database
IDE	Android Studio IDE 3.1.0

❑ PROBLEM DEFINATION, REQUIREMENT SPECIFICATION:

Problem Definition -

- ✓ Teacher can register into the app using unique username and password.
- ✓ Teacher can login into the app using the registered username and password.
- ✓ Students can input their roll no and name according to their departments and years
- ✓ User can rate the app as per their satisfaction.
- ✓ User can give a feedback to admin if they feel to give.

Project Planning

The objective of the project planning is to provide a framework that enables the Admin to make reasonable estimates of resources, and schedule. These Estimates have been made within a limited time frame at the beginning of a Software project and were updated regularly as the project progress.
In the "ATTENDANCE MANAGER" app,
Teacher can register into the app using unique username, password and login into the app using them. Then the teacher selects the department and the particular year and then selects the "student activity".
The students enter their respective roll no and name and submit them into the database.
If the teacher wants to view the student list ,he/she have to go to the "teacher activity", and can view the student list. The list remains until the database is cleared manually.

- **Software Scope:-**

It is the first activity of the software project planning. Here i got information About the data and control to be processed, function, performance, constraints, Interface and reliability. "Functions described in the statement of scope are Android application development
Evaluated and defined more detail prior to the beginning of estimation". Because Both cost and schedule estimates are functionally oriented. Performance Considered encompasses processing and response time requirements. Constraints
Identified limits placed on the software by external hardware, available memory, Or other existing system.

Feasibility Study: - A feasibility study is a preliminary study undertaken to Determine and document a project's viability. The term feasibility study is Also used to refer to the resulting document. The results of this study are used To make a decision whether or not to proceed with the project. If it indeed

Leads to a project being approved, it will — before the real work of the Proposed project starts — be used to ascertain the likelihood of the project's success. It is an analysis of possible alternative solutions to a problem and a Recommendation on the best alternative. It, for example, can decide whether An order processing be carried out by a new system more efficiently than the Previous one.

Feasibility study is one of the most important factors to undergo a process of Software development, it is considered as one of the first step of sdlc (i.e. Software development life-cycle). There remain various kinds of feasibility Studies, but most important of them to be required actually for a software Development is discussed below: -

1. Technical feasibility
2. Economic feasibility
3. Operational feasibility
4. Social feasibility

Technical Feasibility: - This is concern with specifying equipment and software That will successfully satisfied the user requirement. The technical needs of the System may vary considerably but might include some feature such as-

The facility to produce output in a given time.

Response time under certain conditions.

Ability to process a certain volume of transudation at a particular Speed.

In our system the technical feasibility is mostly emphasis on the above points Because in the rush hour it will provide the speed and accuracy to the user of the System. It gives the idea to smooth operation of the system and communication Between them.

Economic Feasibility:-It is the most important now a days the economic performance of software that How much cost cutting provided by the software, how much it cost effective. From the point of view of my software it is give more than ninety percent of Cost benefit because it provide stationary less office and account also partly Maintain by my system itself.

Operational Feasibility:-

It is mainly related to human organizational and political aspects. The point to Considered are: what changes will be brought with the system, what Organizational structures are distributed, what new skills are required, do the Existing staff members have these skills? These types of questions are executed in This feasibility. To solve this types of question my system is implemented such a Way that no high skill personal are not required, only the man who know some English language and has data entry knowledge who can drive this system very Effectively and existing staffs are enough for my system.

Social Feasibility: -

Social feasibility is a determination of whether a proposed project will be Acceptable to the people or not. We tested the app on some teachers of cse department, and they were very satisfied.

Students are also very happy with the new attendance system as it is fully digital and user friendly.

PROJECT SCHEDULING :

“Software development project scheduling is an activity that distributes estimated Effort across the planned project duration by allocating the effort to specific Software developing tasks. Therefore generalized project scheduling tools and Techniques can be applied with little modification to this project.”

When creating project schedule, the planner begins with a set of tasks, if Automated tools are used, the work breakdown is input as a task network or task Outline.

Work & Tasks	Planned Start	Planned Complete	Effort Allocation
1) Identify needs & benefits Identify needs & project constraints	***Wk1, d1*** Wk1, d5	Wk1, d4 Wk1, d5	4p-d 1p-d
2) Feasibility study	Wk2, d1	Wk2, d1	1p-d
3) Project Planning	Wk2, d2	Wk2, d2	1p-d
4) Project scheduling	Wk2, d3	Wk2, d3	1p-d
5) System requirement specification	***Wk2, d4***	Wk4, d4	11p-d
6) System design i) Database design ii) User interface design	***Wk4, d5*** Wk7, d1	Wk6, d5 Wk10, d5	11p-d 20p-d
7) Coding	Wk11, d1	Wk16, d5	39p-d
8) Error checking& validation checking	Wk17, d1	Wk18, d1	7p-d
9) Testing	Wk18, d2	Wk20, d1	10p-d
10) Implementation/Launch	Wk20, d2	Wk20, d2	1p-d

Here, the proposed project is divided into the above scheduled:- Where Wk means –WEEK, d means –Days, p-d means-Person per Day.

Gantt Chart -

	Start Date	Duration	End Date
SYSTEM ANALYSIS & DESIGN	01-August	07	08August
CODING & TESTING	08-August	10	15August
IMPLEMENATION	16-August	15	02September

❑ Scope Of The Solution

The users of this application are students and teachers. Students can give their attendance and teacher who can view the attendance. The app can do the following things:

1. There is a login page where the teacher can give his or her username and password and login.
2. The teacher can then select the department.
3. Then the teacher can select the specific year for whom attendance needs to be taken.
4. After that the teacher will give the students to input their data by selecting student activity.
5. Students can enter their roll number and name and click on the submit button. To update the given information the student can click on the update button.
6. Teachers can view the attendance of the students by clicking on teacher activity and then click on view. To clear the list the teacher can click on the clear all.

Analysis (DFDs)

At the technical level, software design begins with a series of modeling tasks that lead to the complete specification of requirements and comprehensive design representations for the software. The analysis model actually a set of functional models of system, is the first technical representation of a software based "attendance manager".

The analysis model must achieve the following three objectives: -

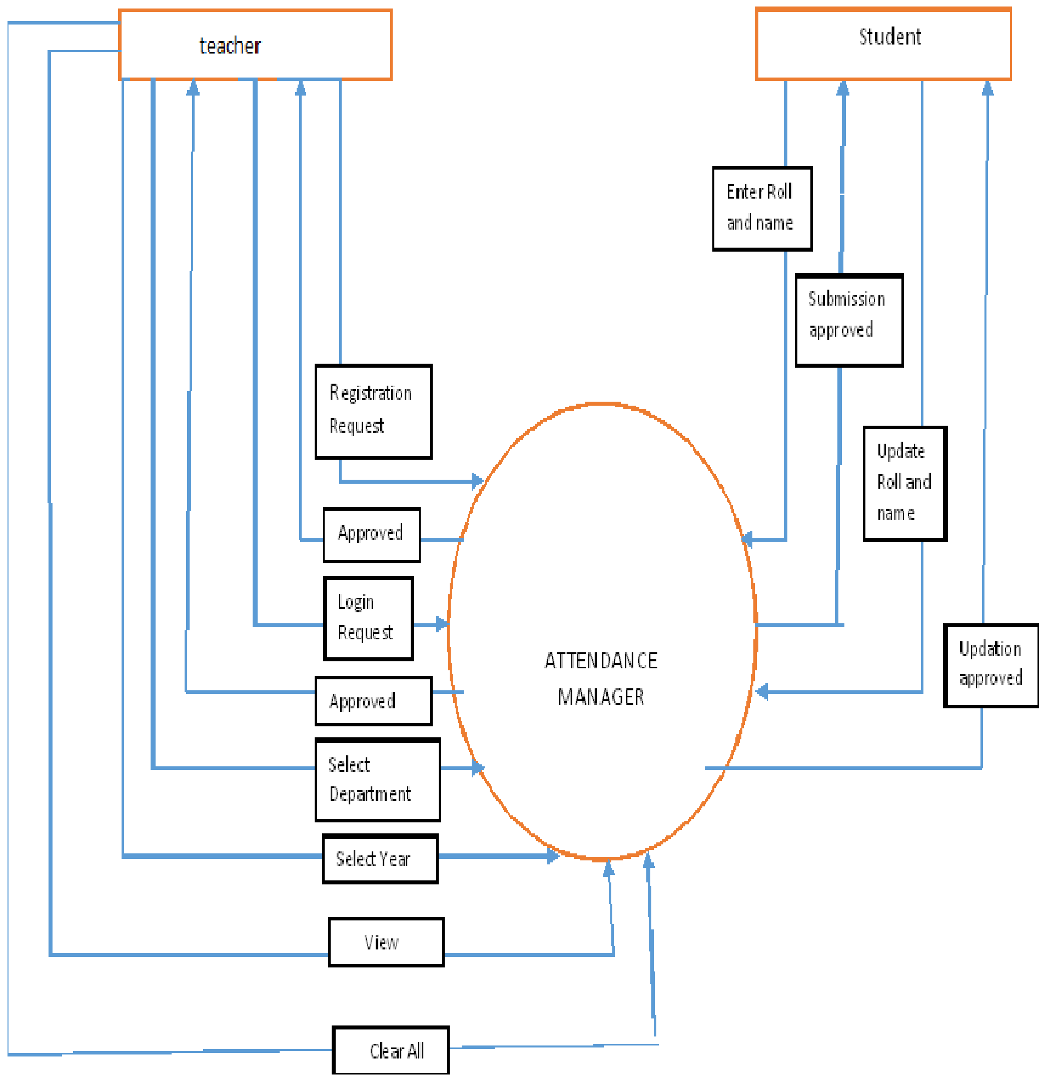
1) To describe the user requirements.

2) To establish a technical and functional requirement of software designing to define set of requirements that can be validated once the "attendance manager" is built.

Analysis phase describes the various operations performed by "attendance manager" and their relationships within and outside of the system. Main aspect of analysis is defining the "attendance manager" boundaries and determining feasibility of the system.

The system analysis focuses on a clear understanding of the needs of the client or the users, and what exact functionality is desired from the software. Analysis phase is involved with interviewing of system user. They and their existing documents help to understand about their operations, are the resources of information for the analysis. Data are collected during analysis on the available documents, and transactions handled by the system user.

Context diagram of "attendance manager



User characteristics

The system has single user that is the faculty of the institute. Here is a summary of the permissions enjoyed by the user.

➢ Faculty

- The faculty logs in through his account and gets the departments list and respective years
- The faculty can take attendance for their respective departments and store the details in the mobile internal database
- The faculty can also view the attendance details if required at later stage.

➢ General constraints

The general constraints of this system are of two types:-

1. Hardware constraints
2. Software constraints

Any devices which can run android operating system. And the following are the software constraints of the system.

➢ Server side

Database server: phpmyadmin or higher

Php: php 4.4.0 or higher (5.2 recommended)

➢ Client side

Any network enabled device which is able to connect to the server and running the android operating system.

Functional requirements of the system

The functional requirements part discuss the functional behavior that should be possessed by the system. Each requirement maps to a high level function (fi) that transforms the given set of input data (ii) into output data (oi)

Different functional requirements possessed by the system are:-

✓ Login

Description: the faculty will login into the application with the given user name and password. If the username and password is correct, user will be prompt to proceed option else error message will be displayed.

Input: username and password

Output: prompt to ”login” option

✓ View departments

Description: after login, the faculty can view the departments.
Input: select the department

Output: view the years under that department.

✓ Select the particular year.

Description: the faculty can select the particular year from 1st year-4th year.
Input: select a particular year.
Output: faculty has a choice to choose between student activity and teacher activity.

✓ Select the student activity.

Description: the faculty chooses the activity for the student.
Input: student activity is chosen

Output: the student activity opens up

✓ Submission of student's roll and name

Description: Students go to the teacher and submit their roll and name into the app's database. If any problem occurred while submitting, students can update their roll and name.

Input: roll and name submitted

Output: roll and name of the students are stored in the app's database.

✓ Viewing of the roll and name of the students by the teacher:

Description: The faculty can view the roll and name of the students from the teacher activity

Input: select the view button in the teacher activity

Output: roll and name of the students are listed.

✓ Clearing of the database:

Description: After the teacher has noted down the attendance for the day, he/she can clear the database by selecting the clear button.

Input: select the clear button

Output: the database gets cleared.

DataBase:

Information pertaining to the structure and usage of data contained in the database, the metadata, is maintained in a data dictionary. The system catalog describes this metadata. The data dictionary, which is a database itself, documents the data. Each programmer can consult the data dictionary to learn what each piece of data and the various synonyms of the data fields mean. In this proposed attendance manager the following data dictionary is being followed:-

Table : user_teacher

Column name	*Datatype*	*Constraints*
Uid	Int	Auto-Increment
Email	Varchar	Primary key
Password	Varchar	Not null

Table : user_feedback

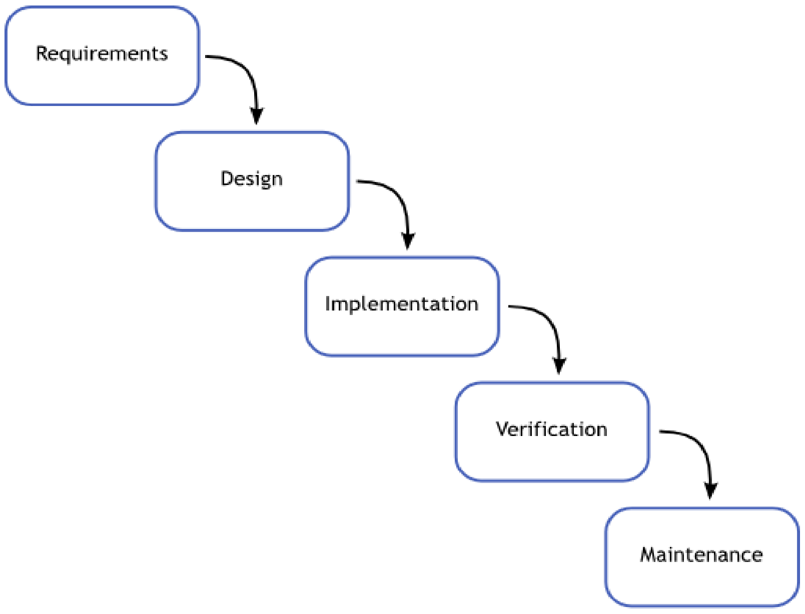
Column name	*Datatype*	*Constraints*
Uid	Int	Auto-Increment
Email	Varchar	Primary key
Comments	Varchar	Not null

❑ **Implementation Methodology**

Here I have used “Waterfall” model to develop or progress my System. It is the simplest model, which state that the phases are organized in a linear order. This model begins with feasibility analysis.
This model is divided into 5 phases. These are:-

- Analysis
- Design
- Code
- Test
- Support

The successfully demonstrating the feasibility of the project, requirements analysis and project planning begins. The design starts after the requirement analysis is complete, and coding begins after the design complete. Once the programming is completed, the code is integrated and testing is done. On successful completion of testing, the System is installed. After this, the regular operation and maintenance of the System take place.



Waterfall model

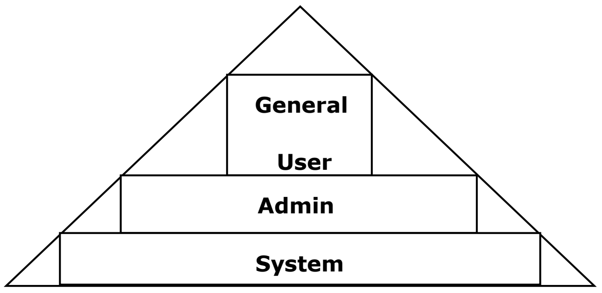
❑ Implementation of security mechanisms at various levels

Security in a Database involves both policies and mechanisms to protect the data and ensure that it isn't accessed, alter, or deleted without proper authorization. To implement security in the form constraints must be built into the system. The DBA, in consultation with the security administrator, specifies these controls. The online system enforces the controls by monitoring the actions of the users and limiting their actions within the constraints specified for them.
Here some controls or securities are inbuilt, depends on the database that is used for permanent storage purpose in the proposed the system. I use Android SQLite as a permanent storage; therefore, the system must maintain the inbuilt security of that database also.

The software has following types of user:

- Administrator (ADMIN)
- General User

Administrator is a super user and he has all power to grant access right to any user, create New user, remove any user accept "ADMIN", change password of any user.



Structure of Security Implementation