<u>NAME: RUSHABH MISHRA</u>

CLASS:SYBSC(CS)

DIV: A

ROLL NO :4344

Practical 1:

Aim : Install Android Studio and Run Hello World App.

Android Studio provides a complete integrated development environment (IDE) including an advanced code editor and a set of app templates. In addition, it contains tools for development, debugging, testing, and performance that make it faster and easier to develop apps. You can test your apps with a large range of preconfigured emulators or on your own mobile device, build production apps, and publish on the Google Play store.

1. Navigate to the Android developers site and follow the instructions to download and install Android Studio.
2. Accept the default configurations for all steps, and ensure that all components are selected for installation.
3. After finishing the install, the Setup Wizard will download and install some additional components including the Android SDK. Be patient, this might take some time depending on your Internet speed, and some of the steps may seem redundant.
4. When the download completes, Android Studio will start, and you are ready to create your first project.

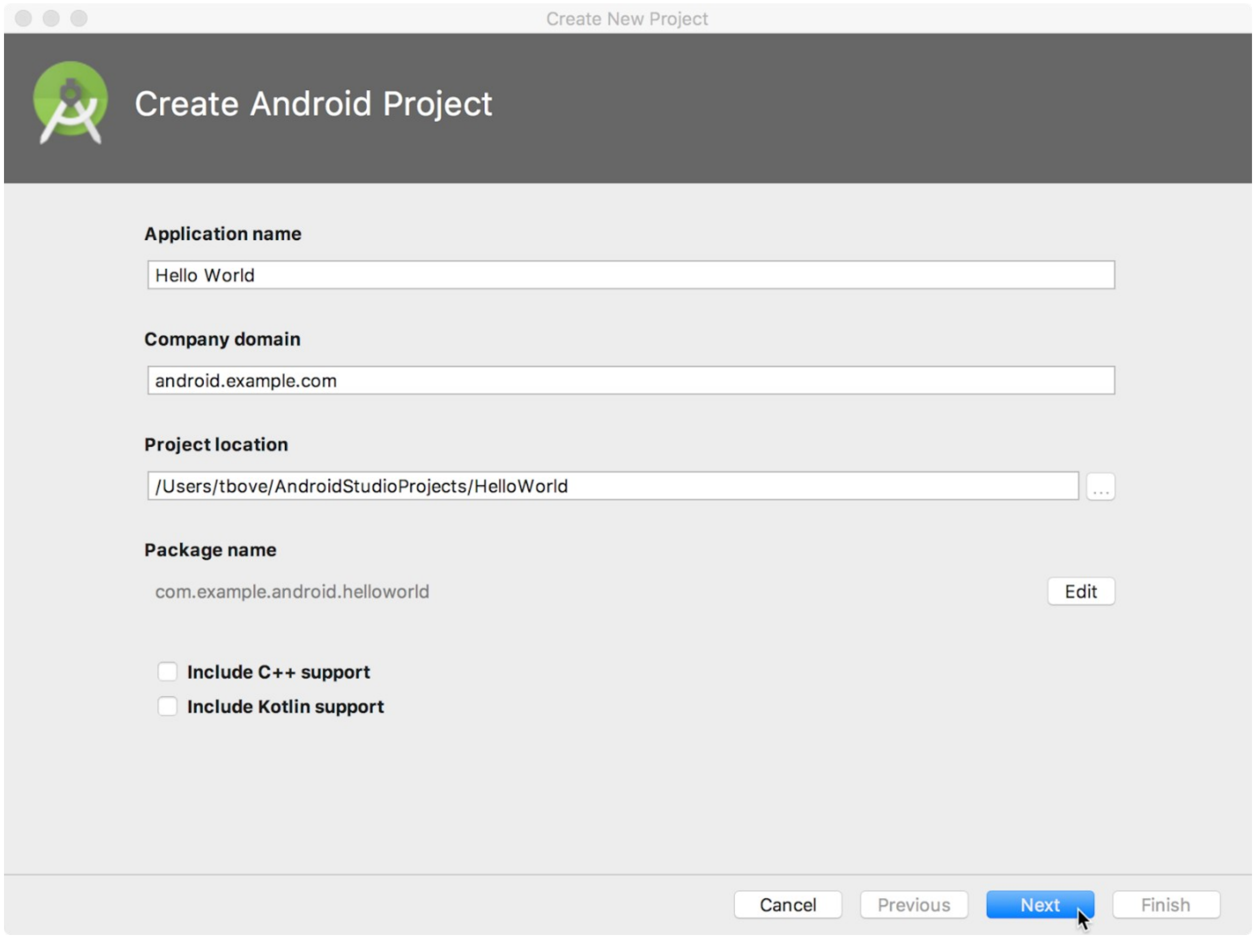
Task 2: Create the Hello World app

In this task, you will create an app that displays "Hello World" to verify that Android studio is correctly installed, and to learn the basics of developing with Android Studio.

2.1 Create the app project

1. Open Android Studio if it is not already opened.
2. In the main **Welcome to Android Studio** window, click **Start a new Android Studio project**.

3. In the **Create Android Project** window, enter **Hello World** for the **Application name**.



Application name

Hello World

Company domain

android.example.com

Project location

/Users/tbove/AndroidStudioProjects/HelloWorld

Package name

com.example.android.helloworld

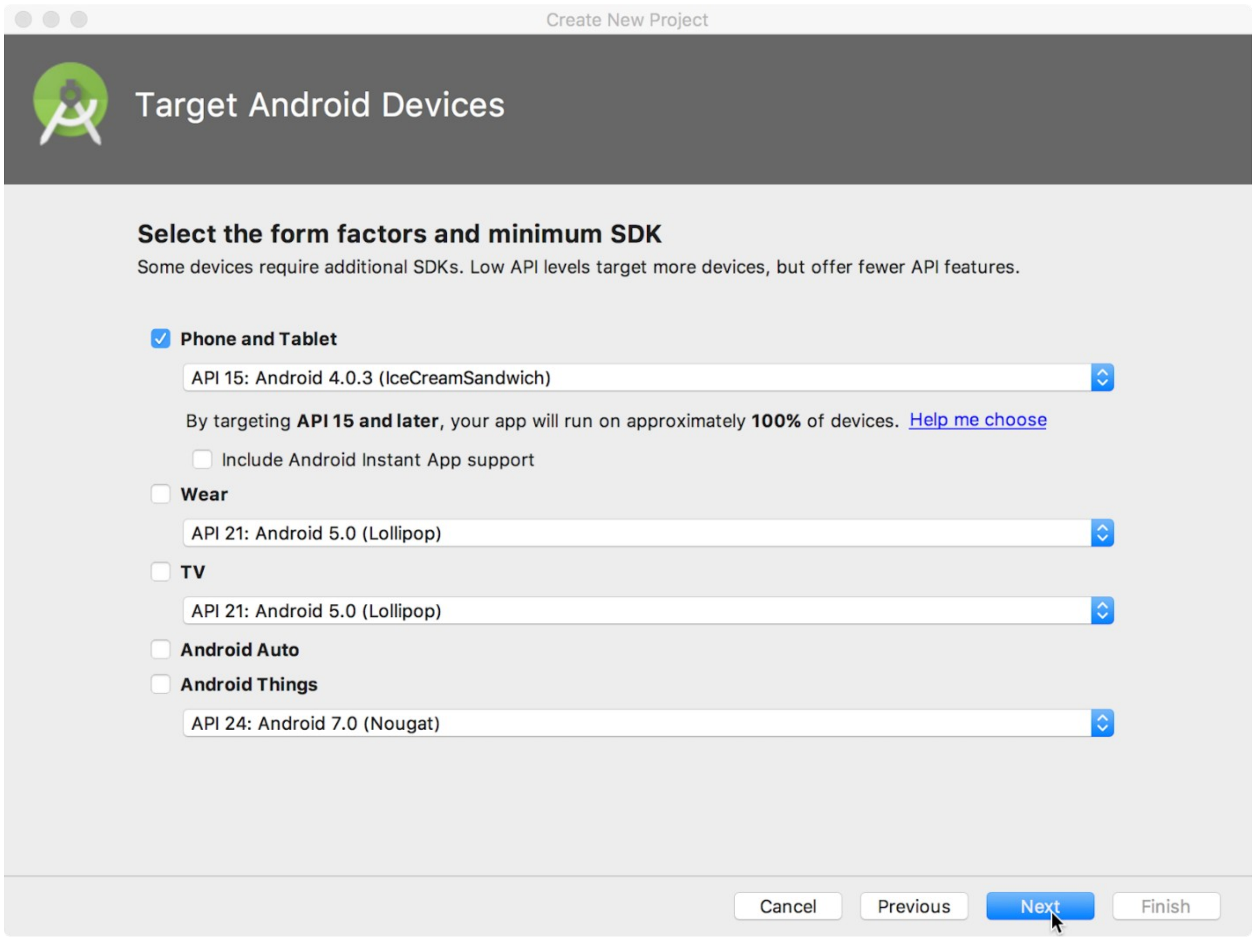
☐ Include C++ support
☐ Include Kotlin support

4. Verify that the default **Project location** is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory.
5. Accept the default **android.example.com** for **Company Domain**, or create a unique company domain.

   If you are not planning to publish your app, you can accept the default. Be aware that changing the package name of your app later is extra work.

6. Leave unchecked the options to **Include C++ support** and **Include Kotlin support**, and click **Next**.

7. On the **Target Android Devices** screen, **Phone and Tablet** should be selected. Ensure that **API 15: Android 4.0.3 IceCreamSandwich** is set as the Minimum SDK; if it is not, use the popup menu to set it.



Select the form factors and minimum SDK

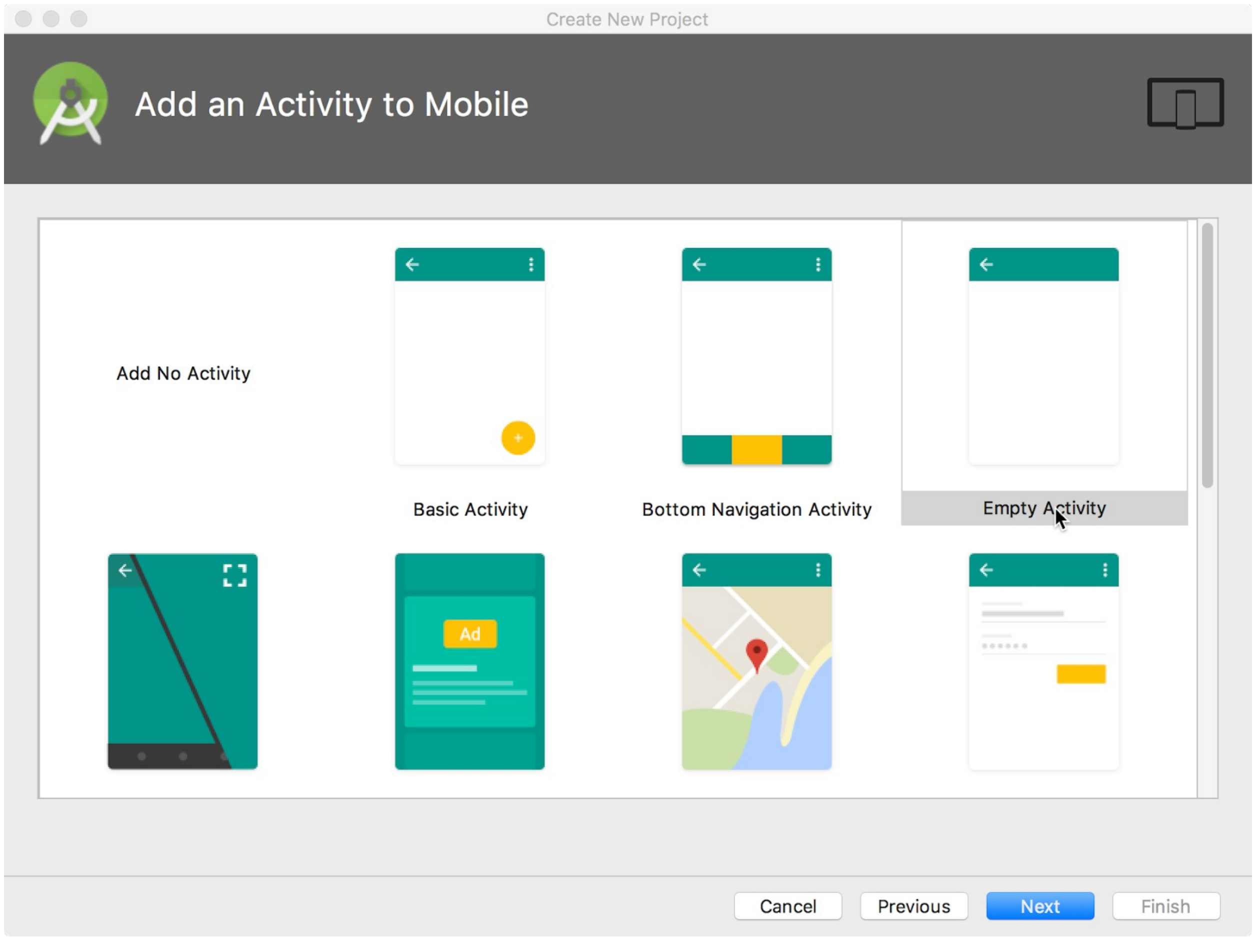
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

- [✓] Phone and Tablet
  - API 15: Android 4.0.3 (IceCreamSandwich)
  - By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose
  - [ ] Include Android Instant App support
- [ ] Wear
  - API 21: Android 5.0 (Lollipop)
- [ ] TV
  - API 21: Android 5.0 (Lollipop)
- [ ] Android Auto
- [ ] Android Things
  - API 24: Android 7.0 (Nougat)

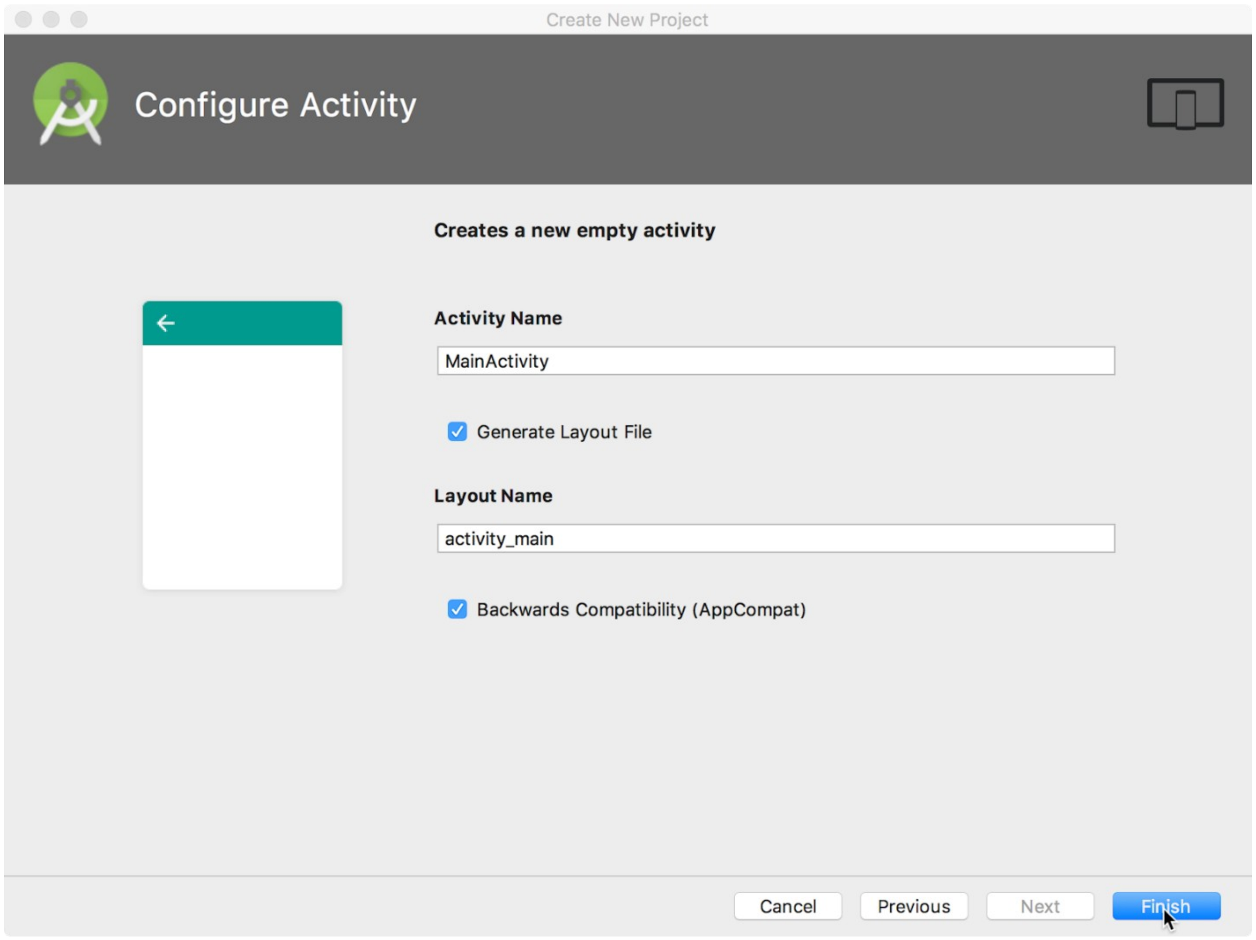
These are the settings used by the examples in the lessons for this course. As of this writing, these settings make your Hello World app compatible with 97% of Android devices active on the Google Play Store.

8. Leave unchecked the **Include Instant App support** and all other options. Then click **Next**. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.

9. The **Add an Activity** window appears. An Activity is a single, focused thing that the user can do. It is a crucial component of any Android app. An Activity typically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, choose **Empty Activity** as shown below, and click **Next**.



10. The **Configure Activity** screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activity provided by the template is named MainActivity. You can change this if you want, but this lesson uses MainActivity.



11. Make sure that the **Generate Layout file** option is checked. The layout name by default is activity_main. You can change this if you want, but this lesson uses activity_main.
12. Make sure that the **Backwards Compatibility (App Compat)** option is checked. This ensures that your app will be backwards-compatible with previous versions of Android.
13. Click **Finish**.

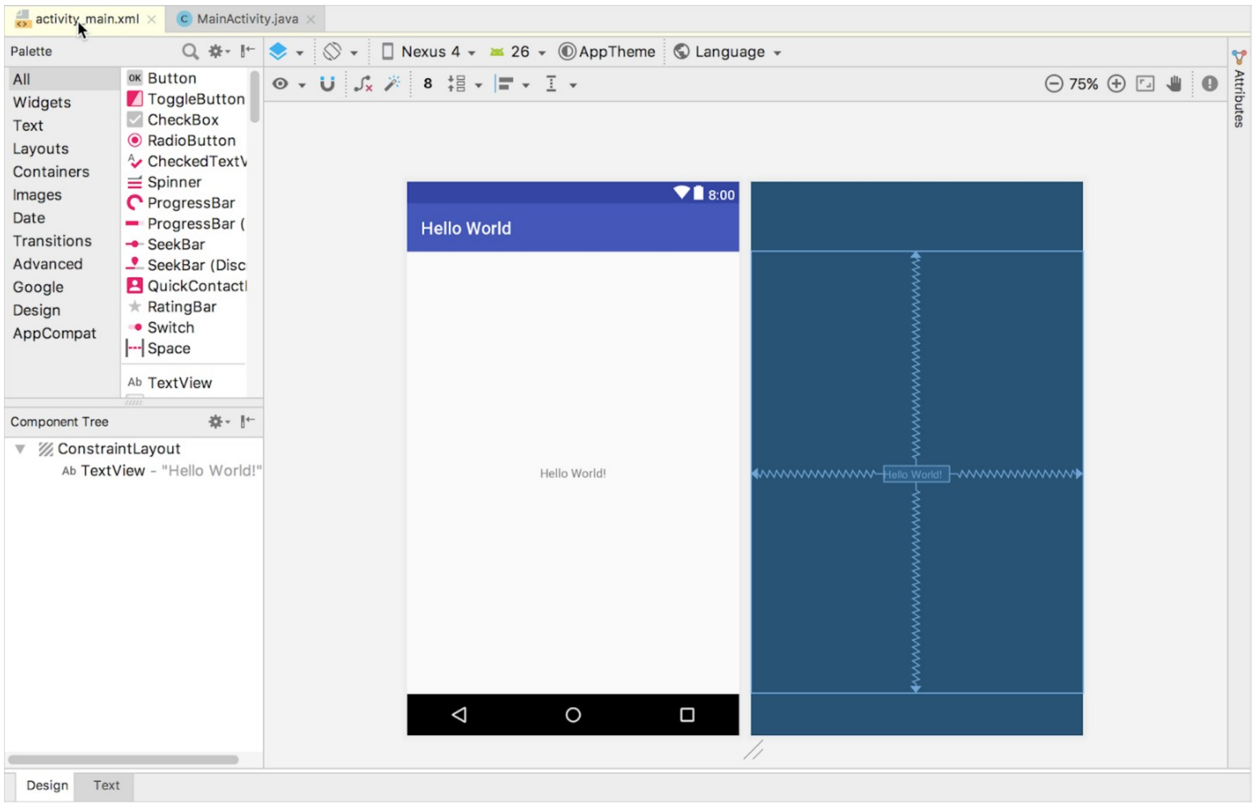
    Android Studio creates a folder for your projects, and builds the project with Gradle (this may take a few moments).

    Tip: See the Configure your build developer page for detailed information.

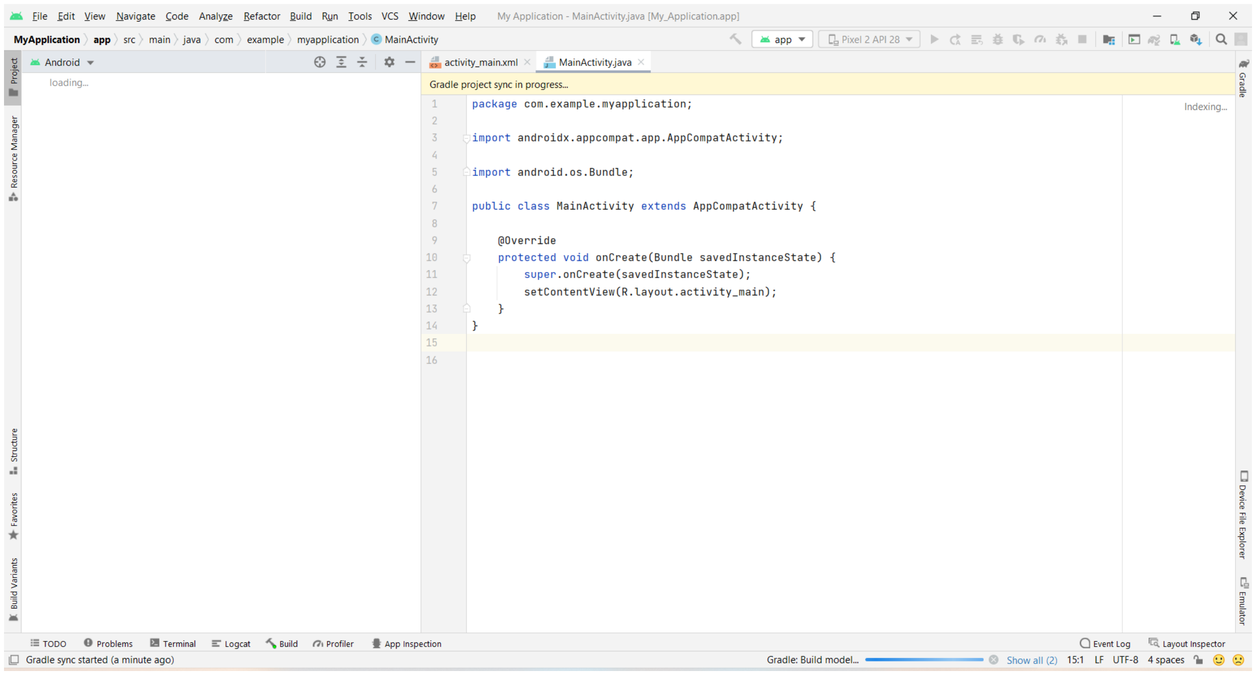
    You may also see a "Tip of the day" message with keyboard shortcuts and other useful tips. Click **Close** to close the message.

    The Android Studio editor appears. Follow these steps:

1. Click the **activity_main.xml** tab to see the layout editor.
2. Click the layout editor **Design** tab, if not already selected, to show a graphical rendition of the layout as shown below.



3. Click the **MainActivity.java** tab to see the code editor as shown below.



My Application - MainActivity.java [My_Application.app]

loading...

Gradle project sync in progress...

Indexing...

```java
package com.example.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);
    }
}
```

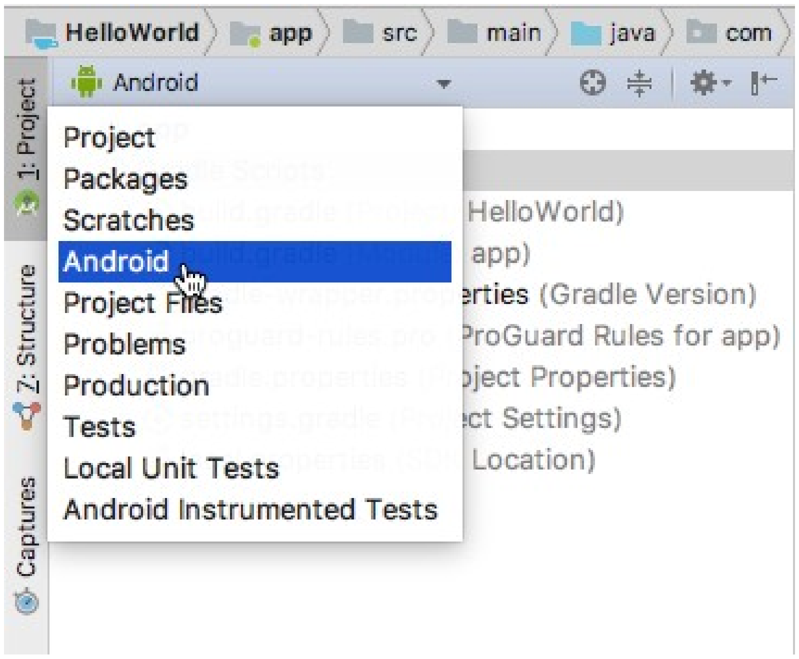
Gradle sync started (a minute ago)

Gradle: Build model...

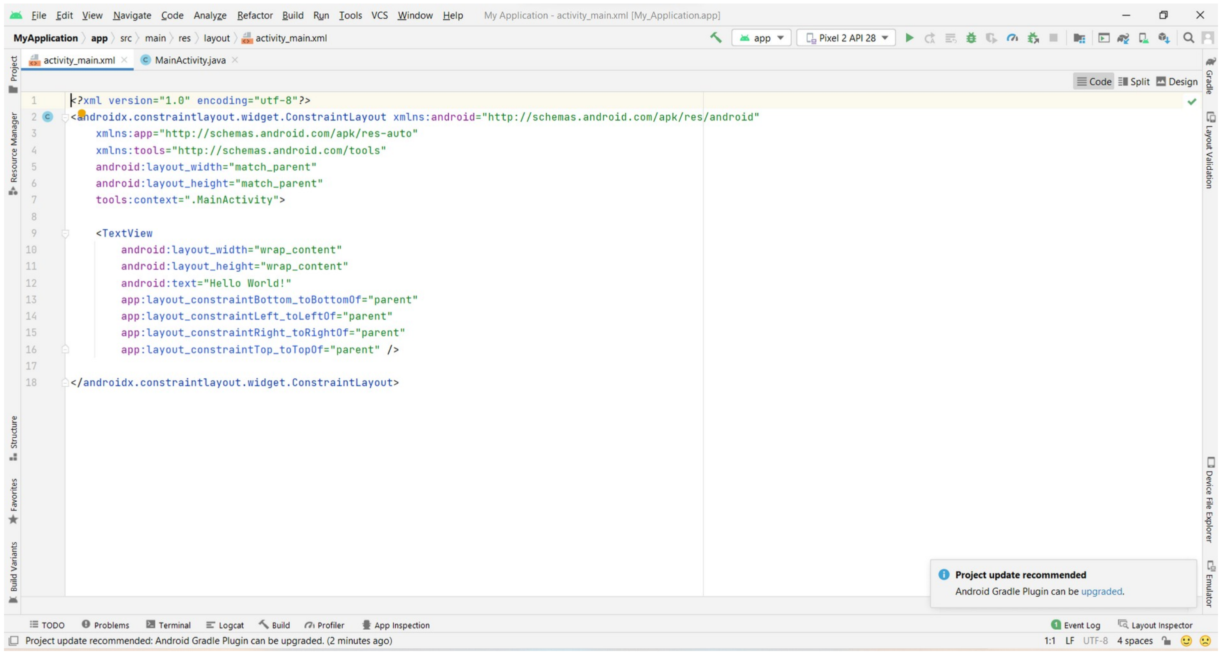
2.2 Explore the Project > Android pane

In this practical, you will explore how the project is organized in Android Studio.

1. If not already selected, click the **Project** tab in the vertical tab column on the left side of the Android Studio window. The Project pane appears.
2. To view the project in the standard Android project hierarchy, choose **Android** from the popup menu at the top of the Project pane, as shown below.



4. Click the **Activity_Main.xml** tab to see the code editor as shown below.



```
<?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:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Project update recommended
Android Gradle Plugin can be upgraded.

Project update recommended: Android Gradle Plugin can be upgraded. (2 minutes ago)

<u>CONCLUSION:</u>

Hence Android Studio is installed in the system successfully.

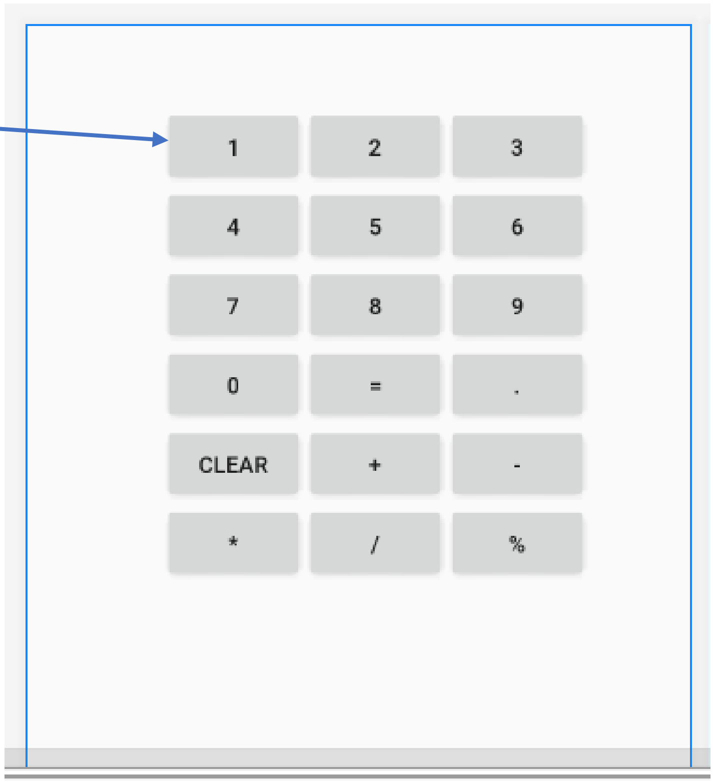
Practical 2:

Aim : Create an android Calculator app with Interactive User Interface using appropriate layouts.

Theory:

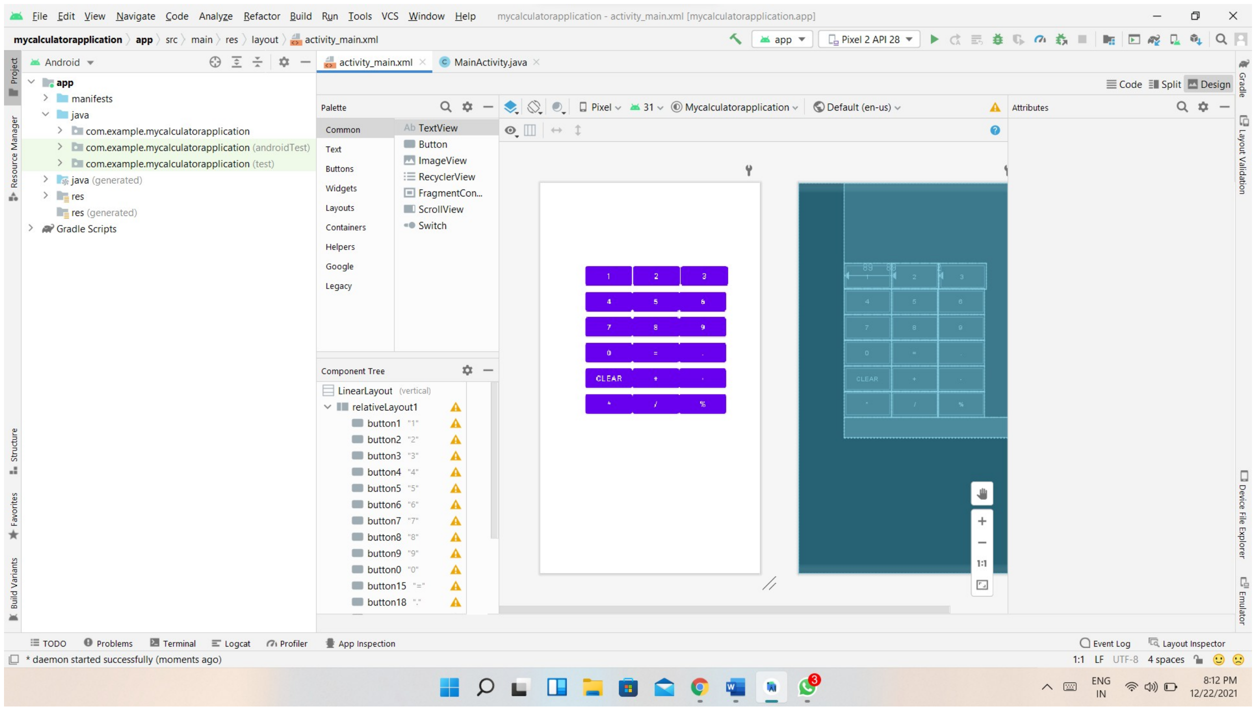
The user interface (UI) for an Android app is built as a hierarchy of layouts and widgets. The layouts are ViewGroup objects, containers that control how their child views are positioned on the screen. Widgets are View objects, UI components such as buttons and text boxes.Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, check box, zoom buttons, toggle buttons, and many more. Here, we will use Input Controls like Buttons,Text Fileds etc for a Calculator application with an Interactive User Interface.

bnutton 1



Layout used:

1. LinearLayout
2. RelativeLayout


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
mycalculatorapplication - activity_main.xml [mycalculatorapplication.app]
mycalculatorapplication › app › src › main › res › layout › activity_main.xml
activity_main.xml
MainActivity.java
Palette
Component Tree
LinearLayout (vertical)
relativeLayout1
button1 "1"
button2 "2"
button3 "3"
button4 "4"
button5 "5"
button6 "6"
button7 "7"
button8 "8"
button9 "9"
button0 "0"
button15 "="
button18 "."
CLEAR
Attributes
* daemon started successfully (moments ago)

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<RelativeLayout
android:id="@+id/relativeLayout1"
android:layout_width="312dp"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginLeft="35dp">

<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
```

```
android:layout_marginTop="150dp"
android:padding="20px"
android:text="1"></Button>

<Button
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button1"
android:layout_alignParentLeft="true"
android:layout_marginLeft="89dp"
android:layout_marginTop="0dp"
android:layout_toRightOf="@+id/button1"
android:padding="20px"
android:text="2"></Button>

<Button
android:id="@+id/button3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button2"
android:layout_marginLeft="2dp"
android:layout_marginTop="0dp"
android:layout_toRightOf="@+id/button2"
android:padding="20px"
android:text="3"></Button>

<Button
android:id="@+id/button4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/button1"
android:padding="20px"
android:text="4"></Button>
<Button
android:id="@+id/button5"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button4"
android:layout_toRightOf="@+id/button4"
android:padding="20px"
android:text="5"></Button>
<Button
android:id="@+id/button6"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button4"
android:layout_toRightOf="@+id/button5"
android:padding="20px"
android:text="6"></Button>
<Button
android:id="@+id/button7"
```

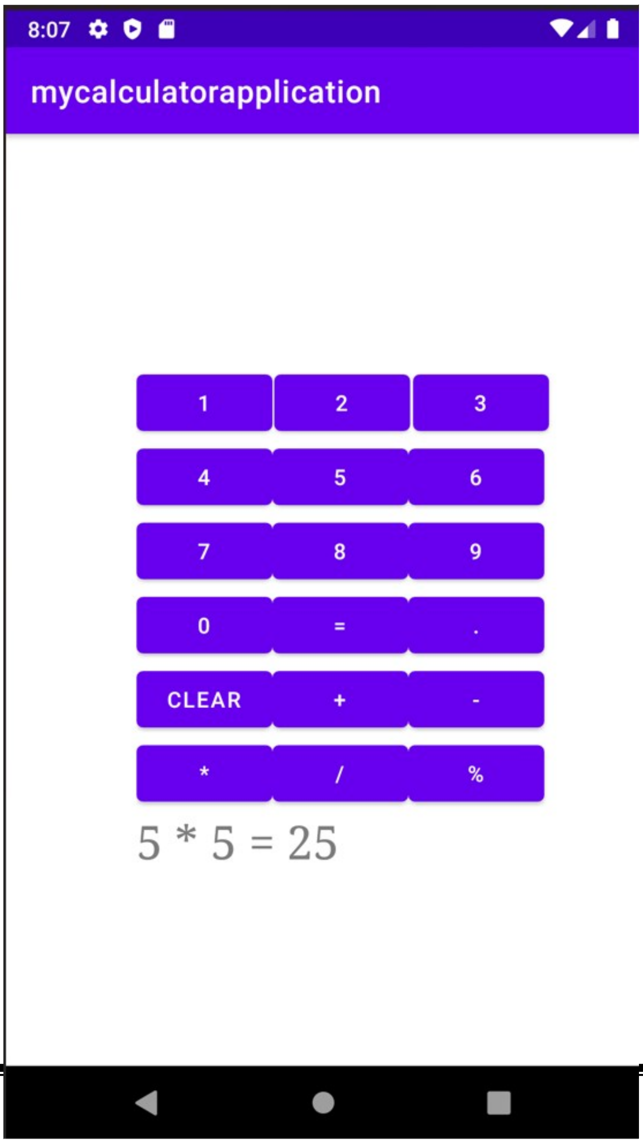
```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/button4"
android:padding="20px"
android:text="7"></Button>
<Button
android:id="@+id/button8"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button7"
android:layout_toRightOf="@+id/button7"
android:padding="20px"
android:text="8"></Button>
<Button
android:id="@+id/button9"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button7"
android:layout_toRightOf="@+id/button8"
android:padding="20px"
android:text="9"></Button>
<Button
android:id="@+id/button0"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/button7"
android:padding="20px"
android:text="0"></Button>
<Button
android:id="@+id/button15"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button0"
android:layout_toRightOf="@+id/button0"
android:padding="20px"
android:text="="></Button>
<Button
android:id="@+id/button18"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button0"
android:layout_toRightOf="@+id/button15"
android:padding="20px"
android:text="."></Button>
<Button
android:id="@+id/button16"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/button0"
android:padding="20px"
```

```
android:text="clear"></Button>
<Button
android:id="@+id/button11"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button16"
android:layout_toRightOf="@+id/button16"
android:padding="20px"
android:text="+"></Button>
<Button
android:id="@+id/button12"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button16"
android:layout_toRightOf="@+id/button11"
android:padding="20px"
android:text="-"></Button>
<Button
android:id="@+id/button13"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/button16"
android:padding="20px"
android:text="*"></Button>
<Button
android:id="@+id/button14"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button13"
android:layout_toRightOf="@+id/button13"
android:padding="20px"
android:text="/"></Button>
<Button
android:id="@+id/button17"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/button13"
android:layout_toRightOf="@+id/button14"
android:padding="20px"
android:text="%"></Button>
<TextView
android:id="@+id/TextView1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/button13"
android:fontFamily="serif"
android:textSize="30dp"></TextView>

</RelativeLayout>
</LinearLayout>
```

MainAtivity.java

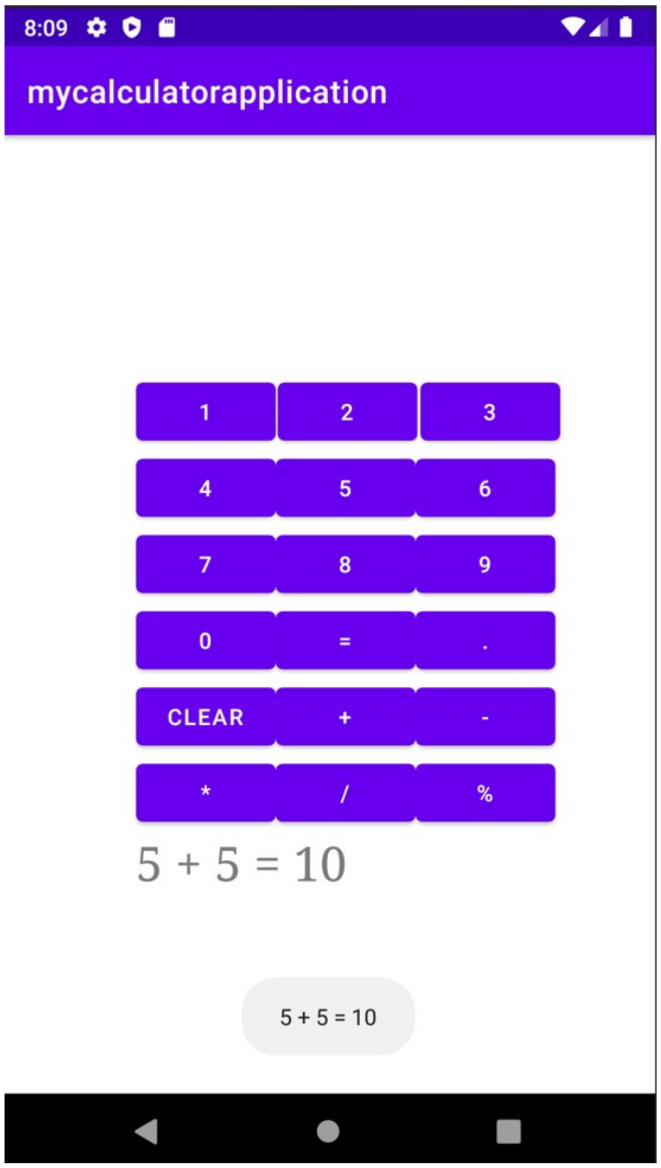
```
package com.example.mycalculatorapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
TextView tv1;
Integer c,d,r,b;
String a = "0", aa;
Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17,b18;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
tv1 = (TextView) findViewById(R.id.TextView1);
b1 = (Button) findViewById(R.id.button1);
b2 = (Button) findViewById(R.id.button2);
b3 = (Button) findViewById(R.id.button3);
b4 = (Button) findViewById(R.id.button4);
b5 = (Button) findViewById(R.id.button5);
b6 = (Button) findViewById(R.id.button6);
b7 = (Button) findViewById(R.id.button7);
b8 = (Button) findViewById(R.id.button8);
b9 = (Button) findViewById(R.id.button9);
b10 = (Button) findViewById(R.id.button0);
b11 = (Button) findViewById(R.id.button11);
b12 = (Button) findViewById(R.id.button12);
b13 = (Button) findViewById(R.id.button13);
b14= (Button) findViewById(R.id.button14);
b15= (Button) findViewById(R.id.button15);
b16= (Button) findViewById(R.id.button16);
b17= (Button) findViewById(R.id.button17);
b18= (Button) findViewById(R.id.button18);

b1.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "1";
tv1.setText(a);
            }
        });
b2.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
```

```
a = tv1.getText().toString();
a = a + "2";
tv1.setText(a);
            }
        });
b3.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "3";
tv1.setText(a);
            }
        });
b4.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "4";
tv1.setText(a);
            }
        });
b5.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "5";
tv1.setText(a);
            }
        });
b6.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "6";
tv1.setText(a);
            }
        });
b7.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "7";
tv1.setText(a);
            }
        });
b8.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "8";
tv1.setText(a);
            }
        });
b9.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "9";
tv1.setText(a);
            }
        });
b10.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
a = tv1.getText().toString();
a = a + "0";
tv1.setText(a);
            }
        });
b11.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
aa = a;
b = 1;
a = "";
//tv1.setText(+);
tv1.setText("");
            }
        });
b12.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
aa = a;
b = 2;
a = "";
//tv1.setText(a);
tv1.setText("-");
tv1.setText("");
            }
        });
b13.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
aa = a;
b = 3;
a = "";
//tv1.setText(a);
tv1.setText("*");
tv1.setText("");
            }
        });
b14.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
aa = a;
b = 4;
a = "";
//tv1.setText(a);
tv1.setText("/");
tv1.setText("");
            }
        });
//OnClickevent handling for =
b15.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
if (b == 1){
c = Integer.parseInt(aa);
d = Integer.parseInt(a);
r = c + d;
            } else if (b == 4){
c = Integer.parseInt(aa);
d = Integer.parseInt(a);
r = c/d;
            } else if (b == 2){
c = Integer.parseInt(aa);
d = Integer.parseInt(a);
r = c - d;
            } else if (b == 3) {
c = Integer.parseInt(aa);
d = Integer.parseInt(a);
r = c * d;
            }
String op ="";
switch (b) {
case 1: op = " + ";
break;
case 2: op = " - ";
break;
case 3: op = " * ";
break;
case 4: op = " / ";
            }
```



5 * 5 = 25

```
tv1.setText(c + op + d + " = " + r);
String ans=c + op + d + " = " + r;
Toast.makeText(MainActivity.this, ans,
Toast.LENGTH_LONG).show();

        }
      });
b16.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
tv1.setText("");
        }
      });

   }

}
```



5 + 5 = 10

5 + 5 = 10

CONCLUSION:

Hence an android Calculator app with Interactive User Interface using appropriate layouts is implemented successfully

Practical 3:

AIM: Create an android app that demonstrates working with TextView Elements.

Theory : TextView in Android is one of the basic and important UI elements. This plays a very important role in the UI experience and depends on how the information is displayed to the user. This TextView widget in android can be dynamized in various contexts. For example, if the important part of the information is to be highlighted then the substring that contains, it is to be italicized or it has to be made bold, one more scenario is where if the information in TextView contains a hyperlink that directs to a particular web URL then it has to be spanned with hyperlink and has to be underlined. Have a look at the following list and image to get an idea of the overall discussion.

1. **Formatting the TextView**
2. **Size of the TextView**
3. **Changing Text Style**
4. **Changing the Text Color**
5. **Text Shadow**
6. **Letter Spacing and All Caps**
7. **Adding Icons for TextView**
8. **HTML Formatting of the TextView**

Step by Step Implementation

Step 1: Create an Empty Activity Project

- Create an empty activity Android Studio Project. Refer to Android | How to Create/Start a New Project in Android Studio? to know how To Create an empty activity Android Studio project.

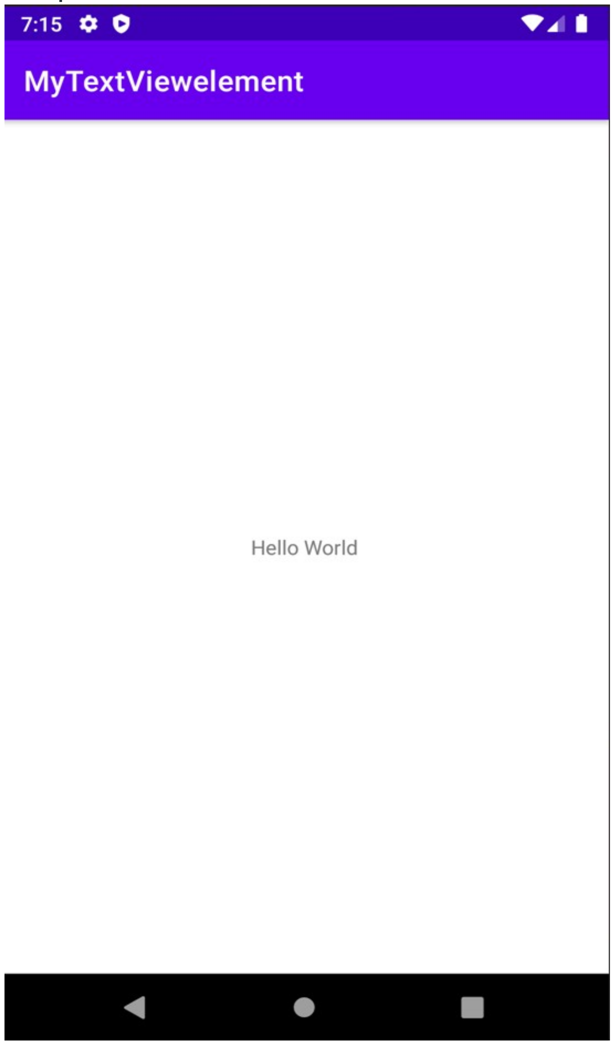
Step 2: Working with the activity_main.xml file

- The main layout and one, which includes only a TextView and as varied as we go on discuss the various contexts.
- To implement the UI of the activity invoke the following code inside the activity_main.xml file

Code:

```
<?xmlversion="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"
    tools:ignore="HardcodedText">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

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

.output:



1. Formatting the TextView

Android offers mainly 3 types of typefaces

- *normal*
- *sans*
- *serif*
- *monospace*

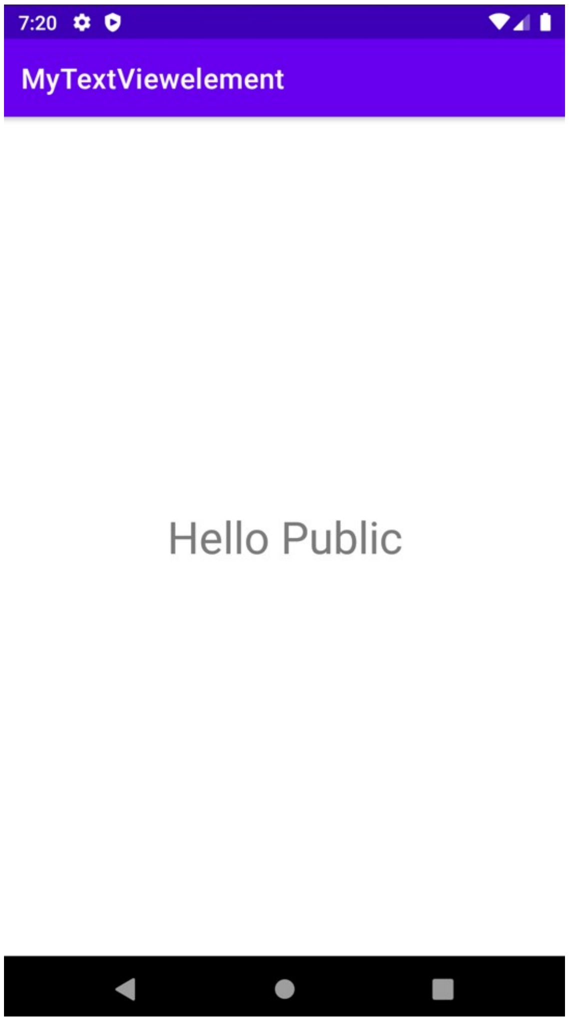
- The above four types of faces are to be invoked under the "**typeFace"** attribute of the TextView in XML.
- Invoke the following code and note the "**typeFace"** attribute of the TextView.

Code:

```
<?xmlversion="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"
    tools:ignore="HardcodedText">

    <!--the below typeFace attribute has to
         invoked with values mentioned-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Public"
        android:textSize="32sp"

        android:typeface="normal"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
```

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

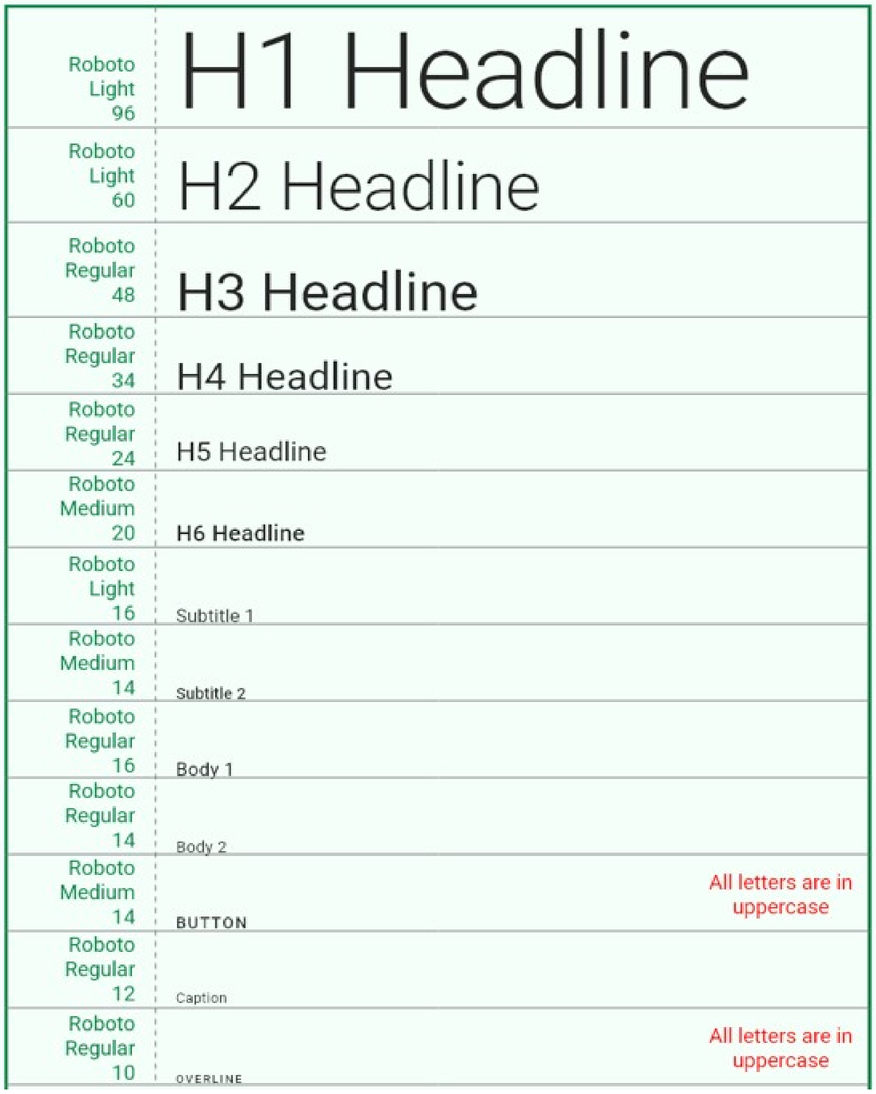
Output:



Hello Public

2. Size of the TextView

- This feature of the Text view upholds what type of content has to be shown to the user. For example, if there is a Heading, there are 6 types of heading that can be implemented have a look at the following image which contains the guidelines for the size of the text view and style of the text view which is recommended by Google's Material Design.



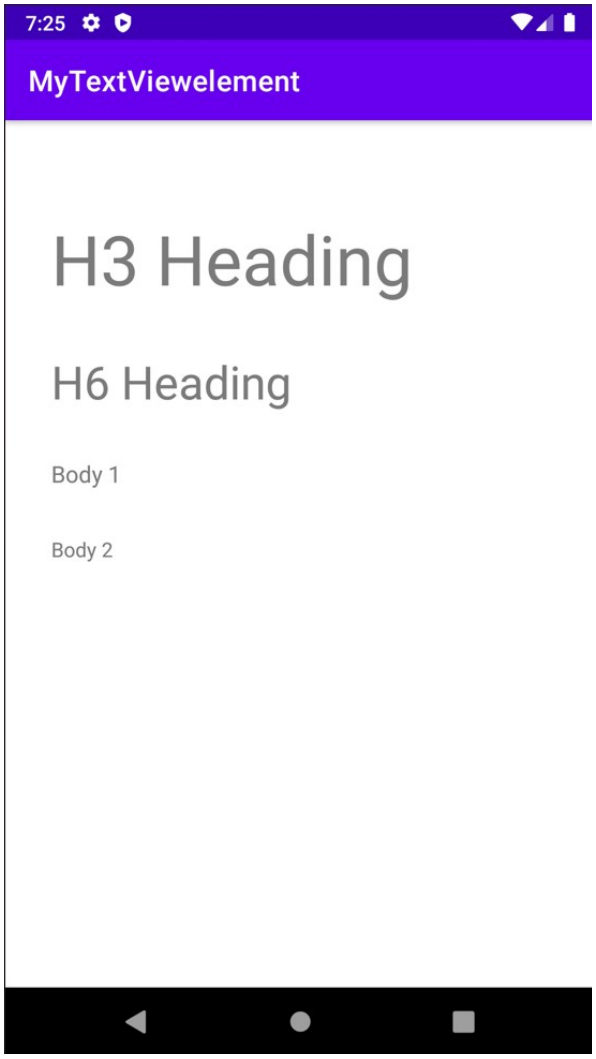
Roboto Light 96	H1 Headline	
Roboto Light 60	H2 Headline	
Roboto Regular 48	H3 Headline	
Roboto Regular 34	H4 Headline	
Roboto Regular 24	H5 Headline	
Roboto Medium 20	H6 Headline	
Roboto Light 16	Subtitle 1	
Roboto Medium 14	Subtitle 2	
Roboto Regular 16	Body 1	
Roboto Regular 14	Body 2	
Roboto Medium 14	BUTTON	All letters are in uppercase
Roboto Regular 12	Caption	
Roboto Regular 10	OVERLINE	All letters are in uppercase

- The attribute which is used to change the size of the Text View in android is **"textSize".**
- Refer to the following code and its output for better understanding.

Code:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
    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:orientation="vertical"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="64dp"

        android:textSize="48sp"

        android:text="H3 Heading"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"

        android:textSize="32sp"

        android:text="H6 Heading"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"

        android:textSize="16sp"

        android:text="Body 1"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"

        android:textSize="14sp"

        android:text="Body 2"/>

</LinearLayout>
```

Output:



```
MyTextViewelement

H3 Heading

H6 Heading

Body 1

Body 2
```

3. Changing Text Style

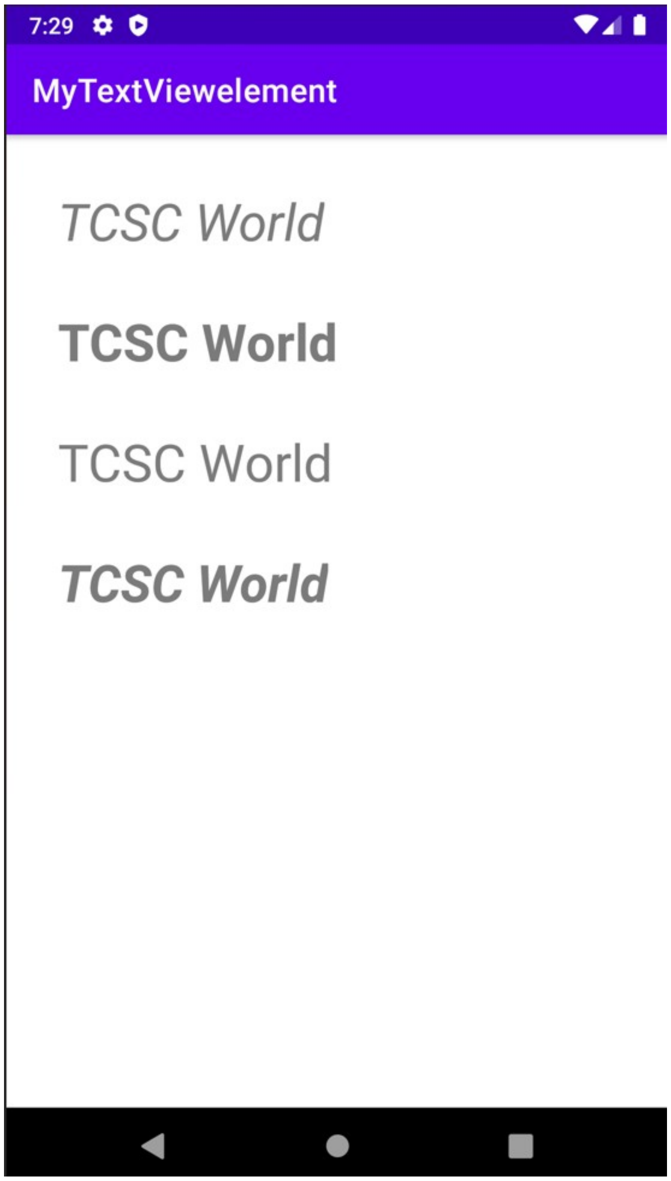
In Android there are basically three text styles:

- *Bold*
- *Italic*
- *Normal*
- The text style of the text in android can be implemented using the attribute **“textStyle”.**
- Multiple text styles can also be implemented using the pipeline operator. Example **“android:textStyle=”bold|italic”.**
- To implement the various text styles refer to the following code and its output.

Code:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
    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=".MainActivity"
    tools:ignore="HardcodedText">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"
        android:orientation="vertical">

        <!--the below textStyle attribute has to
            invoked with values mentioned-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TCSC World

            android:textStyle="italic"

            android:textSize="32sp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="32dp"
            android:text="TCSC World"

            android:textStyle="bold"

            android:textSize="32sp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="32dp"
            android:text="TCSC World"

            android:textStyle="normal"

            android:textSize="32sp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="32dp"
            android:text="TCSC World"

            android:textStyle="bold|italic"

            android:textSize="32sp"/>

    </LinearLayout>

</LinearLayout>
```

Output:



TCSC World

TCSC World

TCSC World

TCSC World

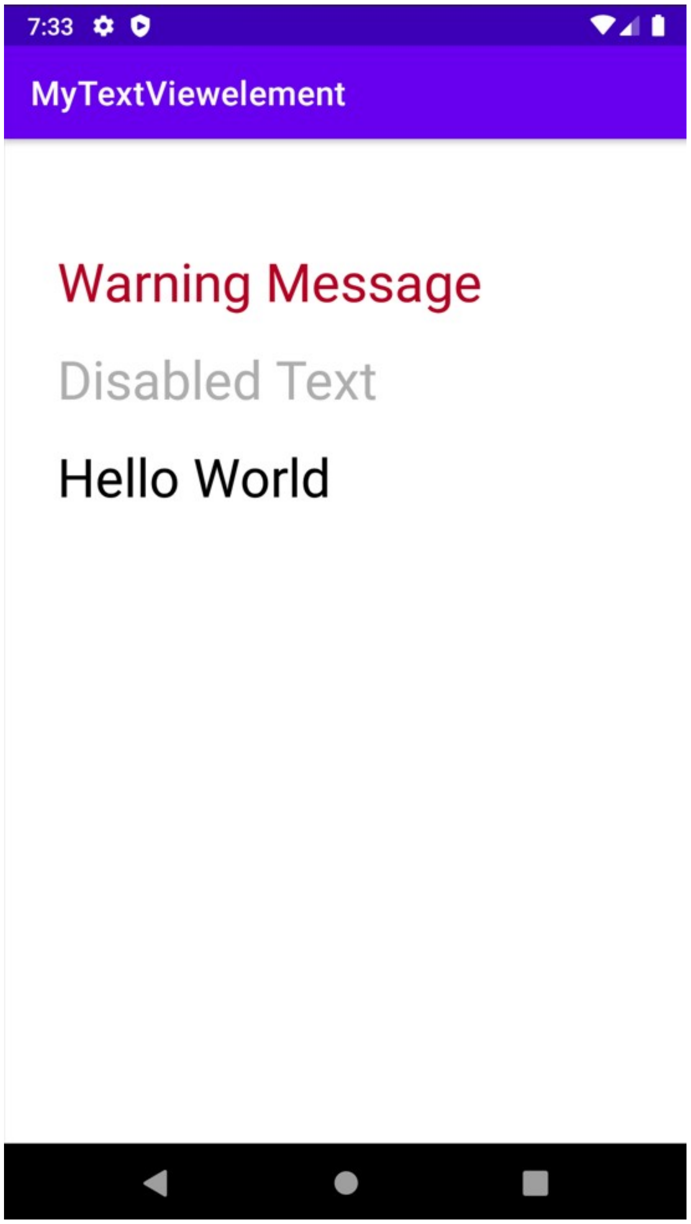
4. Changing the Text Color

- The color of the text should also change according to the change in the context of the information displayed to the user.
- For example, if there is warning text it must be in the red color and for disabled text, the opacity or the text color should be grayish. To change the color of the text, the attribute **“textColor”** is used.
- Android also offers the predefined text colors, which can be implemented using **“@android:color/yourColor”** as value for the **“textColor”**. Here the value may be hex code or the predefined colors offered by the android.
- Refer to the following code and its output for better understanding.

Code:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
    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:orientation="vertical"
    tools:context=".MainActivity"
    tools:ignore="HardcodedText">

    <!--the value predefined by android-->
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="64dp"
        android:text="Warning Message"
        android:textColor="#B00020"
        android:textSize="32sp"/>

    <!--the value predefined by android-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="16dp"
        android:text="Disabled Text"

        android:textColor="@android:color/darker_gray"

        android:textSize="32sp"/>
    <!--the value is hex code-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="16dp"
        android:text="Hello World"

        android:textColor="#000000"

        android:textSize="32sp"/>

</LinearLayout>
```

Output:



Warning Message

Disabled Text

Hello World

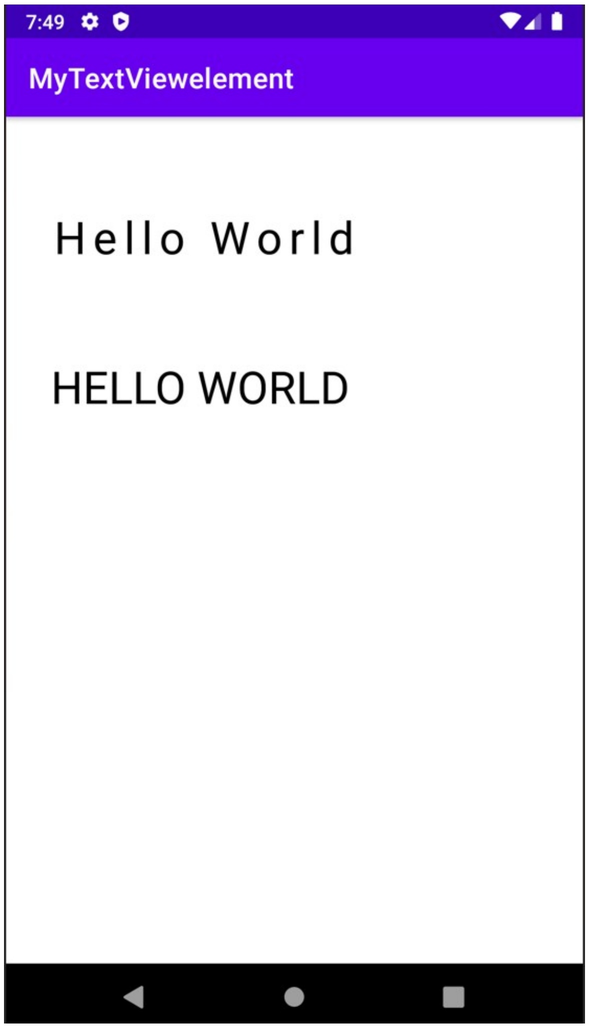
5. Letter Spacing and All Caps

- Letter spacing and capital letters are some of the important properties of the text View in android.
- For the text of buttons and tab layouts, the text should be in uppercase letters recommended by Google Material Design.
- The letter spacing also should be maintained according to the scenario.
- Refer to the following code and its output for better understanding.

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        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:orientation="vertical"
        tools:context=".MainActivity"
        tools:ignore="HardcodedText">

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="32dp"
                android:layout_marginTop="64dp"
                android:letterSpacing="0.15"
                android:text="GeeksforGeeks"
                android:textColor="@android:color/black"
                android:textSize="32sp" />

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="32dp"
                android:layout_marginTop="64dp"
                android:text="GeeksforGeeks"
                android:textAllCaps="true"
                android:textColor="@android:color/black"
                android:textSize="32sp" />

</LinearLayout>
```

Output:



Hello World

HELLO WORLD

<u>CONCLUSION:</u>

Hence TextView is implemented successfully

Practical 4:

Aim : Create an android app that demonstrates Activity Lifecycle and Instance State.

Theory : Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. Theandroid Activity is the subclass of ContextThemeWrapper class.An activity is the single screen in android. It is like window or frame of Java.By the help of activity, you can place all your UI components or widgets in a single screen.aq

activity_main.xml

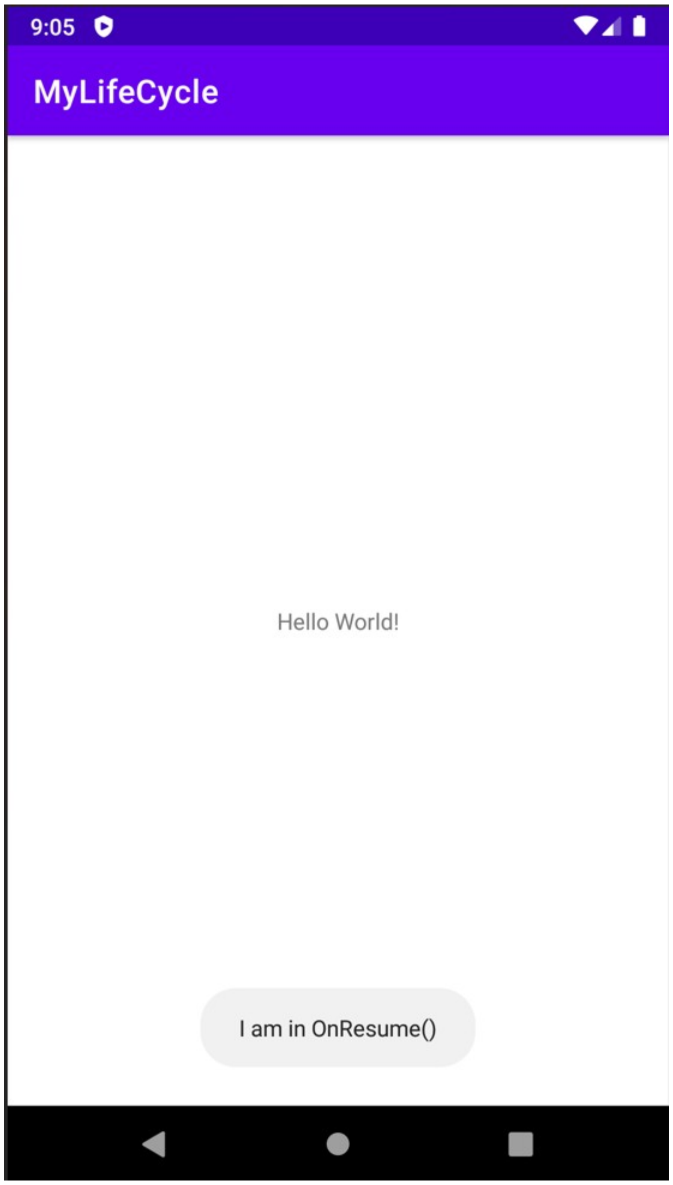
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayoutxmlns:android="http://
schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />

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

MainActivity.java

```
package com.example.mylifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

public class MainActivityextends AppCompatActivity{

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Log.d("lifecycle", "TCSC Message: OnCreate invoked");
    }
protected void onStart() {
super.onStart();
Log.d("Lifecycle", "TCSC Message: OnStart invoked");
Toast.makeText(this, "I am in OnStart()", Toast.LENGTH_LONG).show();
    }
protected void onResume() {
super.onResume();
Toast.makeText(this, "I am in OnResume()", Toast.LENGTH_LONG).show();
Log.d("Lifecycle", "TCSC Message: OnResume invoked");
    }
//4431 Saurabh Pal
protected void onPause() {
super.onPause();
Log.d("Lifecycle", "TCSC Message: OnPause invoked");
    }
protected void onStop() {
super.onStop();
Log.d("Lifecycle", "TCSC Message: OnStop invoked");
    }
protected void onRestart() {
super.onRestart();
Log.d("Lifecycle", "TCSC Message: OnRestart invoked");
    }
protected void onDestroy() {
super.onDestroy();
Log.d("Lifecycle", "TCSC Message: OnDestroy invoked");
    }
}
```

Output:

```
2021-12-2220:42:43.428 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onCreate invoked

2021-12-2220:42:43.440 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onStart invoked

2021-12-2220:42:43.459 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onResume invoked

2021-12-2220:42:52.373 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onPause invoked

2021-12-2220:42:52.384 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onStop invoked

2021-12-2220:42:57.536 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onRestart invoked

2021-12-2221:02:57.537 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onStart invoked

2021-12-2221:03:57.540 6152-6152/com.example.myapplication D/lifecycle: TCSC MEssage: onResume invoked
```



<u>CONCLUSION:</u> **Hence an android app that demonstrates Activity Lifecycle and Instance State implemented successfully.**

Practical 5:

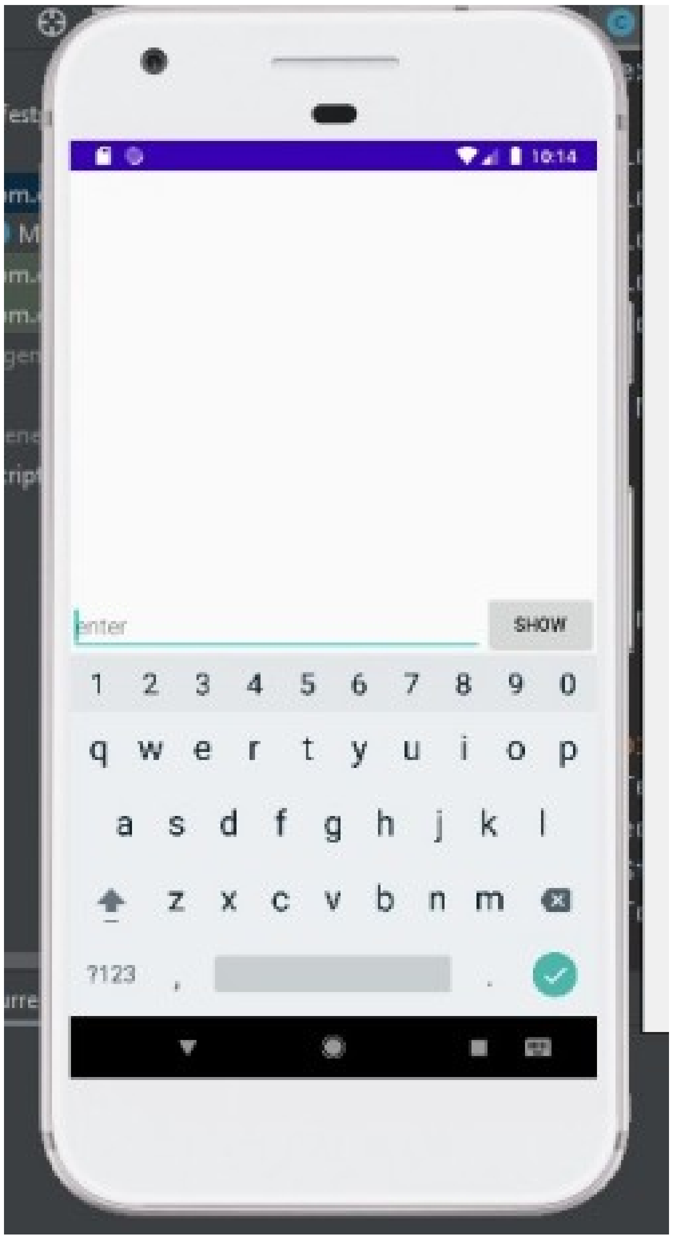
AIM: **Create an android app that demonstrates the use of Keyboards, Input Controls, Alerts, and Pickers.**

Theory:

Droid Cafe created from the Basic Activity template, lets a user tap image buttons to launch an activity, and choose a single delivery option from radio-button choices for a food order. Choices are shown in toast messages

Activity_main.xml

```
<RelativeLayout
 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="com.example.keyboard.MainActivity">
 <Button
 android:id="@+id/button_main"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentBottom="true"
 android:layout_alignParentRight="true"
 android:onClick="showText"
 android:text="show" />
 <EditText
 android:id="@+id/editText_main"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_alignParentBottom="true"
 android:inputType="textPassword"
 android:layout_toLeftOf="@id/button_main"
 android:hint="enter" />
</RelativeLayout>
```



MainActivity.java

```
package com.example.keyboard;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
```

```
}
public void showText(View view) {
```

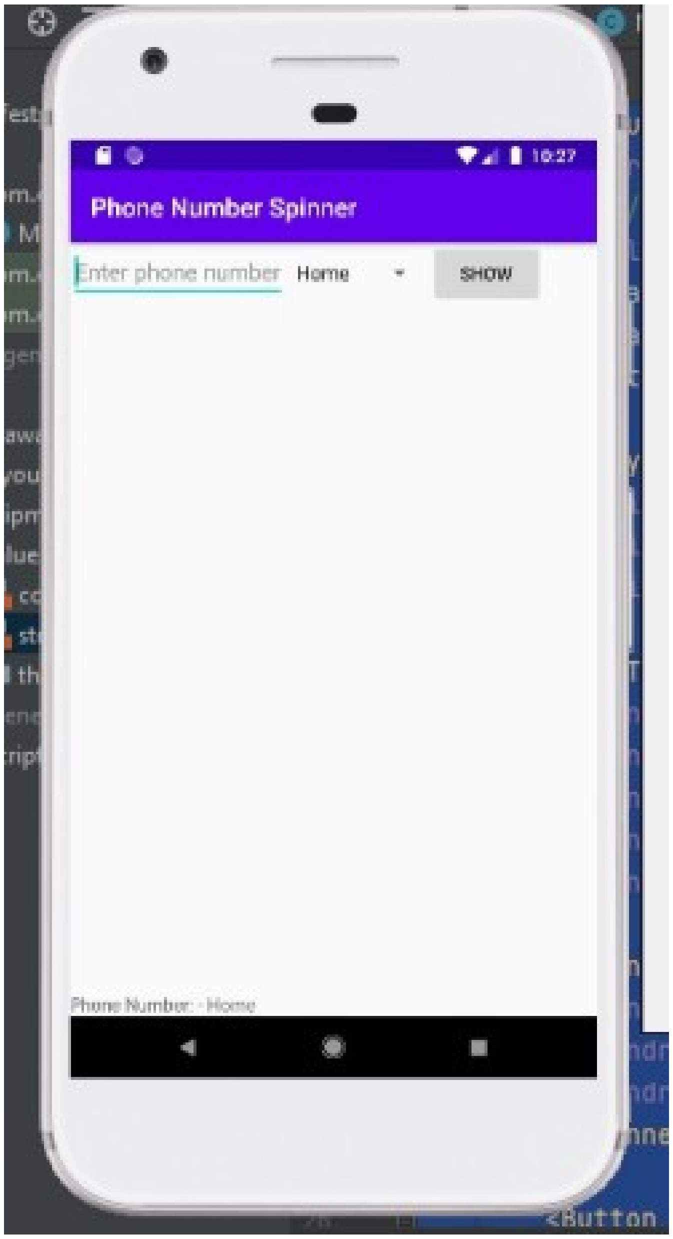
```
EditText editText = (EditText) findViewById(R.id.editText_main);
if (editText != null) {
String showString = editText.getText().toString();
Toast.makeText(this, showString, Toast.LENGTH_SHORT).show();
}
}
}
```

Input Controls

Activity_main.xml

```
<RelativeLayout
 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="com.example.inputcontrols.MainActivity">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal">

 <EditText
 android:id="@+id/editText_main"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:inputType="phone"
 android:hint="@string/hint_phonenumber" />

 <Spinner
 android:id="@+id/label_spinner"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content">
 </Spinner>

 <Button
 android:id="@+id/button_main"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="showText"
 android:text="@string/show_button" />

 </LinearLayout>
 <LinearLayout
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:layout_alignParentBottom="true">

 <TextView
 android:id="@+id/title_phonelabel"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/phonenumber_label"/>

 <TextView
 android:id="@+id/text_phonelabel"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/nothing_entered"/>
```

```
</LinearLayout>
</RelativeLayout>
```



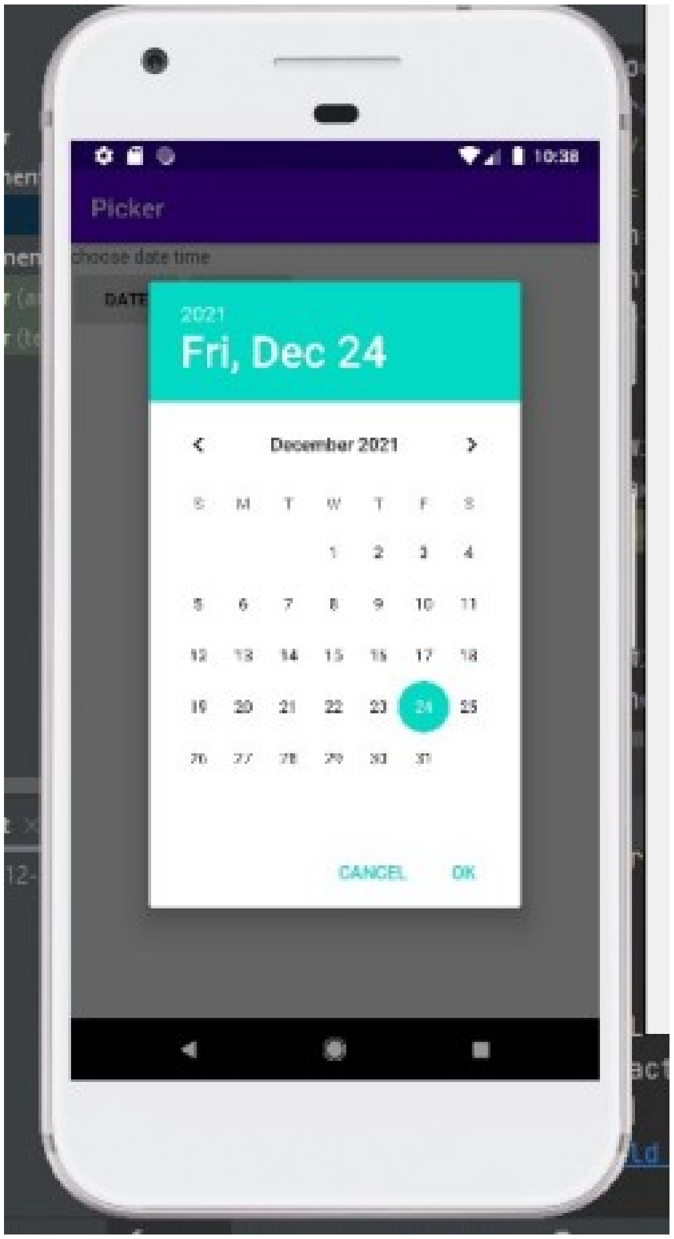
MainActivity.java

```
package com.example.inputcontrols;

import
android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

 private static final String TAG = MainActivity.class.getSimpleName();
 private String mSpinnerLabel = "";
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Spinner spinner = (Spinner) findViewById(R.id.label_spinner);
 if (spinner != null) {
 spinner.setOnItemSelectedListener(this);
 }
 ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
R.array.labels_array, android.R.layout.simple_spinner_item);
 adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  if
(spinner != null) {
 spinner.setAdapter(adapter);
 }
 }
 public void showText(View view) {
 EditText editText = (EditText) findViewById(R.id.editText_main);
 if (editText != null) {
 String showString = (editText.getText().toString() + " - " + mSpinnerLabel);
Toast.makeText(this, showString, Toast.LENGTH_SHORT).show();
 TextView textView = (TextView)findViewById(R.id.text_phonelabel);
textView.setText(showString);
 }
 }
 @Override
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
mSpinnerLabel = adapterView.getItemAtPosition(i).toString();
 showText(view);
 }
 @Override
 public void onNothingSelected(AdapterView<?> adapterView) {
 Log.d(TAG, getString(R.string.nothing_selected));
 }
}
```

Date Time Picker

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.picker.MainActivity">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="choose date time"/>

 <RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/button_date"
 android:text="date"
 android:onClick="showDatePickerDialog"/>
 <Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/button_time"
 android:layout_alignBottom="@id/button_date"
 android:layout_toRightOf="@id/button_date"
 android:text="time"
 android:onClick="showTimePickerDialog"/>
 </RelativeLayout>
</LinearLayout>
```


10:38
Picker
choose date time
DATE
2021
Fri, Dec 24
December 2021
S M T W T F S
CANCEL
OK

```
package com.example.picker;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

 public void showDatePickerDialog(View view) {
```

```
 DialogFragment newFragment = new DatePickerFragment();
 newFragment.show(getSupportFragmentManager(),
 getString(R.string.date_picker));
 }

 public void showTimePickerDialog(View view) {
 DialogFragment newFragment = new TimePickerFragment();
 newFragment.show(getSupportFragmentManager(),
 getString(R.string.time_picker));
 }

 public void processDatePickerResult(int year, int month, int day) {
 String month_string = Integer.toString(month+1);
 String day_string = Integer.toString(day);
 String year_string = Integer.toString(year);
 String dateMessage = (month_string + "/" +
 day_string + "/" + year_string);
 Toast.makeText(this, getString(R.string.date) + dateMessage,
 Toast.LENGTH_SHORT).show();
 }

 public void processTimePickerResult(int hourOfDay, int minute)
{
 String hour_string = Integer.toString(hourOfDay);
 String minute_string = Integer.toString(minute);
 String timeMessage = (hour_string + ":" + minute_string);
 Toast.makeText(this, getString(R.string.time) + timeMessage,
 Toast.LENGTH_SHORT).show();
 }
}
```

DatePickerFragment.java

```
package com.example.picker;

import android.app.Dialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.text.format.DateFormat;
import android.support.v4.app.DialogFragment;
import android.app.TimePickerDialog;
import android.widget.TimePicker;
import java.util.Calendar;
/**
 * A simple {@link Fragment} subclass.
 */
public class TimePickerFragment extends DialogFragment
 implements TimePickerDialog.OnTimeSetListener {

 @NonNull
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState)
 {
 // Use the current time as the default values for the picker.
 final Calendar c = Calendar.getInstance();
 int hour = c.get(Calendar.HOUR_OF_DAY);
 int minute = c.get(Calendar.MINUTE);
```

```
 // Create a new instance of TimePickerDialog and return it.
 return new TimePickerDialog(getActivity(), this, hour, minute,
 DateFormat.is24HourFormat(getActivity()));
 }

 public void onTimeSet(TimePicker view, int hourOfDay, int minute)
 {
 // Do something with the time chosen by the user.
 // Set the activity to the Main Activity.
 MainActivity activity = (MainActivity) getActivity();
 // Invoke Main Activity's processTimePickerResult() method.
 activity.processTimePickerResult(hourOfDay, minute);
 }
 }
```

TimePickerFragment.java

```
package com.example.picker;
```

```
import android.app.Dialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.text.format.DateFormat;
import android.support.v4.app.DialogFragment;
import android.app.TimePickerDialog;
import android.widget.TimePicker;
import java.util.Calendar;
/**
 * A simple {@link Fragment} subclass.
 */
public class TimePickerFragment extends DialogFragment
 implements TimePickerDialog.OnTimeSetListener {

 @NonNull
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState) {
 // Use the current time as the default values for the picker.
 final Calendar c = Calendar.getInstance();
 int hour = c.get(Calendar.HOUR_OF_DAY);
 int minute = c.get(Calendar.MINUTE);

 // Create a new instance of TimePickerDialog and return it.
 return new TimePickerDialog(getActivity(), this, hour, minute,
 DateFormat.is24HourFormat(getActivity()));
 }

 public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  //
Do something with the time chosen by the user.
 // Set the activity to the Main Activity.
 MainActivity activity = (MainActivity) getActivity();
 // Invoke Main Activity's processTimePickerResult() method.
```

```
 activity.processTimePickerResult(hourOfDay, minute);
 }
}
```

CONCLUSION: **Hence an app that demonstrates the use of Keyboard, Input Controls, Alerts, and Pickers.(e.g. Droid café) is implemented successfully.**

Practical 6:

Aim : Create an android app that demonstrates the use of an Options Menu.

Theory : **The options menu is the primary collection of menu items for an activity. It's where actions that have a global impact on the app, such as "Search," "Compose email," and "Settings”.**

Mainactivity.java

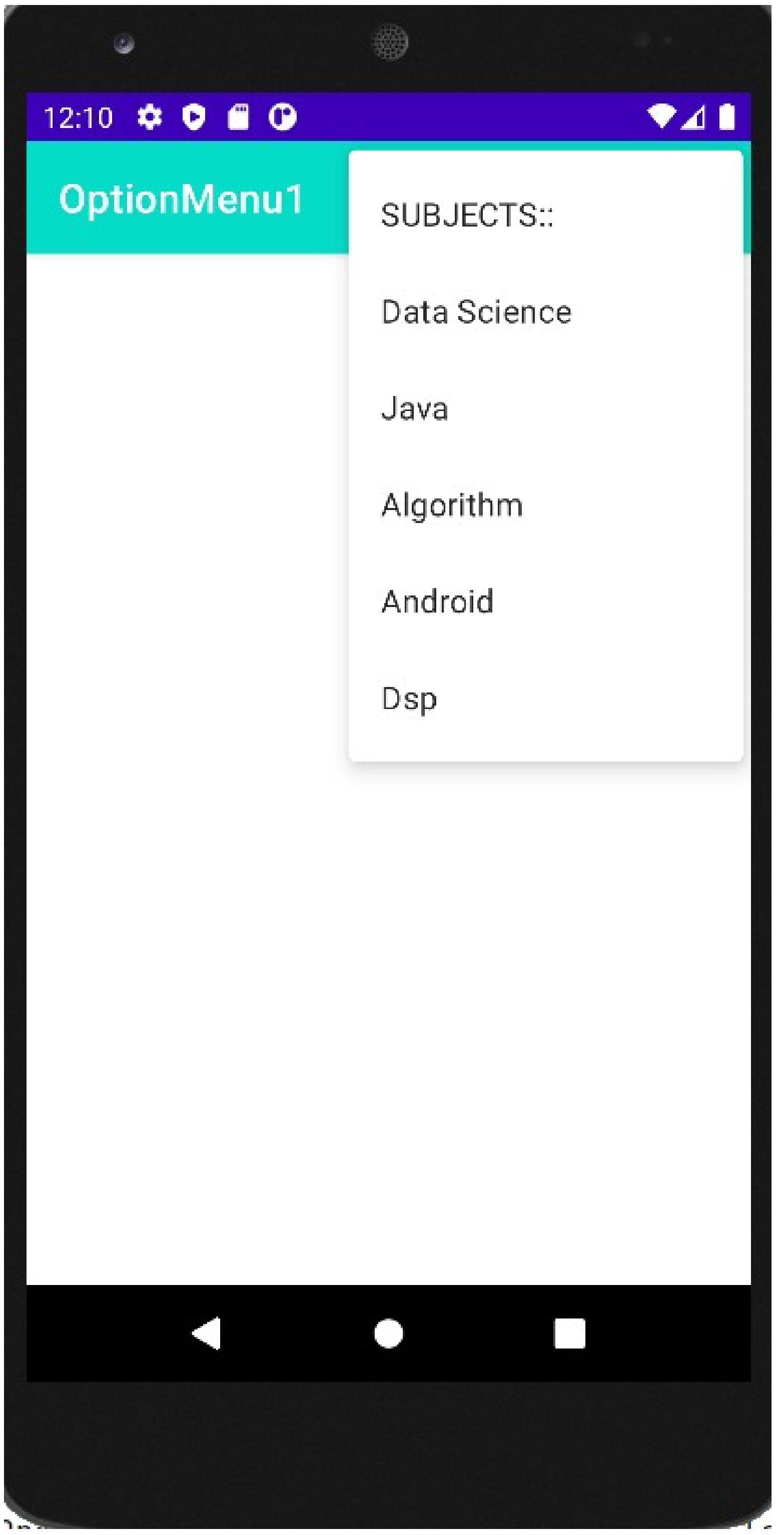
```
package com.example.optionmenu1;

import android.os.Bundle;

import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;

import android.view.View;

import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.example.optionmenu1.databinding.ActivityMainBinding;

import android.view.Menu;
import android.view.MenuItem;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MainActivityextends AppCompatActivity{

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Toolbar toolbar= findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
    }

@Override
public booleanonCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
   }

@Override
public booleanonOptionsItemSelected(MenuItemitem) {
// Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();

switch (id) {
case R.id.item1:
Toast.makeText(getApplicationContext(), "Your favorite Subjects ",
Toast.LENGTH_LONG).show();
return true;
case R.id.item2:
Toast.makeText(getApplicationContext(), "You like Data Science ",
Toast.LENGTH_LONG).show();
return true;
case R.id.item3:
Toast.makeText(getApplicationContext(), "Is Java is your favourite",
Toast.LENGTH_LONG).show();
return true;
case R.id.item4:
Toast.makeText(getApplicationContext(), "You selected Algorithm  ",
Toast.LENGTH_LONG).show();
return true;
case R.id.item5:
Toast.makeText(getApplicationContext(), " Android is favorite ",
Toast.LENGTH_LONG).show();
return true;
case R.id.item6:
Toast.makeText(getApplicationContext(), " Dsp selected ",
Toast.LENGTH_LONG).show();
return true;



default:
return super.onOptionsItemSelected(item);
        }

    }
}
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayoutxmlns: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"
app:layout_behavior="@string/appbar_scrolling_view_behavior">

<fragment
android:id="@+id/nav_host_fragment_content_main"
android:name="androidx.navigation.fragment.NavHostFragment"
android:layout_width="0dp"
android:layout_height="0dp"
app:defaultNavHost="true"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintHorizontal_bias="1.0"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.0"
app:navGraph="@navigation/nav_graph" />
</androidx.constraintlayout.widget.ConstraintLayout>
menu_main.xml

<menu 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"
tools:context="com.example.optionmenu1.MainActivity">
<item  android:id="@+id/item1"
android:title="SUBJECTS::"/>
<item  android:id="@+id/item2"
android:title="Data Science"/>
<item  android:id="@+id/item3"
android:title="Java"/>
<item  android:id="@+id/item4"
android:title="Algorithm"/>
<item  android:id="@+id/item5"
android:title="Android"/>
<item  android:id="@+id/item6"
android:title="Dsp"/>




</menu>



activity_main.xml

<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayoutxmlns: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">

<com.google.android.material.appbar.AppBarLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:theme="@style/Theme.OptionMenu1.AppBarOverlay">
<androidx.appcompat.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/teal_200"
app:popupTheme="@style/Theme.OptionMenu1.PopupOverlay"/>

</com.google.android.material.appbar.AppBarLayout>
<!--<include layout="@layout/content_main" /> -->

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="WELCOME" />


</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Output-


12:10
OptionMenu1
SUBJECTS::
Data Science
Java
Algorithm
Android
Dsp

CONCLUSION:

Hence Option Menu is implemented successfully

Practical 7:

Aim : Create an android app that demonstrate Screen Navigation Using the App Bar and Tabs.

<u>Theory</u> - We all have come across apps that have a Bottom Navigation Bar. Some popular examples include Instagram, WhatsApp, etc. In this article, let's learn how to implement such a functional Bottom Navigation Bar in the Android Studio

Basics

Typically, an implementation of tabs in Android consists of:

1. Swipe views
2. Tabs UI element

These are two independent navigation patterns, but they can be combined with each other.

In general: swipe views *can* be combined with tabs, and tabs be combined with swipe views. However, tabs benefit tremendously from being combined with swipe views, as explained below.

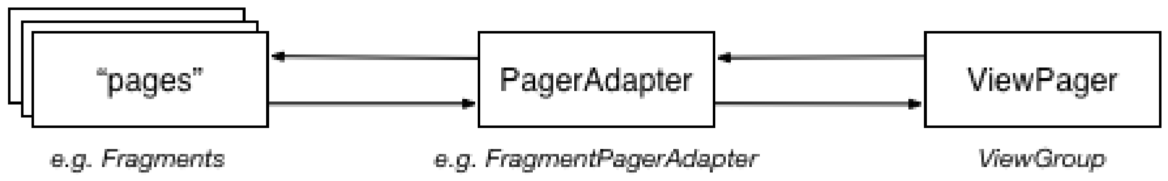
Swipe Views

Swipe views allow to flip through a set of "pages" by swiping horizontally on the screen.

Combined with tabs, this allows the user to switch to the next or previous tab by just swiping anywhere on the screen, rather than having to click on the tab itself.

- Swipe views are implemented by the ViewPager ViewGroup (declared in the activity's layout XML)
- A page is typically implemented as a Fragment
- A PagerAdapter supplies the ViewPager with the pages (fragments) to display. In the case of using fragments as pages, this PagerAdapter is a FragmentPagerAdapter or FragmentStatePagerAdapter

The swipe views architecture is illustrated below:



TabLayout Tabs

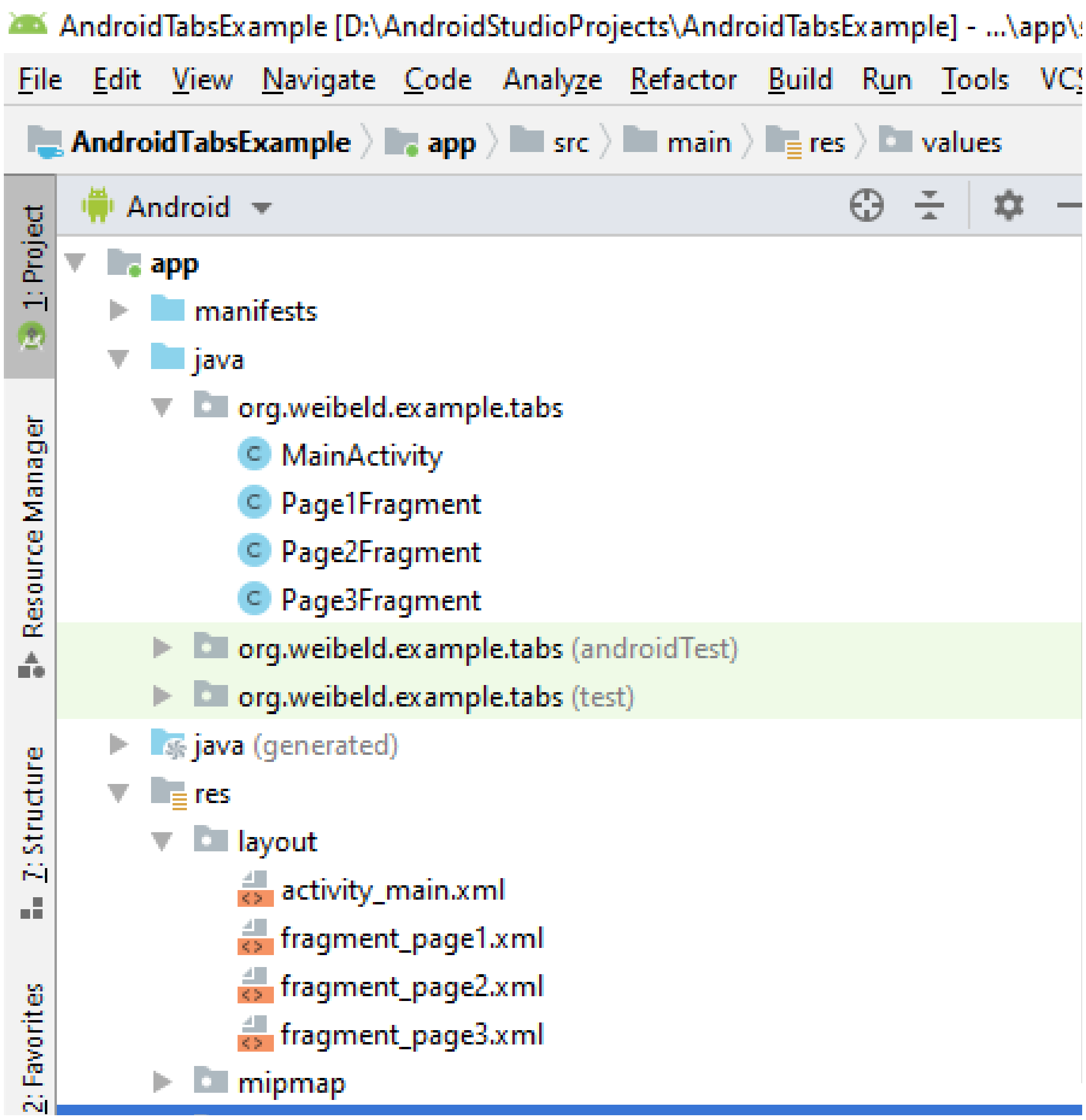
This is the preferred approach as it's easier to implement than ActionBar tabs and does not rely on an ActionBar.

TabLayout tabs are especially easy to implement if they are used in combination with a formerly implemented ViewPager (see Swipe Views), because then the tabs can be automatically populated by the ViewPager (i.e. no need to create and add the individual tabs manually). This is shown in the following:

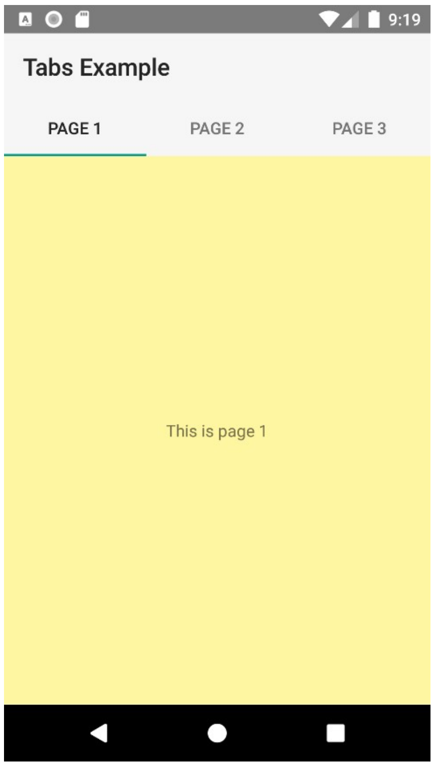
1. In the activity's layout XML, add a TabLayout element above the ViewPager element (and below the Toolbar element, if a Toolbar is used)
2. In the activity's onCreate method, call setupWithViewPager(ViewPager) on the TabLayout to populate and integrate the TabLayout with the ViewPager (requires that getPageTitle of the PagerAdapter is overriden)

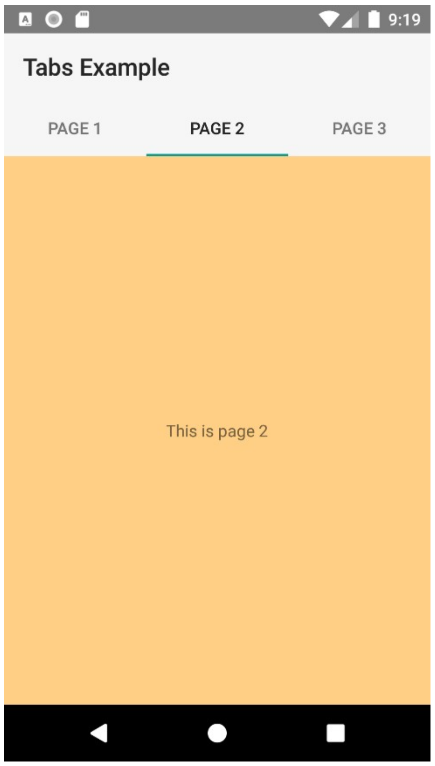
Note: to use TabLayout, the *Design Support Library* must be added to the project. In the module build.gradle file, do:

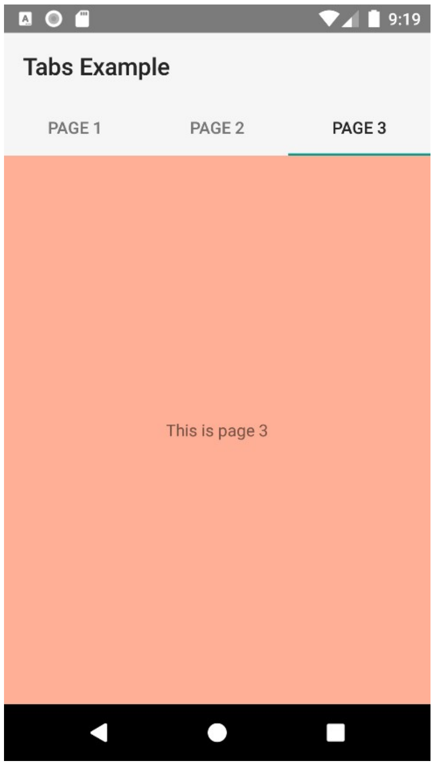
```
dependencies {
   compile 'com.android.support:support-v13:24.2.1'
}
```


AndroidTabsExample [D:\AndroidStudioProjects\AndroidTabsExample] - ...\app\
File Edit View Navigate Code Analyze Refactor Build Run Tools VC
AndroidTabsExample 〉 app 〉 src 〉 main 〉 res 〉 values
Android
1: Project
Resource Manager
Z: Structure
2: Favorites

```
app
  manifests
  java
    org.weibeld.example.tabs
      MainActivity
      Page1Fragment
      Page2Fragment
      Page3Fragment
    org.weibeld.example.tabs (androidTest)
    org.weibeld.example.tabs (test)
  java (generated)
  res
    layout
      activity_main.xml
      fragment_page1.xml
      fragment_page2.xml
      fragment_page3.xml
    mipmap
```







<u>activity_main.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/a
pk/res/android"

xmlns:tools="http://schemas.android.com/too
ls"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

tools:context="org.weibeld.example.tabs.Mai
nActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"

android:layout_height="wrap_content"

android:minHeight="?attr/actionBarSize"

android:background="?attr/colorPrimary"
        />


<android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"

android:layout_height="wrap_content"
        android:layout_width="match_parent"
```

```
android:background="?attr/colorPrimary"
        />

    <android.support.v4.view.ViewPager

xmlns:android="http://schemas.android.com/a
pk/res/android"
        android:id="@+id/viewpager"
        android:layout_width="match_parent"

android:layout_height="match_parent"
        />

</LinearLayout>
```

fragment_page1.xml

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout

xmlns:android="http://schemas.android.com/a
pk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/pageColor1">

    <TextView
        android:layout_width="wrap_content"

android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="This is page 1" />

</FrameLayout>
```

<u>fragment_page2.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout

xmlns:android="http://schemas.android.com/a
pk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/pageColor2">

    <TextView
        android:layout_width="wrap_content"

android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="This is page 2" />

</FrameLayout>
```

<u>fragment_page3.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout

xmlns:android="http://schemas.android.com/a
```

```
pk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/pageColor3">

    <TextView
        android:layout_width="wrap_content"

android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="This is page 3" />

</FrameLayout>
```

Page1Fragment.java

```
package org.weibeld.example.tabs;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import org.weibeld.example.R;

/* Fragment used as page 1 */
public class Page1Fragment extends Fragment
{

    @Override
    public View onCreateView(LayoutInflater
inflater, ViewGroup container, Bundle
savedInstanceState) {
        View rootView =
inflater.inflate(R.layout.fragment_page1,
container, false);
        return rootView;
    }
}
```

Page2Fragment.java

```
package org.weibeld.example.tabs;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import org.weibeld.example.R;

/* Fragment used as page 2 */
public class Page2Fragment extends Fragment
{

    @Override
    public View onCreateView(LayoutInflater
inflater, ViewGroup container, Bundle
savedInstanceState) {
        View rootView =
inflater.inflate(R.layout.fragment_page2,
container, false);
        return rootView;
    }

}
```

Page3Fragment.java

```
package org.weibeld.example.tabs;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import org.weibeld.example.R;

/* Fragment used as page 3 */
public class Page3Fragment extends Fragment
{

    @Override
    public View onCreateView(LayoutInflater
inflater, ViewGroup container, Bundle
savedInstanceState) {
        View rootView =
inflater.inflate(R.layout.fragment_page3,
container, false);
        return rootView;
    }
}
```

MainActivity.java

```
package org.weibeld.example.tabs;

import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import
android.support.design.widget.TabLayout;
import
android.support.v13.app.FragmentPagerAdapte
r;
import android.support.v4.view.ViewPager;
import
android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

import org.weibeld.example.R;

public class MainActivity extends
AppCompatActivity {

    private final String LOG_TAG =
MainActivity.class.getSimpleName();

    // Titles of the individual pages
(displayed in tabs)
    private final String[] PAGE_TITLES =
new String[] {
            "Page 1",
            "Page 2",
            "Page 3"
    };

    // The fragments that are used as the
individual pages
    private final Fragment[] PAGES = new
Fragment[] {
            new Page1Fragment(),
            new Page2Fragment(),
            new Page3Fragment()
    };

    // The ViewPager is responsible for
sliding pages (fragments) in and out upon
user input
    private ViewPager mViewPager;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

        // Set the Toolbar as the
activity's app bar (instead of the default
ActionBar)
        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);

        // Connect the ViewPager to our
custom PagerAdapter. The PagerAdapter
supplies the pages
        // (fragments) to the ViewPager,
which the ViewPager needs to display.
        mViewPager = (ViewPager)
findViewById(R.id.viewpager);
        mViewPager.setAdapter(new
MyPagerAdapter(getFragmentManager()));

        // Connect the tabs with the
ViewPager (the setupWithViewPager method
does this for us in
        // both directions, i.e. when a new
tab is selected, the ViewPager switches to
this page,
        // and when the ViewPager switches
to a new page, the corresponding tab is
selected)
        TabLayout tabLayout = (TabLayout)
findViewById(R.id.tab_layout);

tabLayout.setupWithViewPager(mViewPager);
    }


    /* PagerAdapter for supplying the
ViewPager with the pages (fragments) to
display. */
    public class MyPagerAdapter extends
```

```
FragmentPagerAdapter {

        public
MyPagerAdapter(FragmentManager
fragmentManager) {
            super(fragmentManager);
        }

        @Override
        public Fragment getItem(int
position) {
            return PAGES[position];
        }

        @Override
        public int getCount() {
            return PAGES.length;
        }

        @Override
        public CharSequence
getPageTitle(int position) {
            return PAGE_TITLES[position];
        }

    }
}
```

CONCLUSION:

Hence Navigation Bar is implemented successfully.

Practical 8:

<u>AIM:</u> Create an android app to Connect to the Internet and use BroadcastReceiver.

Theory : **Android apps can send or receive broadcast messages from the Android system and other Android apps.These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging. Here we will check for internet connectivity.**

BROADCAST RECEIVER

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/andro
id"
    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:text="Airplane mode of
BroadcastReceiver"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

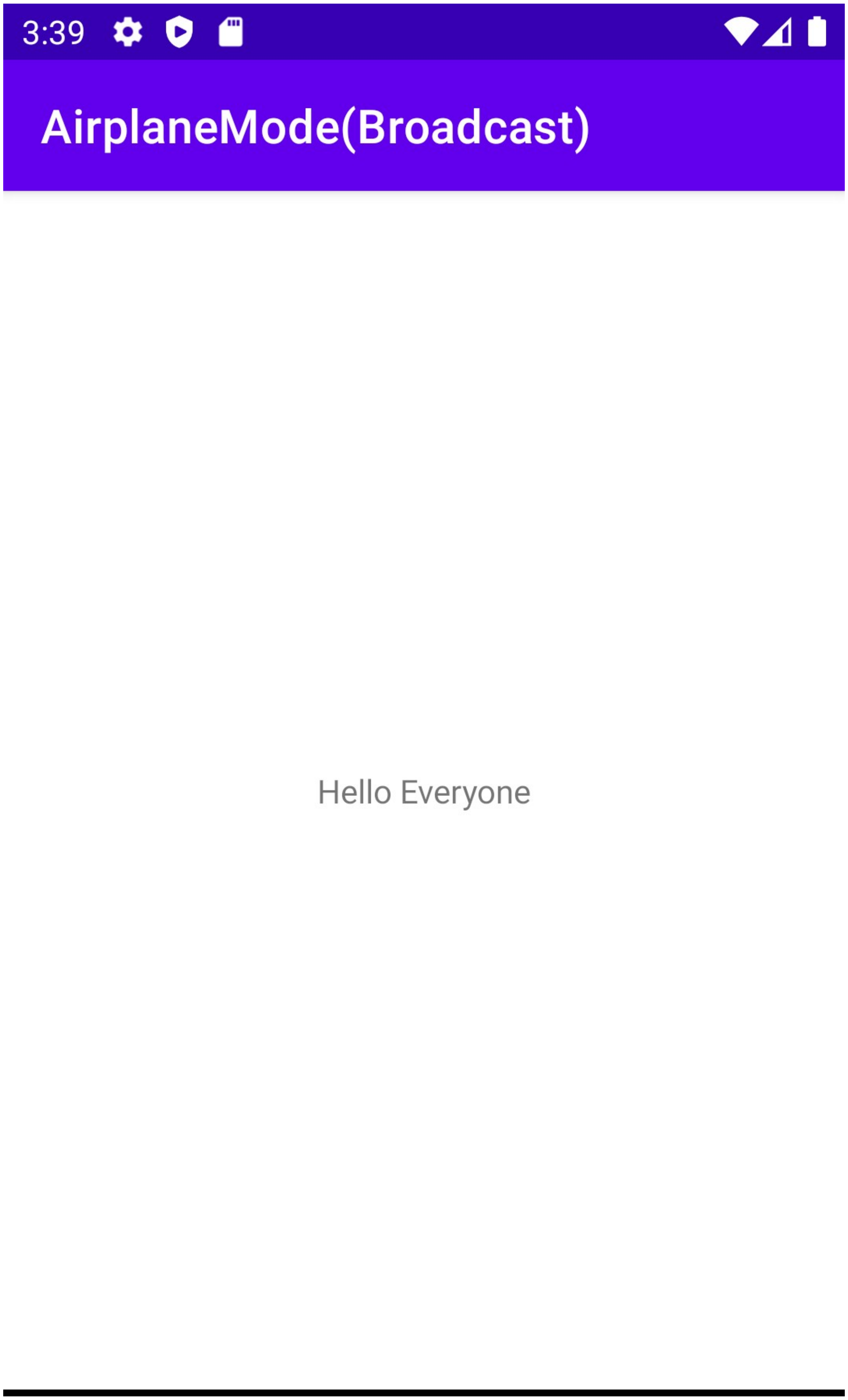
MainActivity.java

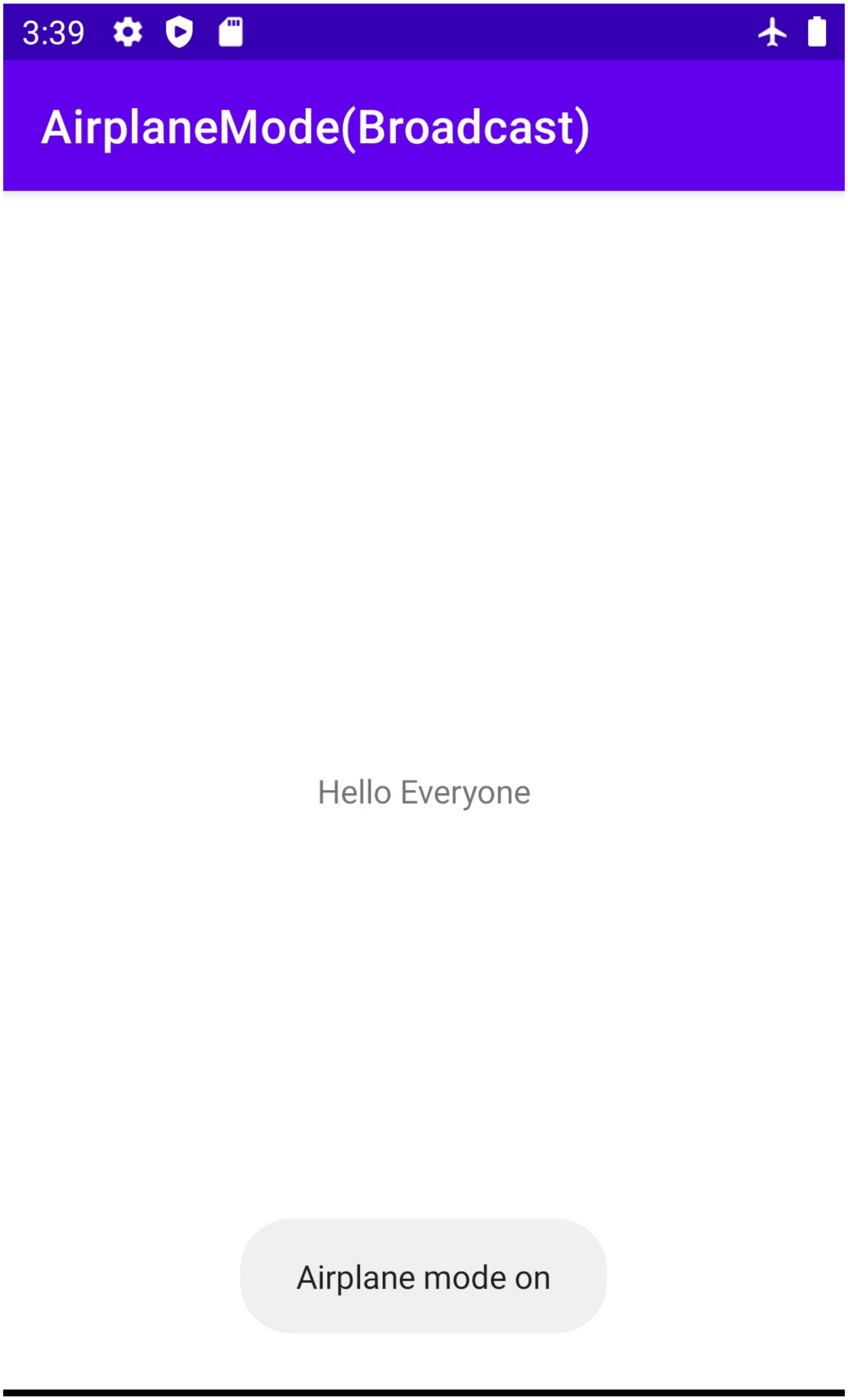
```java
package com.example.airplanemodebroadcast;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.IntentFilter;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    MyReceiver myReceiver = new MyReceiver();




    @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    protected void onStart() {
        super.onStart();
        IntentFilter filter = new
IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED);
        registerReceiver(myReceiver,filter);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unregisterReceiver(myReceiver);
    }
}
```

MyReceiver.java

```java
package com.example.airplanemodebroadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.provider.Settings;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent
intent) {
        // TODO: This method is called when the
BroadcastReceiver is receiving
        // an Intent broadcast.
       // throw new UnsupportedOperationException("Not
yet implemented");
        if
(isAirplaneModeOn(context.getApplicationContext())) {
            Toast.makeText(context, "AirPlane mode on",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "AirPlane mode
off", Toast.LENGTH_SHORT).show();
        }
    }

    private static boolean isAirplaneModeOn(Context
context) {
        return
Settings.System.getInt(context.getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) != 0;
    }
}
```

OUTPUT-


3:39
AirplaneMode(Broadcast)
Hello Everyone


3:39
AirplaneMode(Broadcast)
Hello Everyone
Airplane mode on

<u>CONCLUSION:</u>

Hence Broadcast Receiver is implemented successfully.

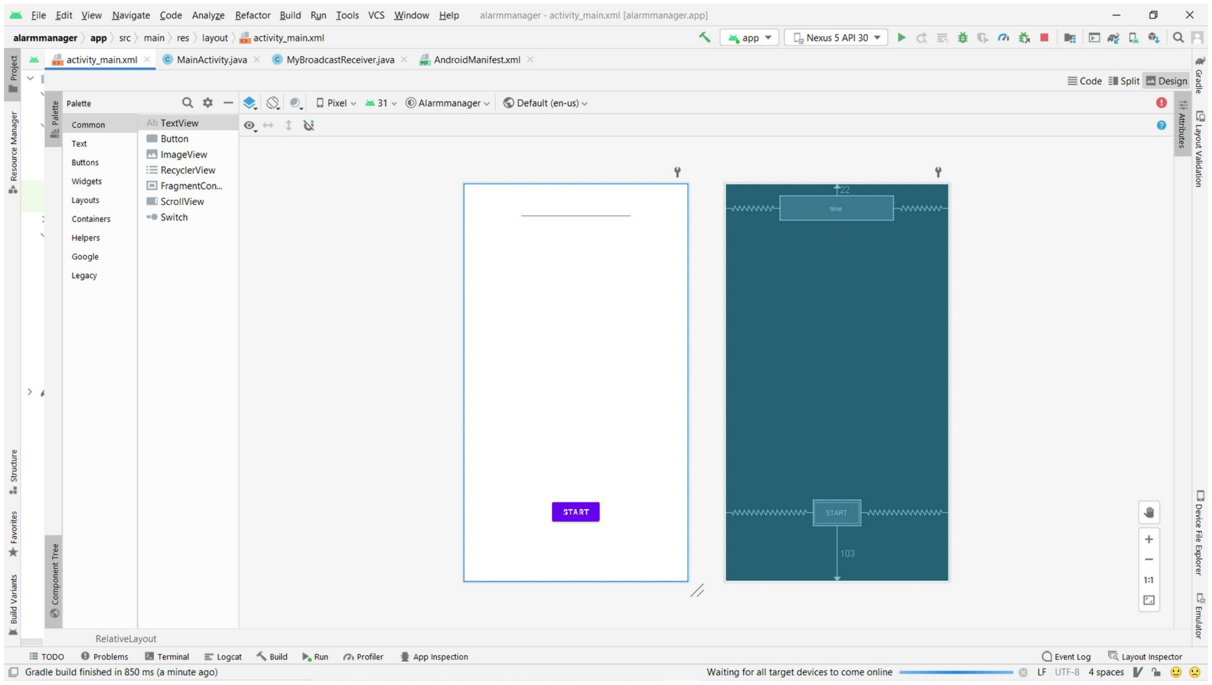
Practical 9:

AIM: **Create an android app to show Notifications and Alarm manager.**

Theory: **Alarms in Android have the following characteristics: Alarms let you send intents at set times or intervals. You can use alarms with broadcast receivers to start services and perform other operations.**

ALARM MANAGER

activity_main.xml



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
```

```
xmlns:android="http://schemas.android.com/a
pk/res/android"

xmlns:app="http://schemas.android.com/apk/r
es-auto"

xmlns:tools="http://schemas.android.com/too
ls"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#7FFFD4"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"

android:layout_height="wrap_content"
        android:text=" Start"

android:layout_alignParentBottom="true"

android:layout_centerHorizontal="true"
        android:layout_marginBottom="103dp"
/>
    <EditText
        android:id="@+id/time"
        android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_alignParentTop="true"

android:layout_centerHorizontal="true"
        android:layout_marginTop="22dp"
        android:ems="10" />


</RelativeLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/a
pk/res/android"
    package="com.example.alarmmanager">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"

android:roundIcon="@mipmap/ic_launcher_roun
d"
        android:supportsRtl="true"

android:theme="@style/Theme.Alarmmanager">
        <receiver

android:name=".MyBroadcastReceiver"
            android:enabled="true"

android:exported="true"></receiver>

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" /
>

                <category
android:name="android.intent.category.LAUNC
HER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

MyBroadcastReceiver.java

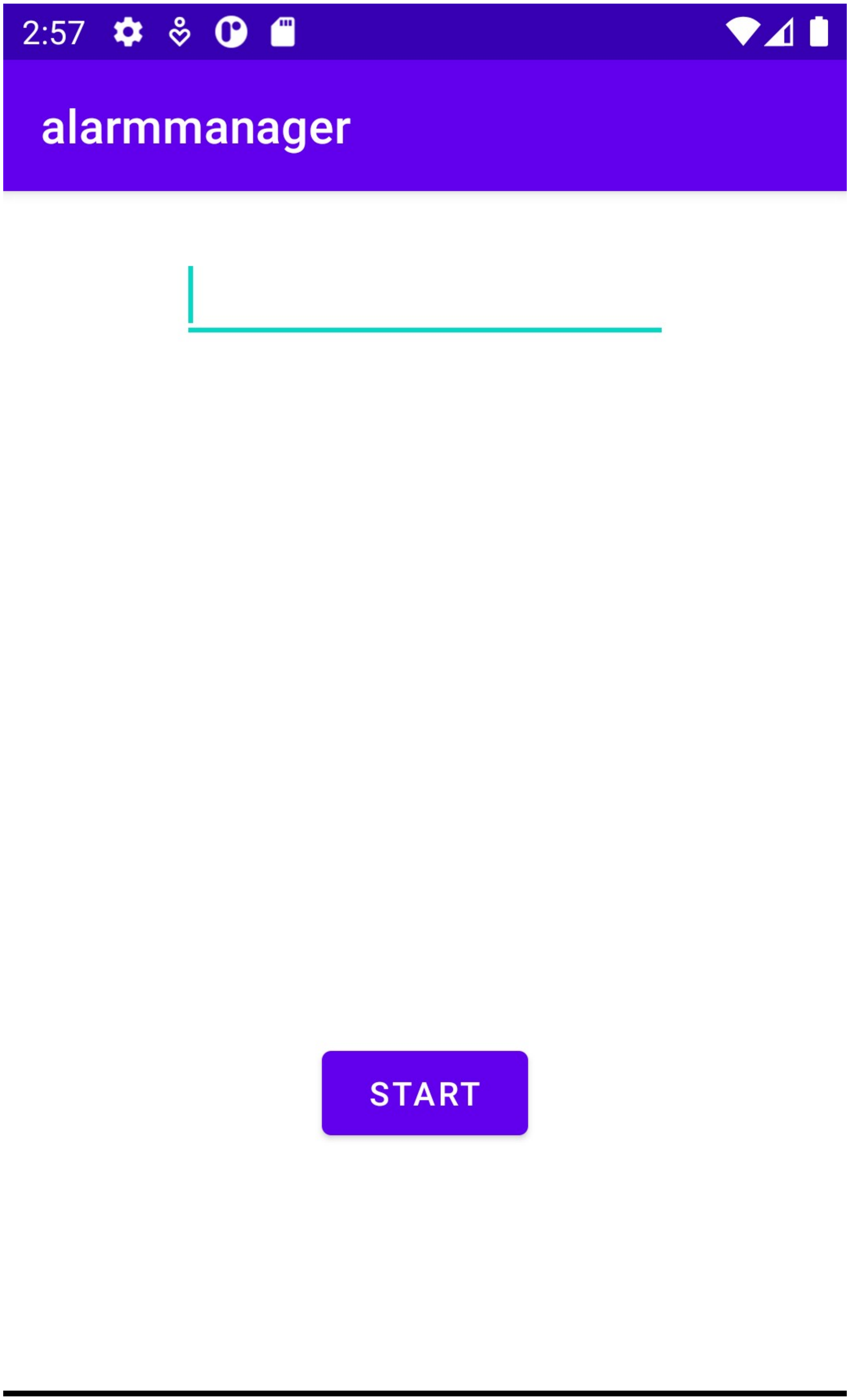
```
package com.example.alarmmanager;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.media.MediaPlayer;
import android.widget.Toast;


public class MyBroadcastReceiver extends
BroadcastReceiver {
    MediaPlayer mp;
    @Override
    public void onReceive(Context context,
Intent intent) {
        mp=MediaPlayer.create(context,
R.raw.alarm);
        mp.start();
        Toast.makeText(context,
"Ringing......", Toast.LENGTH_LONG).show();

    }
}
```

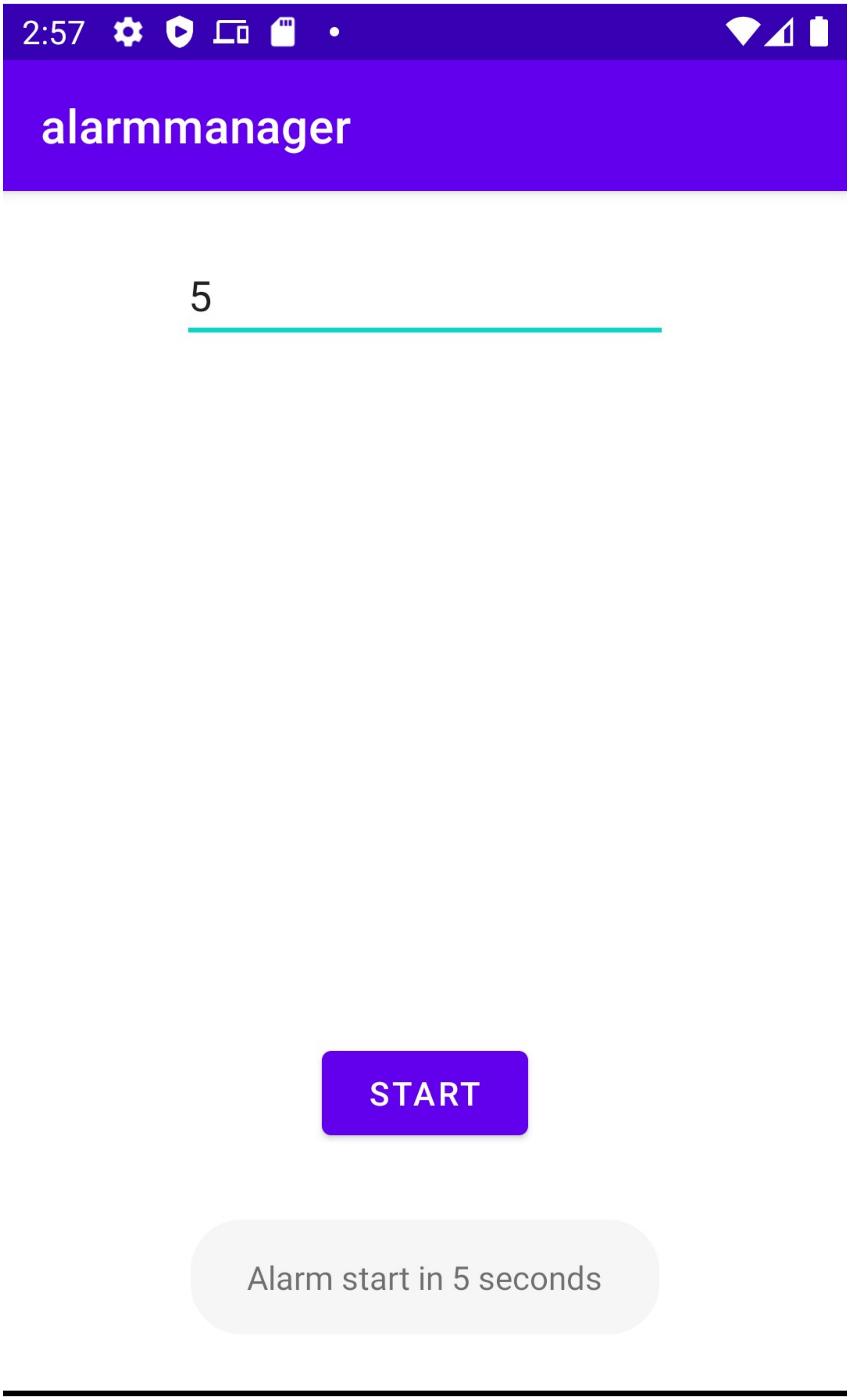
MainActivity.java

```
package com.example.alarmmanager;

import
androidx.appcompat.app.AppCompatActivity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import android.os.Bundle;

public class MainActivity extends
AppCompatActivity {
    Button start;
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);
        start= findViewById(R.id.button);

        start.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view)
{
                startAlert();
            }
        });
    }

    public void startAlert(){
        EditText text =
findViewById(R.id.time);
        int i =
Integer.parseInt(text.getText().toString())
;
        Intent intent = new Intent(this,
MyBroadcastReceiver.class);
        PendingIntent pendingIntent =
PendingIntent.getBroadcast(

this.getApplicationContext(), 0, intent,
0);
        AlarmManager alarmManager =
(AlarmManager)
getSystemService(ALARM_SERVICE);
```

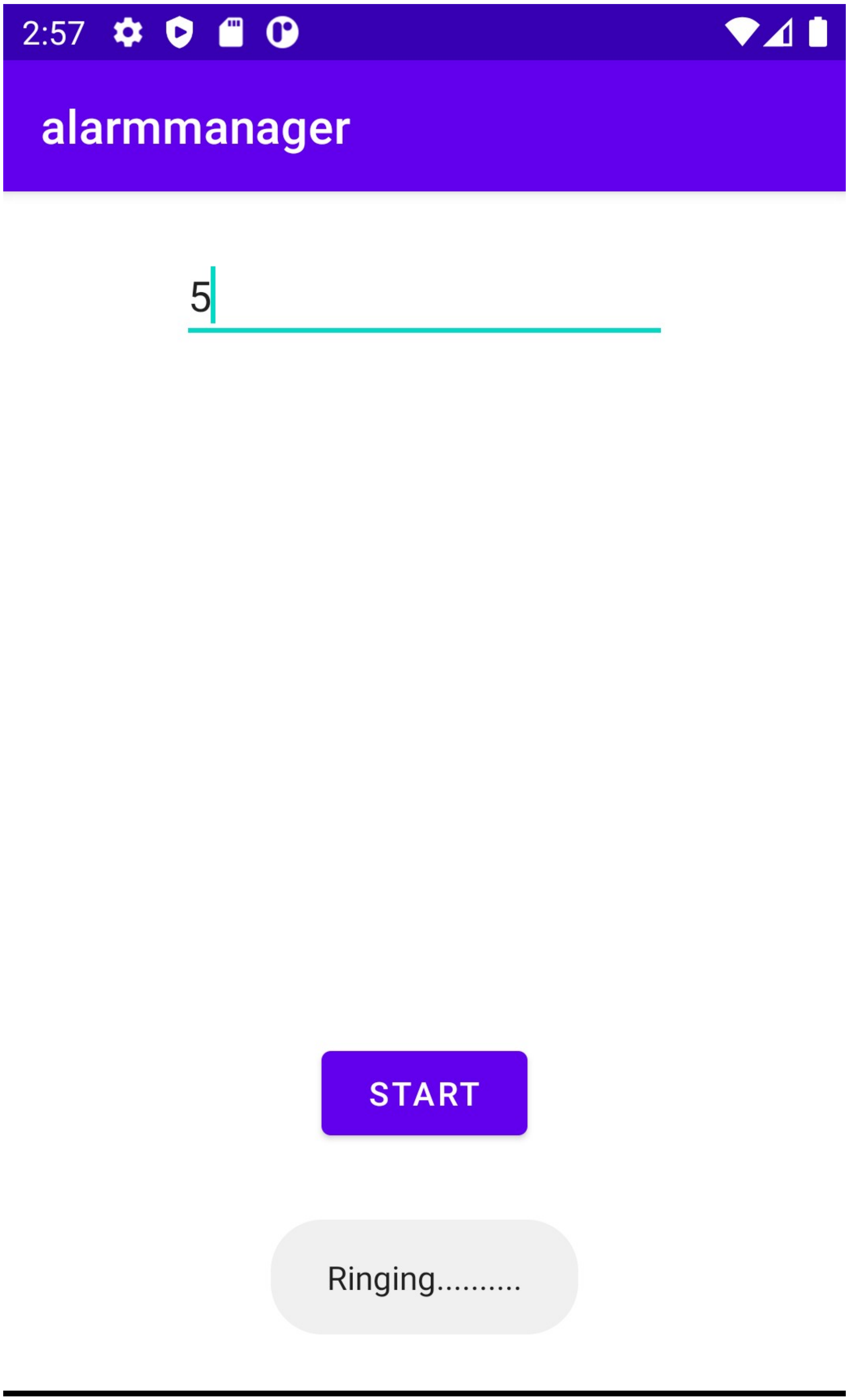
```
alarmManager.set(AlarmManager.RTC_WAKEUP,
System.currentTimeMillis()
                + (i * 1000),
pendingIntent);
        Toast.makeText(this, "Alarm start
in " + i + "
seconds",Toast.LENGTH_LONG).show();
    }
}
```

OUTPUT-

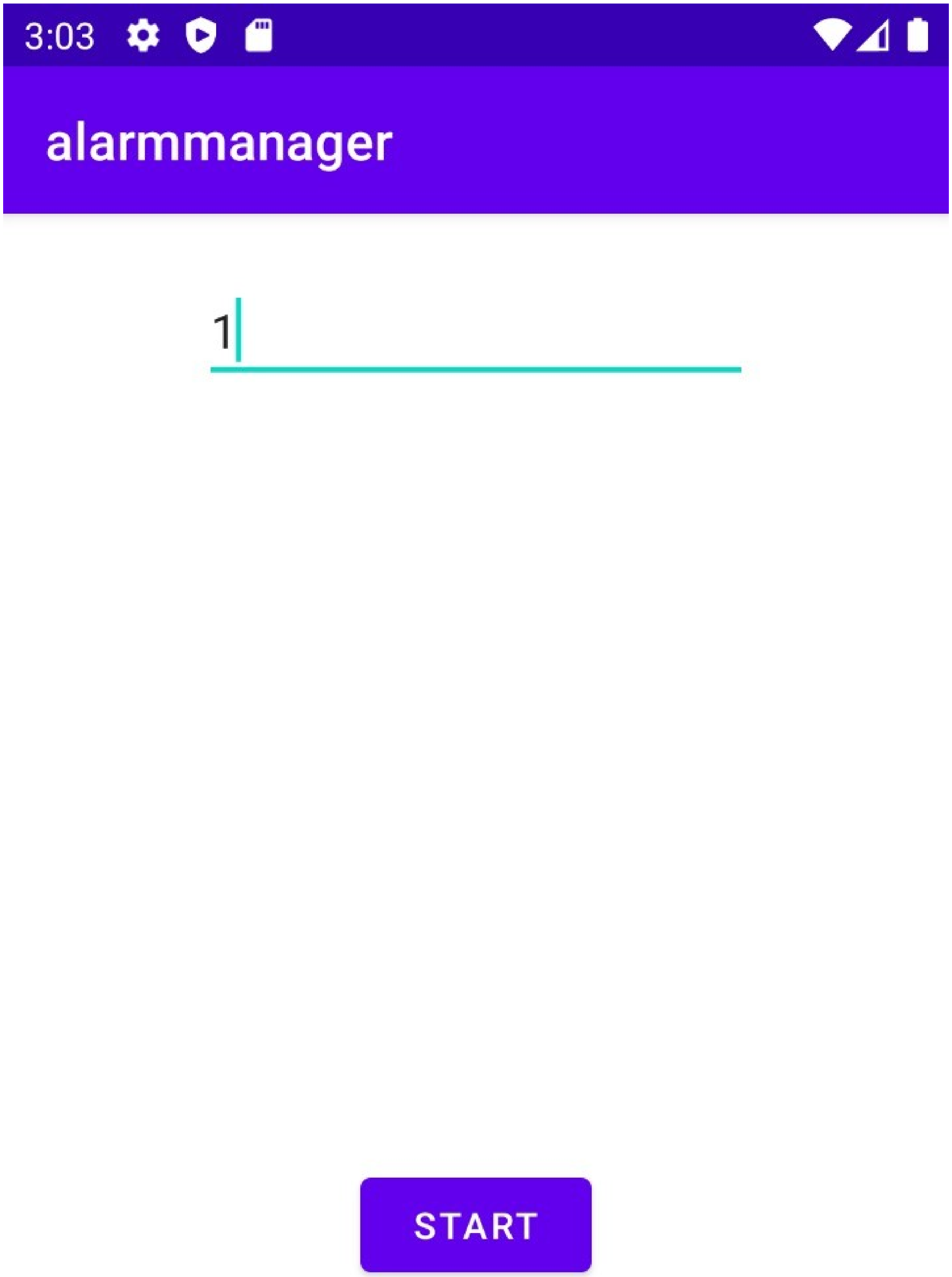

2:57
alarmmanager
START

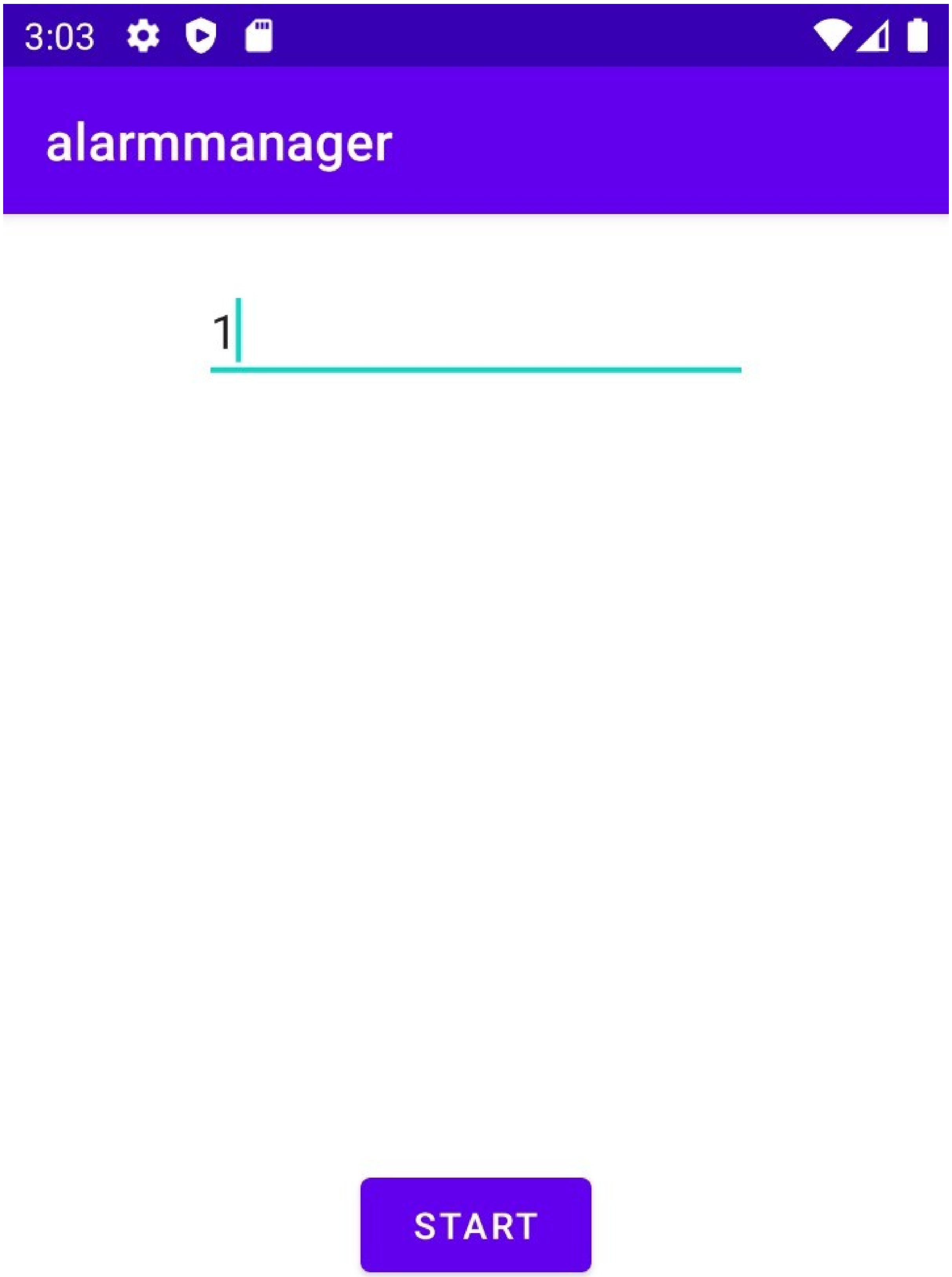


Alarm start in 5 seconds



Ringing..........


3:03
alarmmanager
1
START


3:03
alarmmanager
1
START

CONCLUSION:

Hence Alarm manager is implemented successfully