D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 1

Problem Statement: -

Download Install and Configure Eclipse / Android Studio on Linux/windows platform.

Objectives: -

The objective of this assignment is to learn how to install and configure different software **Software used:-**

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Android is an operating system that is built basically for Mobile phones. It is based on the Linux Kernel and other open-source software and is developed by Google. It is used for touchscreen mobile devices such as smartphones and tablets. But nowadays these are used in Android Auto cars, TV, watches, camera, etc. It has been one of the best-selling OS for smartphones. Android OS was developed by Android Inc. which Google bought in 2005. Various applications (apps) like games, music player, camera, etc. are built for these smartphones for running on Android. Google Play store features more than 3.3 million apps. The app is developed on an application known as Android Studio. These executable apps are installed through a bundle or package called APK(Android Package Kit).

Android Fundamentals

1. Android Programming Languages

In Android, basically, programming is done in two languages JAVA or **C++** and XML(Extension Markup Language). Nowadays KOTLIN is also preferred. The XML file deals with the design, presentation, layouts, blueprint, etc (as a front-end) while the JAVA or KOTLIN deals with the working of buttons, variables, storing, etc (as a back-end).

2. Android Components

The App components are the building blocks of Android. Each component has its own role and life cycles i.e from launching of an app till the end. Some of these components depend upon others also. Each component has a definite purpose. The four major app components are:

- Activities Services
- Broadcast Receivers:

- Content Provider:

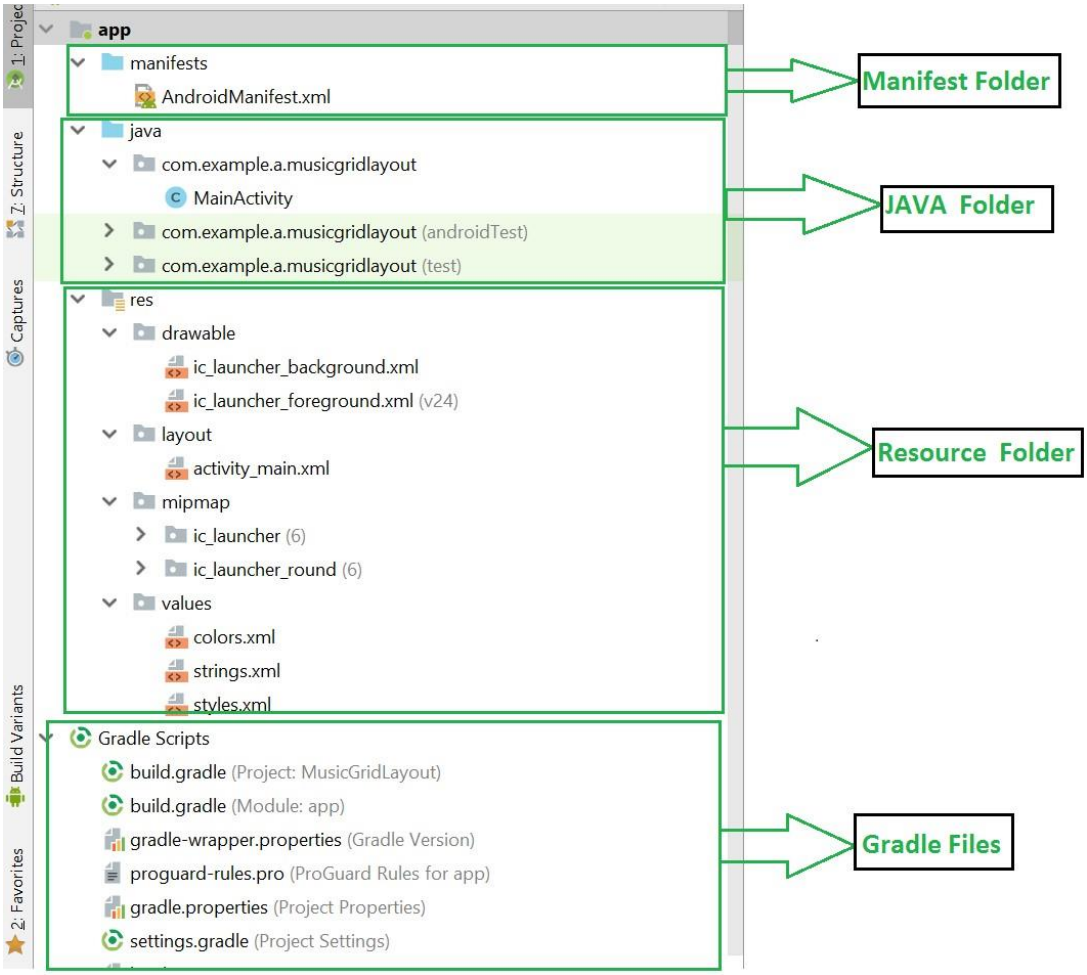
Activities: It deals with the UI and the user interactions to the screen. In other words, it is a User Interface that contains activities. These can be one or more depending upon the App. It starts when the application is launched. At least one activity is always present which is known as MainActivity. The activity is implemented through the following.

Syntax:

```
public class MainActivity extends Activity{
  // processes
}
```

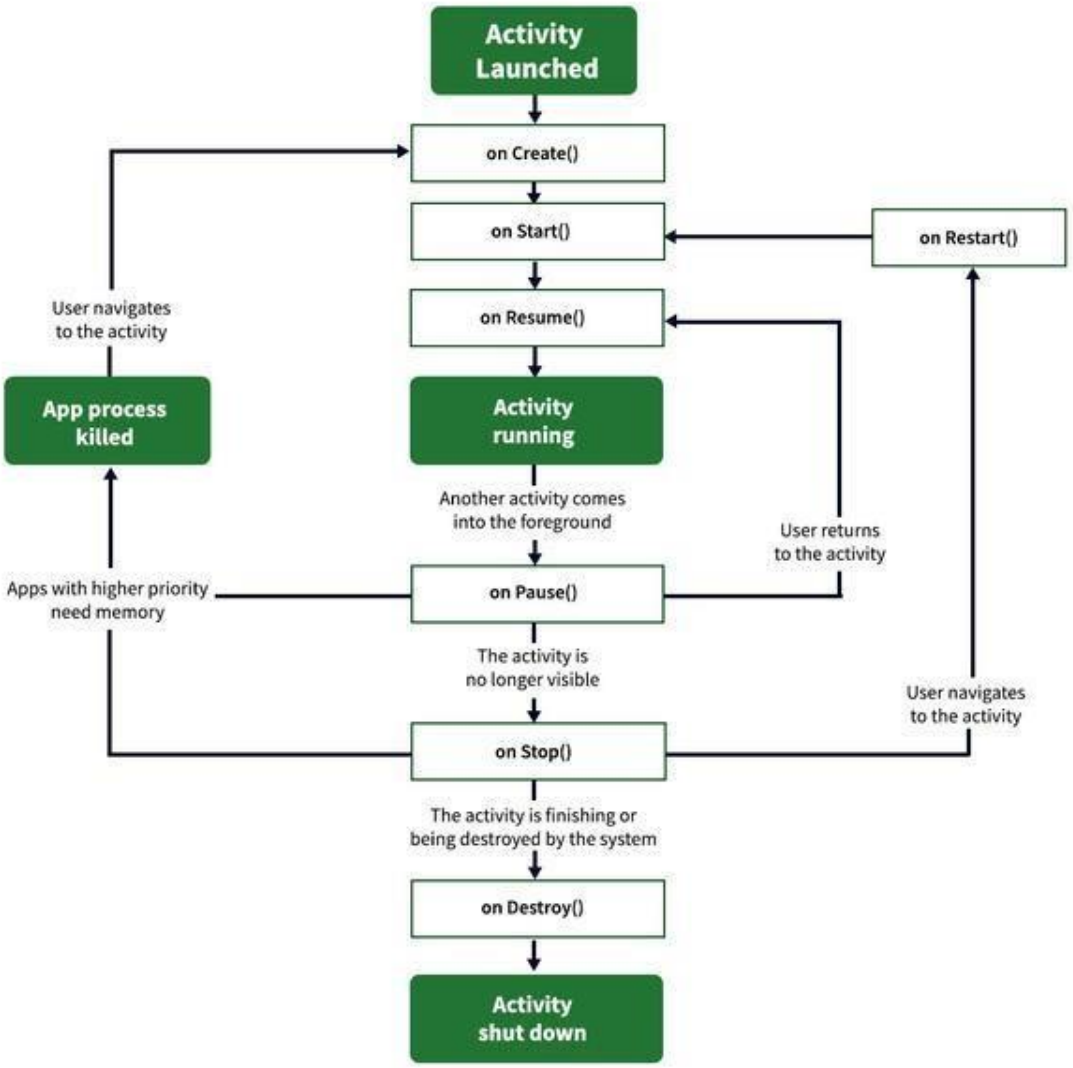
3. Structural Layout Of Android Studio

The basic structural layout of Android Studio is given below:



4. Lifecycle of Activity in Android App

The **Lifecycle** of Activity in Android App can be shown through this diagram:



Activity Lifecycle in Android

States of Android Lifecycle:

1. **OnCreate:** This is called when activity is first created.
2. **OnStart:** This is called when the activity becomes visible to the user.
3. **OnResume:** This is called when the activity starts to interact with the user.
4. **OnPause:** This is called when activity is not visible to the user.
5. **OnStop:** This is called when activity is no longer visible.
6. **OnRestart:** This is called when activity is stopped, and restarted again.
7. **OnDestroy:** This is called when activity is to be closed or destroyed.

Conclusion:

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 2

Problem Statement: -

Design a mobile application using implicit intent and explicit intent .

Objectives: -

The objective of this assignment is to learn about Intents and its types with implementation.

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Intents

Android Intent is the message that is passed between components such as activities, content providers, broadcast receivers, services etc.

It is generally used with startActivity() method to invoke activity, broadcast receivers etc.

The dictionary meaning of intent is intention or purpose. So, it can be described as the intention to do action.

Below are some applications of Intents:

Sending the User to Another App

Getting a Result from an Activity

Allowing Other Apps to Start Your Activity

Types of Android Intents

There are two types of intents in android

Implicit

Explicit

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.javatpoint.com"));
startActivity(intent);
```

Step by Step Implementation

Creating an Android App to Open a Webpage Using Implicit Intent

Step 1: Create a New Project in Android Studio

Create new project with desired name. One mainactivity.java and xml file get created.

Step 2: Working with the XML Files

Next, go to the **activity_main.xml file**, which represents the UI of the project. Add the necessary widgets(components such as editText, Button and configure them.

Syntax:

```
android:id="@+id/id_name"
```

Step 3: Working with the MainActivity File

Now, we will create the Backend of the App. For this, Open the MainActivity file and instantiate the component (Button) created in the XML file using the findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax:

```
ComponentType object = (ComponentType) findViewById(R.id.IdOfTheComponent);
```

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);   startActivity(i);
```

Android Explicit intent specifies the component to be invoked from activity. In other words, we can call another activity in android by explicit intent.

We can also pass the information from one activity to another using explicit intent.

Here, we are going to see an example to call one activity from another and vice-versa.

Step by Step Implementation

How to create an Android App to move to the next activity using Explicit Intent

Step 1: Create a New Project in Android Studio

Create new project with desired name. One mainactivity.java and xml file get created.

Step 2: Working with the activity_main.xml File

Next, go to the **activity_main.xml file**, which represents the UI of the project. Add the necessary widgets(components such as editText, Button and configure them.

Syntax:

android:id="@+id/id_name"

Step 3: Working with the MainActivity File

Now, we will create the Backend of the App. For this, Open the MainActivity file and instantiate the component (Button, TextView) created in the XML file using the findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax:

ComponentType object = (ComponentType) findViewById(R.id.IdOfTheComponent);

Intent i = new Intent(getApplicationContext(), <className>); startActivity(i);

Step 4: Working with the activity_main2.xml File

Now we have to create a second activity as a destination activity. The steps to create the second activity are **File > new > Activity > Empty Activity.**

Next, go to the **activity_main2.xml file**, which represents the UI of the project.

Step 5: Working with the MainActivity2 File

Now, we will create the Backend of the App. For this, Open the MainActivity file and instantiate the component (Button, TextView) created in the XML file using the findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax:

ComponentType object = (ComponentType) findViewById(R.id.IdOfTheComponent);

Intent i = new Intent(getApplicationContext(), <className>); startActivity(i);

Code

MainActivity.java

```
package com.example.intent_1;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;



public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void newsScreen(View view) {
      Intent i = new Intent(getApplicationContext(), MainActivity2.class);
      startActivity(i);
   }
}
```

Activity_main.xml

```
<?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"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="@color/Pink"
      android:text="Go to second activity"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.585" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="Next"
        android:onClick="newsScreen"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.525"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.679"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

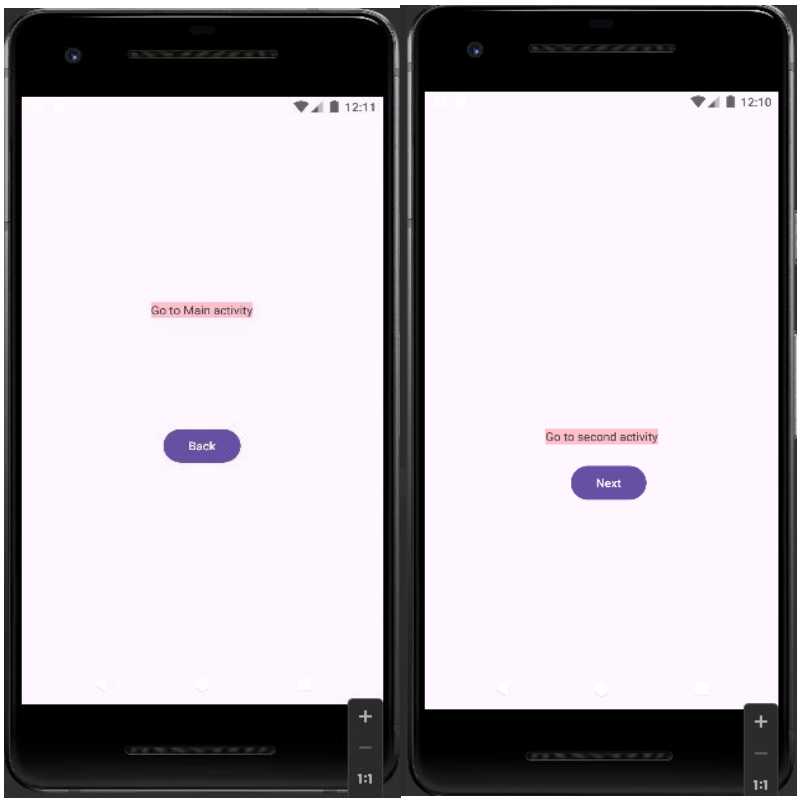
MainActivity2

```
package com.example.intent_1;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity2 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
    }
    public void homeScreen(View view) {
        Intent i = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(i);
    }
}
```

Activity_main2.xml

```
<?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"
    tools:context=".MainActivity2">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Main activity"
        android:background="@color/blue"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.348" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="372dp"
        android:onClick="homeScreen"
        android:text="Back"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Output:



Conclusion:-

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 3

Problem Statement: -

Design a mobile application to create two fragment and pass the data from one fragment to another

Objectives: -

The objective of this assignment is to learn about fragments and its types with implementation.

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Fragment

A **Fragment** is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**.

Following are important points about fragment −

- A fragment has its own layout and its own behaviour with its own life cycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-pane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behaviour that has no user interface component.
- Fragments were added to the Android API in Honeycomb version of Android which API version 11.

You create fragments by extending **Fragment** class and You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a **<fragment>** element.

Prior to fragment introduction, we had a limitation because we can show only a single activity on the screen at one given point in time. So we were not able to divide device screen and control different parts separately. But with the introduction of fragment we got more flexibility and removed the limitation of having a single activity on the screen at a time. Now we can have a single activity but

each activity can comprise of multiple fragments which will have their own layout, events and complete life cycle.

How to use Fragments?

This involves number of simple steps to create Fragments.

- First of all decide how many fragments you want to use in an activity. For example let's we want to use two fragments to handle landscape and portrait modes of the device.
- Next based on number of fragments, create classes which will extend the *Fragment* class. The Fragment class has above mentioned callback functions. You can override any of the functions based on your requirements.
- Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.
- Finally modify activity file to define the actual logic of replacing fragments based on your requirement.

Types of Fragments

Basically fragments are divided as three stages as shown below.

- Single frame fragments − Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.
- List fragments − fragments having special list view is called as list fragment
- Fragments transaction − Using with fragment transaction. we can move one fragment to another fragment.

This example demonstrate about How to pass data from one fragment to another fragment in android

Step 1 − Create a new project in Android Studio, go to File ⇒ New Project and fill all required details to create a new project.

Step 2 − Add the following code to res/layout/activity_main.xml.

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"  android:layout_width = "match_parent"
android:layout_height = "match_parent">
  <LinearLayout
    android:id = "@+id/linearlayout01"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "#ccc"
android:layout_weight = "1"
android:orientation = "vertical">
```

```
<fragment android:name = "com.example.myapplication.FirstFragment"
android:id = "@+id/frag_1" android:layout_width = "fill_parent"
android:layout_height = "fill_parent" />
</LinearLayout> <LinearLayout
android:id = "@+id/linearlayout02"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:layout_weight = "1"
android:background = "#eee"
android:orientation = "vertical">
<fragment android:name = "com.example.myapplication.SecondFragment"
android:id = "@+id/frag_2" android:layout_width = "fill_parent"
android:layout_height = "fill_parent" />
</LinearLayout></LinearLayout>
```

In the above code, we have taken fragments to pass the data between two fragments.

Step 3 − Add the following code to src /MainActivity.java

```
<?xml version = "1.0" encoding = "utf-8"?> import android.os.Bundle; import
android.support.v4.app.FragmentActivity; public class MainActivity extends
FragmentActivity implements OnButtonPressListener {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
public void onButtonPressed(String msg) {
// TODO Auto-generated method stub
SecondFragment Obj = (SecondFragment) getSupportFragmentManager().findFragmentById(R.id.frag_2);
Obj.onFragmentInteraction(msg);
}
}
```

Step 4 − Add the following code to src / FirstFragment.java

```
<?xml version = "1.0" encoding = "utf-8"?> import
android.annotation.SuppressLint;              import
android.app.Activity;                         import
android.content.Context;                      import
android.os.Bundle;                            import
android.support.annotation.NonNull;           import
android.support.annotation.Nullable;          import
android.support.v4.app.Fragment;              import
android.view.LayoutInflater;                  import
android.view.View;                            import
android.view.ViewGroup;                       import
android.widget.TextView;          public      class
FirstFragment extends Fragment{
  @Override
public View onCreateView(LayoutInflater inflater, ViewGroup
 container,Bundle savedInstanceState) {
    ViewGroup root = (ViewGroup) inflater.inflate(R.layout.fragment, null);
init(root);
return root;
  }
  OnButtonPressListener buttonListener;
  @Override
public void onAttach(Activity activity) {
super.onAttach(activity);
try {
      buttonListener = (OnButtonPressListener) getActivity();
    }
catch (ClassCastException e) {
      throw new ClassCastException(activity.toString() + " must implement onButtonPressed");
    }
 }
  void init(ViewGroup root) {
    TextView but = (TextView)root.findViewById(R.id.text);
but.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
 public void onClick(View v) {
          // TODO Auto-generated method stub
buttonListener.onButtonPressed("Message From First Fragment");
        }
      });
}}
```

Step 5 − Add the following code to src / SecondFragment.java

```
<?xml version = "1.0" encoding = "utf-8"?>
import android.annotation.SuppressLint;
import android.os.Bundle; import
android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log; import
android.view.LayoutInflater; import
android.view.View; import
android.view.ViewGroup; import
android.widget.TextView;
public class SecondFragment extends Fragment {
  TextView textView;
  View view;
  @Nullable @Override public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) { view =
inflater.inflate(R.layout.fragment, container, false); return
view;
  }
  public void onFragmentInteraction(String uri) {
Log.d("sai",uri);
    textView = view.findViewById(R.id.text);
textView.setText(uri);  }}
```

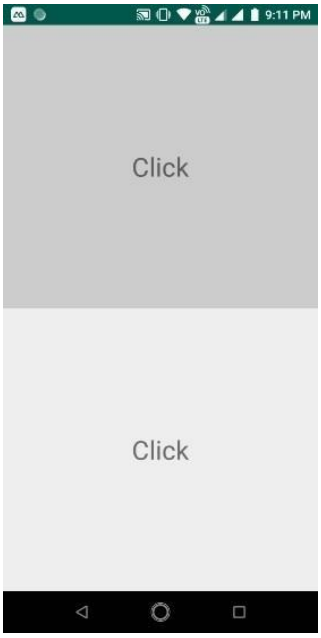
Step 6 − Add the following code to res/layout/ fragment.xml.

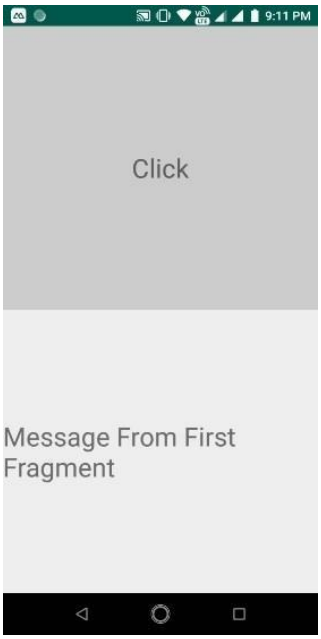
```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout
xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "match_parent"
android:gravity = "center"
android:layout_height = "match_parent">
  <TextView
android:id = "@+id/text"
android:textSize = "30sp"
android:text = "Click"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content" />
</LinearLayout>
```

Step 7 − Add the following code to src/ OnButtonPressListener.java.

```
<?xml version = "1.0" encoding = "utf-8"?> public interface
OnButtonPressListener {  public void onButtonPressed(String msg);}
```

Output





Conclusion

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 4

Problem Statement: -

Design a mobile application to create home page using grid layout

Objectives: -

The objective of this assignment is to learn how to design grid layout and arrange different widgets

.

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Layouts in android :

Android Layout is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. Following are some layouts used in android to arrange different widgets.

1. Android Linear Layout: LinearLayout is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property.
2. Android Relative Layout: RelativeLayout is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fix to the top of the parent).
3. Android Constraint Layout: ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.

4. Android Frame Layout: FrameLayout is a ViewGroup subclass, used to specify the position of View elements it contains on the top of each other to display only a single View inside the FrameLayout.
5. Android Table Layout: TableLayout is a ViewGroup subclass, used to display the child View elements in rows and columns.
6. Android Web View: WebView is a browser that is used to display the web pages in our activity layout.
7. Android ListView: ListView is a ViewGroup, used to display scrollable lists of items in a single column.
8. Android Grid View: GridView is a ViewGroup that is used to display a scrollable list of items in a grid view of rows and columns.

The XML layout file contains at least one root element in which additional layout elements or widgets can be added to build a View hierarchy.

GridView

Android **GridView** shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a **ListAdapter**

Grid view

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc.

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.

GridView Attributes

Following are the important attributes specific to GridView −

Sr.No	Attribute & Description
1	**android: id** This is the ID which uniquely identifies the layout.

2	**android: columnWidth** This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.
3	**android: gravity** Specifies the gravity within each cell. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
4	**android: horizontalSpacing** Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.
5	**android: numColumns** Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space.
6	**android: stretchMode** Defines how columns should stretch to fill the available empty space, if any. This must be either of the values − • none − Stretching is disabled. • spacingWidth − The spacing between each column is stretched. • columnWidth − Each column is stretched equally. • spacingWidthUniform − The spacing between each column is uniformly stretched..
7	**android: verticalSpacing** Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.

Activity_main.xml

```
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="5"
    android:columnCount="2"
    android:background="#3F51B5"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Programming        Logo"
        android:background="@color/Orange"/>


    <ImageView
        android:layout_width="195dp"
        android:layout_height="174dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_4"/>

    <ImageView
        android:layout_width="200dp"
        android:layout_height="172dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_5"/>

    <ImageView
        android:layout_width="193dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_6"/>

    <ImageView
        android:layout_width="200dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_7"/>

    <ImageView
        android:layout_width="193dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_8"/>
```

```
    <ImageView
        android:layout_width="200dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_9"/>
    <ImageView
        android:layout_width="200dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_10"/>
    <ImageView
        android:layout_width="200dp"
        android:layout_height="154dp"
        android:layout_margin="10dp"
        android:src="@drawable/img_11"/>

</GridLayout>
```

Output:

Conclusion:-

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 5

Problem Statement: -

Design a mobile application to create different dialog boxes and menu (popup, option , context)

Objectives: -

The objective of this assignment is to learn how to create and use different types of Dialog boxes and Menus **Software used:-**

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Dialogs in Android:

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:

AlertDialog

A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.

DatePickerDialog or TimePickerDialog

A dialog with a pre-defined UI that allows the user to select a date or time.

```
// Example of AlertDialog

public class StartGameDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
// Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setMessage(R.string.dialog_start_game)
               .setPositiveButton(R.string.start, new
DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // START THE GAME!
                   }
               })
               .setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // User cancelled the dialog
                   }
               });
        // Create the AlertDialog object and return it
return builder.create();
    }
}
```

Menus in Android:

In android, Menu is an important part of the UI component which is used to provide some common functionality around the application. With the help of menu, users can experience a smooth and consistent experience throughout the application. In order to use menu, we should define it in a separate XML file and use that file in our application based on our requirements. Also, we can use menu APIs to represent user actions and other options in our android application activities.

Types of Menus

In android, we have three types of Menus available to define a set of options and actions in our android applications. The Menus in android applications are the following:

- Android Options Menu
- Android Context Menu
- Android Popup Menu

Android Options Menu is a primary collection of menu items in an android application and is useful for actions that have a global impact on the searching application. **Android Context Menu** is a floating menu that only appears when the user clicks for a long time on an element and is useful for elements that affect the selected content or context frame. **Android Popup Menu** displays a list of items in a vertical list which presents the view that invoked the menu and is useful to provide an overflow of actions related to specific content.

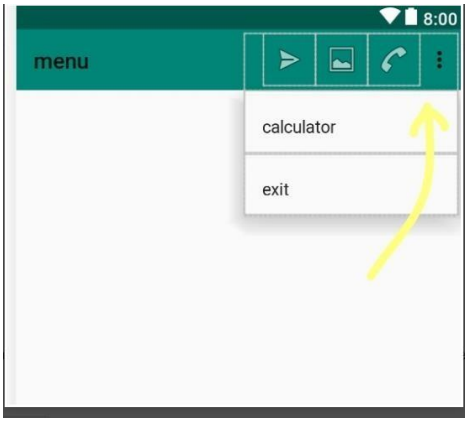
Way to create menu directory and menu resource file:

To create the menu directory just right-click on **res** folder and navigate to **res->New->Android Resource Directory.** Give resource directory name as menu and resource type also menu. one directory will be created under res folder.

To create xml resource file simply right-click on menu folder and navigate to **New->Menu Resource File.** Give name of file as menu_example. One menu_example.xml file will be created under menu folder.

Popup menu

Context Menu



Option Menu

Program:

activity_main.xml | MainActivity.java | popup.xml

```
package com.example.menu_exp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.PopupMenu;
import android.widget.Toast;
4 usages
public class MainActivity extends AppCompatActivity { Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button=(Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                PopupMenu popup= new PopupMenu( context: MainActivity.this, button);
                popup.getMenuInflater().inflate(R.menu.popup,popup.getMenu());
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem menuItem) {
                        Toast.makeText( context: MainActivity.this,  text: "you selected :" + menuItem.getTitle(), Toast.LENGTH_SHORT).show();
                        return true;
                    }
                });
                popup.show();
            }
        });
    }}
```

activity_main.xml | MainActivity.java | popup.xml

Code | Split | Design

```xml
<?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"
    tools:context=".MainActivity">


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Languages" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_main.xml | MainActivity.java | popup.xml

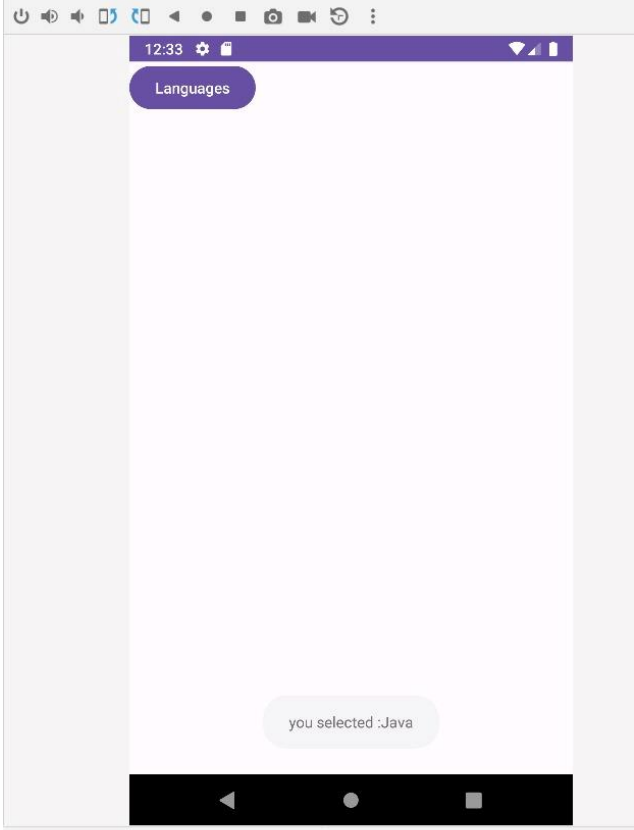
```xml
<?xml version="1.0" encoding="utf-
<menu xmlns:android="http://schemas.android
    <item
        android:id="@+id/one"
        android:title="Java" />
    <item
        android:id="@+id/two"
        android:title="Python"/>
    <item
        android:id="@+id/three"
        android:title="Cpp"/>
</menu>
```

Palette: Cast Button, Menu Item, Search Item, Switch Item, Menu, Group

Component Tree

menu/popup.xml | Pixel

menu



Conclusion:

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 6

Problem Statement: -

Design a mobile application to Activity using fragment

Objectives: -

The objective of this assignment is to learn how to create and use fragments

Software used:-

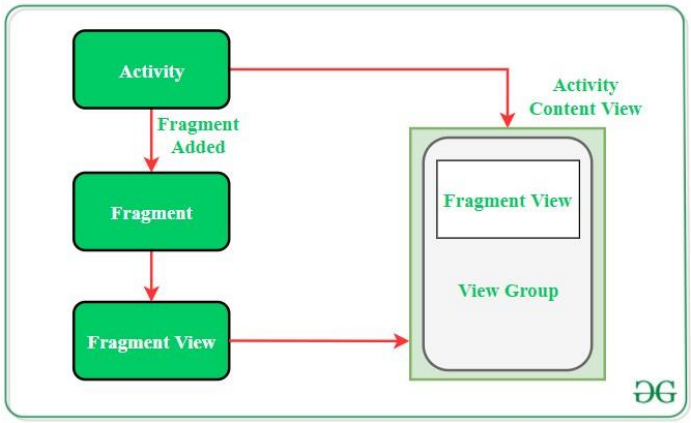
64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

In Android, the fragment is the part of Activity which represents a portion of User Interface(UI) on the screen. It is the modular section of the android activity that is very helpful in creating UI designs that are flexible in nature and auto-adjustable based on the device screen size. The UI flexibility on all devices improves the user experience and adaptability of the application. Fragments can exist only inside an activity as its lifecycle is dependent on the lifecycle of host activity. For example, if the host activity is paused, then all the methods and operations of the fragment related to that activity will stop functioning, thus fragment is also termed as **sub-activity**. Fragments can be added, removed, or replaced dynamically i.e., while activity is running.

<fragment> tag is used to insert the fragment in an android activity layout. By dividing the activity's layout multiple fragments can be added in it.

Below is the pictorial representation of fragment interaction with the activity:



*https://www.geeksforgeeks.org/fragment-lifecycle-in-android

Types of Android Fragments

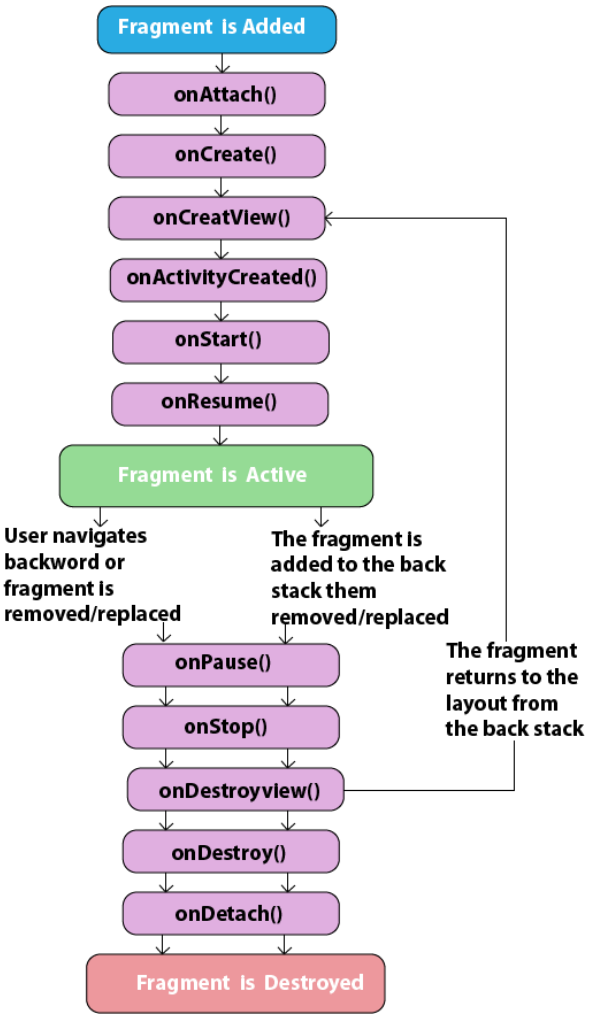
Single Fragment: Display only one single view on the device screen. This type of fragment is mostly used for mobile phones.

List Fragment: This Fragment is used to display a list-view from which the user can select the desired sub-activity. The menu drawer of apps like Gmail is the best example of this kind of fragment.

Fragment Transaction: This kind of fragments supports the transition from one fragment to another at run time. Users can switch between multiple fragments like switching tabs.

Android Fragment Lifecycle

The lifecycle of android fragment is like the activity lifecycle. There are 12 lifecycle methods for fragment.



***https://www.javatpoint.com/android-fragments**

Android Fragment Lifecycle Methods

No.	Method	Description
1)	onAttach(Activity)	it is called only once when it is attached with activity.
2)	onCreate(Bundle)	It is used to initialize the fragment.

3)	onCreateView(LayoutInflater, ViewGroup, Bundle)	creates and returns view hierarchy.
4)	onActivityCreated(Bundle)	It is invoked after the completion of onCreate() method.
5)	onViewStateRestored(Bundle)	It provides information to the fragment that all the saved state of fragment view hierarchy has been restored.
6)	onStart()	makes the fragment visible.
7)	onResume()	makes the fragment interactive.
8)	onPause()	is called when fragment is no longer interactive.
9)	onStop()	is called when fragment is no longer visible.
10)	onDestroyView()	allows the fragment to clean up resources.
11)	onDestroy()	allows the fragment to do final clean up of fragment state.
12)	onDetach()	It is called immediately prior to the fragment no longer being associated with its activity.

Program:

activity_main.xml | colors.xml | fragment_1.xml | fragment_2.xml | fragment_2.java | fragment_1.java | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="wrap_content"
        android:layout_height="wrap_content"S
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <fragment
        android:layout_width="3dp"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:id="@+id/exp06"
        android:name="com.example.exp06.fragment_1"
        />

    <fragment
        android:layout_width="3dp"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:name="com.example.exp06.fragment_2"
        android:id="@+id/exp006"
        />
</LinearLayout>
```

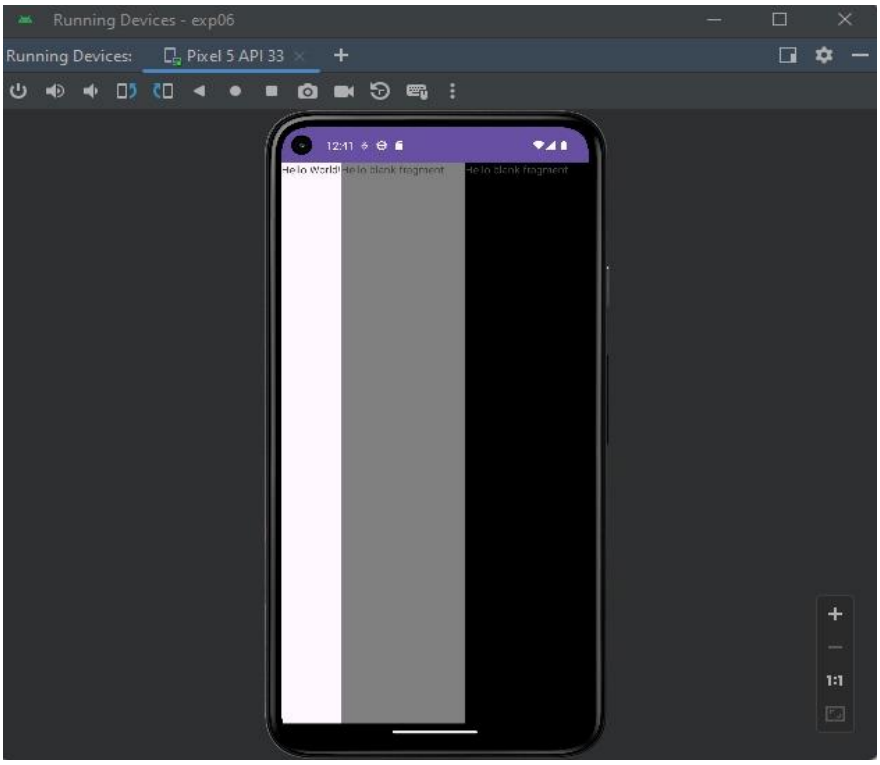
activity_main.xml | colors.xml | fragment_1.xml | fragment_2.xml | fragment_2.java | fragment_1.java | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragment_1">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/hello_blank_fragment"
        android:background="@color/grey"/>

    <TextView
        android:layout_width="3dp"
        android:layout_height="match_parent"
        android:id="@+id/exp06"
        android:layout_marginTop="50dp"
        android:layout_marginLeft="50dp"
       />


</FrameLayout>
```

activity_main.xml | colors.xml | fragment_1.xml | fragment_2.xml | fragment_2.java | fragment_1.java | MainActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragment_2">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello blank fragment"
        android:background="@color/black"/>
    <TextView
        android:layout_width="3dp"
        android:layout_height="match_parent"
        android:id="@+id/exp06"
        android:layout_marginTop="50dp"
        android:layout_marginLeft="50dp"
        />

</FrameLayout>
```

Output



Conclusion:

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 7

Problem Statement: -

Design a mobile application to create registration application which having spinner (subject), radio button (gender), qualification (check box), first insert the value and then show the data in show activity.

Objectives: -

The objective of this assignment is to learn how to create and use different types of buttons and views.

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

Spinner view

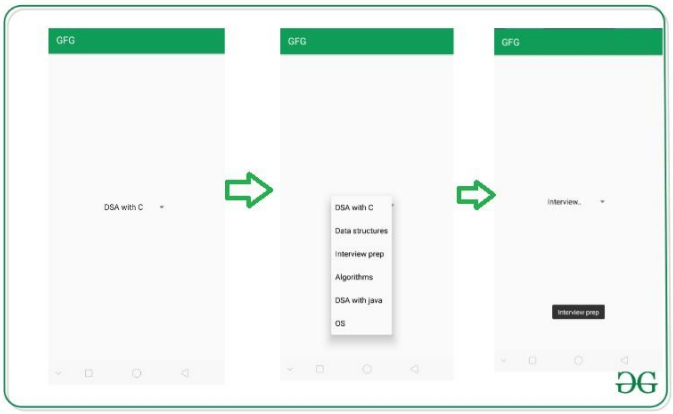
Android Spinner is a view similar to the dropdown list which is used to select one option from the list of options. It provides an easy way to select one item from the list of items and it shows a dropdown list of all values when we click on it. The default value of the android spinner will be the currently selected value and by using **Adapter** we can easily bind the items to the spinner objects.

Different Attributes for Spinner Widget

XML attributes	Description
android:id	Used to specify the id of the view.
android:textAlignment	Used to the text alignment in the dropdown list.
android:background	Used to set the background of the view.
android:padding	Used to set the padding of the view.

XML attributes	Description
android:visibility	Used to set the visibility of the view.
android:gravity	Used to specify the gravity of the view like center, top, bottom, etc

Spinner output



***https://www.geeksforgeeks.org/spinner-in-android-using-java-with-example/**

Radio button and Radio group

In Android, RadioButton are mainly used together in a RadioGroup. In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others. It means at one time we can checked only one radio button from a group of radio buttons which belong to same radio group.

RadioButon is a two state button that can be checked or unchecked. If a radio button is unchecked then a user can check it by simply clicking on it. Once a RadiaButton is checked by user it can't be unchecked by simply pressing on the same button. It will automatically unchecked when you press any other RadioButton within same RadioGroup. RadioGroup is a widget used in Android for the grouping of radio buttons and provide the feature of selecting only one radio button from the set. When a user try to select any other radio button within same radio group the previously selected radio button will be automatically unchecked.

CheckBox

In Android, CheckBox is a type of two state button either unchecked or checked in Android. Or you can say it is a type of on/off switch that can be toggled by the users. You should use checkbox when presenting a group of selectable options to users that are not mutually exclusive. CompoundButton is the parent class of CheckBox class.

In android there is a lot of usage of check box. For example, to take survey in Android app we can list few options and allow user to choose using CheckBox. The user will simply checked these checkboxes rather than type their own option in EditText. Another very common use of CheckBox is as remember me option in Login form.

You can check the current state of a check box programmatically by using isChecked() method. This method returns a Boolean value either true or false, if a check box is checked then it returns true otherwise it returns false. Below is an example code in which we checked the current state of a check box.

Conclusion:

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 8

Problem Statement: -

Design a mobile application to show list using Recycler View

Objectives: -

The objective of this assignment is to learn how to use Recycler View for displaying a list

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

RecyclerView

RecyclerView is a ViewGroup added to the android studio as a successor of the GridView and ListView. It is an improvement on both of them and can be found in the latest v-7 support packages. It has been created to make possible construction of any lists with **XML** layouts as an item which can be customized vastly while *improving on the efficiency of ListViews and GridViews*. This improvement is achieved by recycling the views which are out of the visibility of the user. For example, if a user scrolled down to a position where items 4 and 5 are visible; items 1, 2, and 3 would be cleared from the memory to reduce memory consumption. **Implementation**: To implement a basic RecyclerView three sub-parts are needed to be constructed which offer the users the degree of control they require in making varying designs of their choice.

1. **The Card Layout:** The card layout is an XML layout which will be treated as an item for the list created by the RecyclerView.
2. **The ViewHolder:** The ViewHolder is a java class that stores the reference to the card layout views that have to be dynamically modified during the execution of the program by a list of data obtained either by online databases or added in some other way.
3. **The Data Class:** The Data class is a custom java class that acts as a structure for holding the information for every item of the RecyclerView.

To click on recycler item:

To click on item of recycler view pass the instance of click interface in constructor of adapter

```
public class ClickListiner{

// here index is index
// of item clicked
public click(int index);

}
```

The Adapter: The adapter is the main code responsible for RecyclerView. It holds all the important methods dealing with the implementation of RecylcerView. The basic methods for a successful implementation are:

- onCreateViewHolder: which deals with the inflation of the card layout as an item for the RecyclerView.
- onBindViewHolder: which deals with the setting of different data and methods related to clicks on particular items of the RecyclerView.
- getItemCount: which Returns the length of the RecyclerView.
- onAttachedToRecyclerView: which attaches the adapter to the RecyclerView.

// XML code for layout

```
<!-- XML Code illustrating card layout usage. -->
<?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"
 android:layout_width="match_parent"
 android:layout_height="105dp">


<TextView
 android:layout_width="200dp"
android:id="@+id/examName"
android:textSize="16sp"
android:layout_marginStart="20dp"
android:text="First Exam"
```

```
android:textColor="@color/black"
 android:layout_marginEnd="20dp"
android:maxLines="1"
android:layout_marginTop="15dp"
 android:layout_height="wrap_content"/>

 <ImageView
android:id="@+id/examPic"
android:layout_width="20dp"
android:layout_height="20dp"
 android:layout_below="@+id/examName"
android:tint="#808080"
android:layout_marginStart="20dp"
android:layout_marginTop="7dp"
 app:srcCompat="@drawable/baseline_schedule_black_36dp"/>

 <TextView
android:id="@+id/examDate"
 android:layout_toEndOf="@+id/examPic"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/examName"
 android:layout_marginTop="5dp"
 android:layout_marginEnd="20dp"
 android:layout_marginStart="10dp"
 android:gravity="center"
android:text="May 23, 2015"
 android:textSize="16sp"/>

 <ImageView
android:id="@+id/examPic2"
 android:layout_width="20dp"
 android:layout_height="20dp"
```

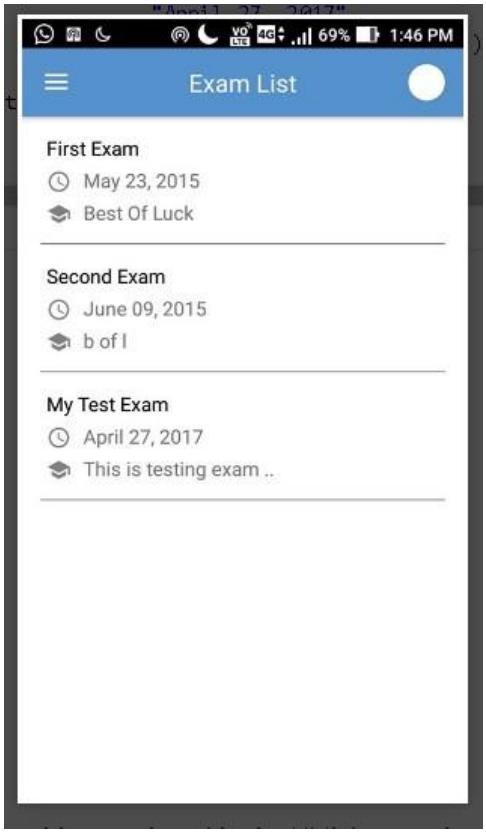
```
 android:layout_below="@+id/examDate"
 android:tint="#808080"
 android:layout_marginStart="20dp"
 android:layout_marginTop="7dp"
 app:srcCompat="@drawable/baseline_school_black_36dp
"/>

 <TextView
android:id="@+id/examMessage"
 android:layout_toEndOf="@+id/examPic2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/examDate"
 android:layout_marginEnd="20dp"
 android:layout_marginTop="5dp"
 android:layout_marginStart="10dp"
 android:gravity="center"
android:text="Best Of Luck"
 android:textSize="16sp"/>

 <TextView
android:id="@+id/border2"
 android:layout_width="match_parent"
 android:layout_height="1dp"
 android:layout_marginStart="15dp"
 android:layout_marginEnd="15dp"
 android:layout_alignParentBottom="true"
 android:background="#808080"/>

</RelativeLayout>
```

```
// Java Code for RecyclerView
package com.example.admin.example;

import android.content.Context;
import android.content.Intent; import
android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import com.prolificinteractive
        .materialcalendarview
        .MaterialCalendarView;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class exam extends AppCompatActivity
implementsNavigationView.OnNavigationItemSelectedListen
er {

         ImageGalleryAdapter2 adapter;
        RecyclerView recyclerView;
        ClickListiner listiner;
```

```
@Override
  protected void onCreate(Bundle savedInstanceState)
{
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_exam);
          Toolbar toolbar
          = (Toolbar)findViewById(R.id.toolbar);
 toolbar.setTitle("");
 setSupportActionBar(toolbar);

 List<examData> list = new ArrayList<>();
 list = getData();

 recyclerView
          = (RecyclerView)findViewById(R.id.recyclerView);
 listiner = new ClickListiner() {
 @Override
  public void click(int index)
    {
       Toast.makeTexT(this,"clicked item index is "+index,Toast.LENGTH_LONG). show();
     }
 };
 adapter = new ImageGalleryAdapter2( list,
 getApplication(),listiner);
 recyclerView.setAdapter(adapter);
 recyclerView.setLayoutManager(newLinearLayoutMan
 ager(exam.this));
}
 @Override
 public void onBackPressed()
{
          super.onBackPressed();
}
```

```
 // Sample data for RecyclerView
private List<examData> getData()
{
  List<examData> list = new ArrayList<>();
   list.add(new examData("First Exam", "May 23, 2015",
                                                "Best Of Luck"));
   list.add(new examData("Second Exam", "June 09, 2015", "b of l"));
  list.add(new examData("My Test Exam", "April 27, 2017",
                                                    "This is testing exam .."));
   return list;
          }
}
```

Output



***https://www.geeksforgeeks.org/android-recyclerview/**

Conclusion

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 9

Problem Statement: -

Design a mobile application to Show any website using web view

Objectives: -

The objective of this assignment is to learn how to use web view to show website

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

WebView

WebView is a view that displays web pages inside the application. It is used to turn the application into a web application.

public class WebView extends AbsoluteLayout implements

ViewTreeObserver.OnGlobalFocusChangeListener,

ViewGroup.OnHierarchyChangeListener

Class Hierarchy: java.lang.Object ↳
android.view.View ↳
android.view.ViewGroup ↳
android.widget.AbsoluteLayout
↳ android.webkit.WebView

Step by Step Implementation

Step 1: Create a New Project in Android Studio

Step 2: Write the webview code in MainActivity File

Step 3: Write the webview widget code in activity_main.xml file.
Step 4: Adding Permissions to the AndroidManifest.xml File

In AndroidManifest.xml, one needs to include the below permission, in order to access the internet
<uses-permission android:name="android.permission.INTERNET" />

Step 5: Run the APP and get the desired output.

Program

MainActivity.java

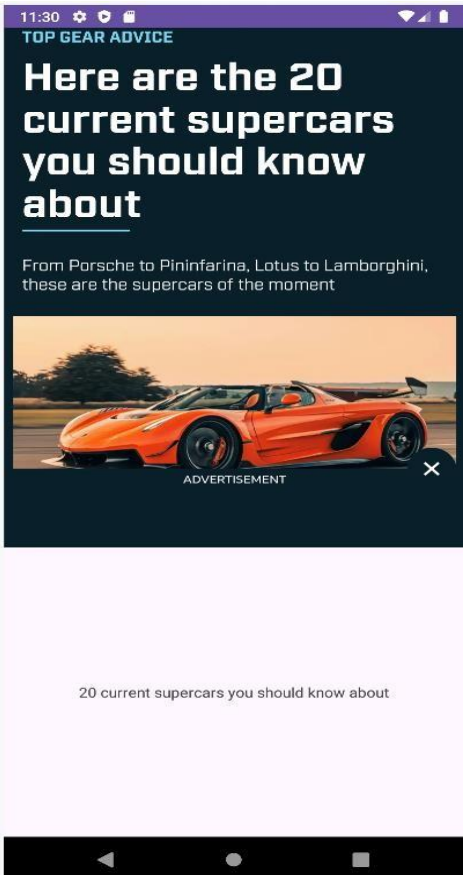
```
package com.aditya.myapplication;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import    android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {

@Override
   protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       WebView wb = findViewById(R.id.webv);
       wb.loadUrl("https://www.topgear.com/car-news/top-gear-advice/here-are-20-current- supercars-you-should-know-about");
       wb.getSettings().setJavaScriptEnabled(true);
       wb.setWebViewClient(new WebViewClient());
   }
}
```

activity_main.xml

```
<?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"
tools:context=".MainActivity">


<WebView
       android:id="@+id/webv"
       android:layout_width="411dp"
       android:layout_height="541dp"
       app:layout_constraintBottom_toBottomOf="parent"
       app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
       app:layout_constraintTop_toTopOf="parent"
       app:layout_constraintVertical_bias="0.0" />

 <TextView
```

```
        android:id="@+id/textView"
        android:layout_width="wrap_content"
         android:layout_height="wrap_content"
        android:text="20 current supercars you should know about"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"app:layout_constraintTop_toBottomOf="@+i
d/webv" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Output :



Conclusion

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 10

Problem Statement: -

Design a mobile application for media player.

Objectives: -

The objective of this assignment is to learn how to add media player in application

Software used:-

64 bit Linux/Windows Operating System, JDK 8.1 or later, Eclipse/ Netbeans, Android studio

Theory :-

MediaPlayer.

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. These methods are listed below −

Sr.No	Method & description

1	**isPlaying()** This method just returns true/false indicating the song is playing or not
2	**seekTo(position)** This method takes an integer, and move song to that particular position millisecond
3	**getCurrentPosition()** This method returns the current position of song in milliseconds
4	**getDuration()** This method returns the total time duration of song in milliseconds
5	**reset()** This method resets the media player
6	**release()** This method releases any resource attached with MediaPlayer object
7	**setVolume(float leftVolume, float rightVolume)** This method sets the up down volume for this player
8	**setDataSource(FileDescriptor fd)** This method sets the data source of audio/video file
9	**selectTrack(int index)** This method takes an integer, and select the track from the list on that particular index
10	**getTrackInfo()** This method returns an array of track information

Example

Here is an example demonstrating the use of MediaPlayer class. It creates a basic media player that allows you to forward, backward, play and pause a song.

To experiment with this example, you need to run this on an actual device to hear the audio sound.

Steps	Description
1	You will use Android studio IDE to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add MediaPlayer code.
3	Modify the res/layout/activity_main to add respective XML components
4	Create a new folder under MediaPlayer with name as raw and place an mp3 music file in it with name as song.mp3
5	Run the application and choose a running android device and install the application on it and verify the results

*** https://www.tutorialspoint.com/android/android_mediaplayer.htm#**

Conclusion :

D. Y. Patil College of Engineering, Akurdi, Pune

Department of Electronics & Telecommunication Engineering

Class: BE **Subject : Android development**

Date:- **Roll No:-**

Experiment No. 11

Problem Statement: -

Create a drawable file in res folder.

MainActivity.java

```
package com.aditya.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity
{
@Override
   Protected void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
   }
}
```

activity_main.xml

```
<?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"
tools:context=".MainActivity">

<TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="A Circular Button made using Modifying a normal button using another xml file from res/drawable/ "
      android:textAlignment="center"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="1.0"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.217" />

<Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="@drawable/circle_button"
      android:text="Button"
      android:textColor="#FFFFFF"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.499"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

circle_button.xml

```
<shape
xmlns:android="http://schemas.android.com/apk/res/android"
        android:shape="oval">
    <stroke
        android:width="2dp"
        android:color="#FFFFFF" />
    <size
        android:layout_width="219dp"
        android:layout_height="85dp" />
</shape>
```

Output:

Conclusion: