

# **Android Internship Report**

## **Personal Details**

Name : Priyeshkumar Chikhaliya  
College Name : Babaria Institute of Technology.  
Degree : BE  
Semester : 7<sup>th</sup>  
Roll no. : 180050131005  
GitHub URL : [GitHub Profile](https://github.com/Priyeshchikhaliya)

Or

<https://github.com/Priyeshchikhaliya>

## **Company Details**

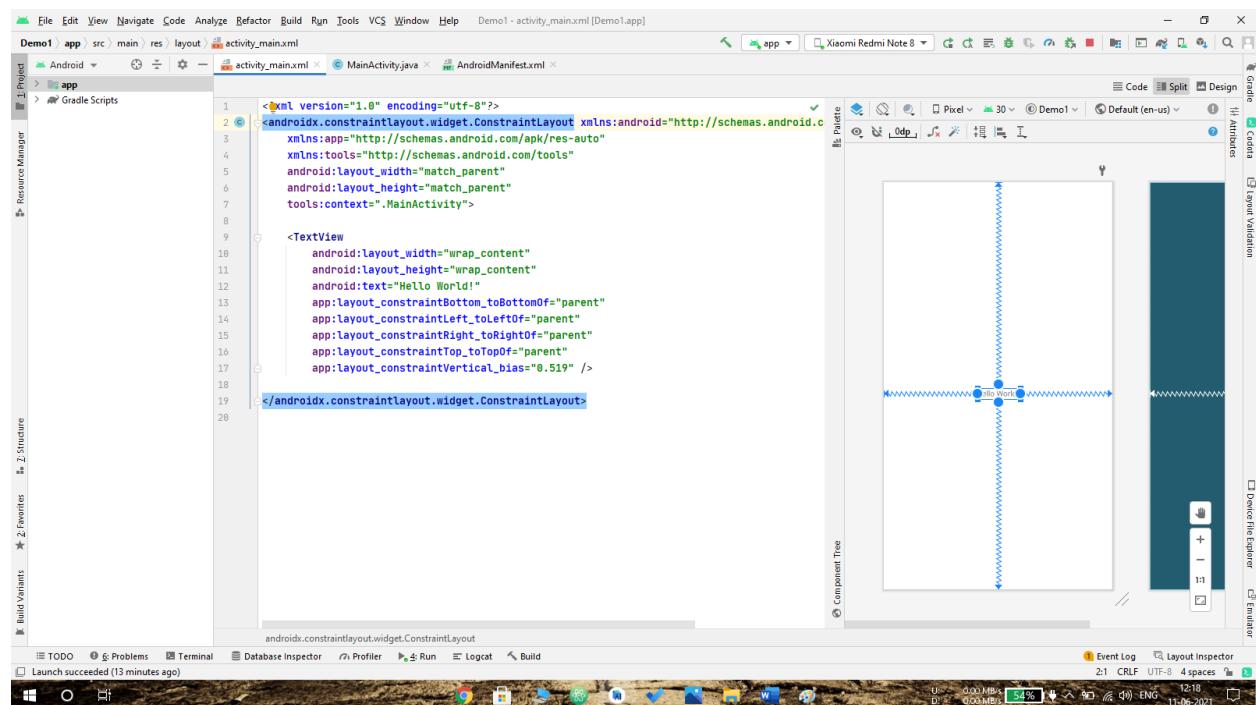
Company Name : Akash Technolabs  
External Guide : Devansi Prajapati  
CEO : Akash Padhiyar  
Training Duration : 26-05-2021 to 14-06-2021

# Index

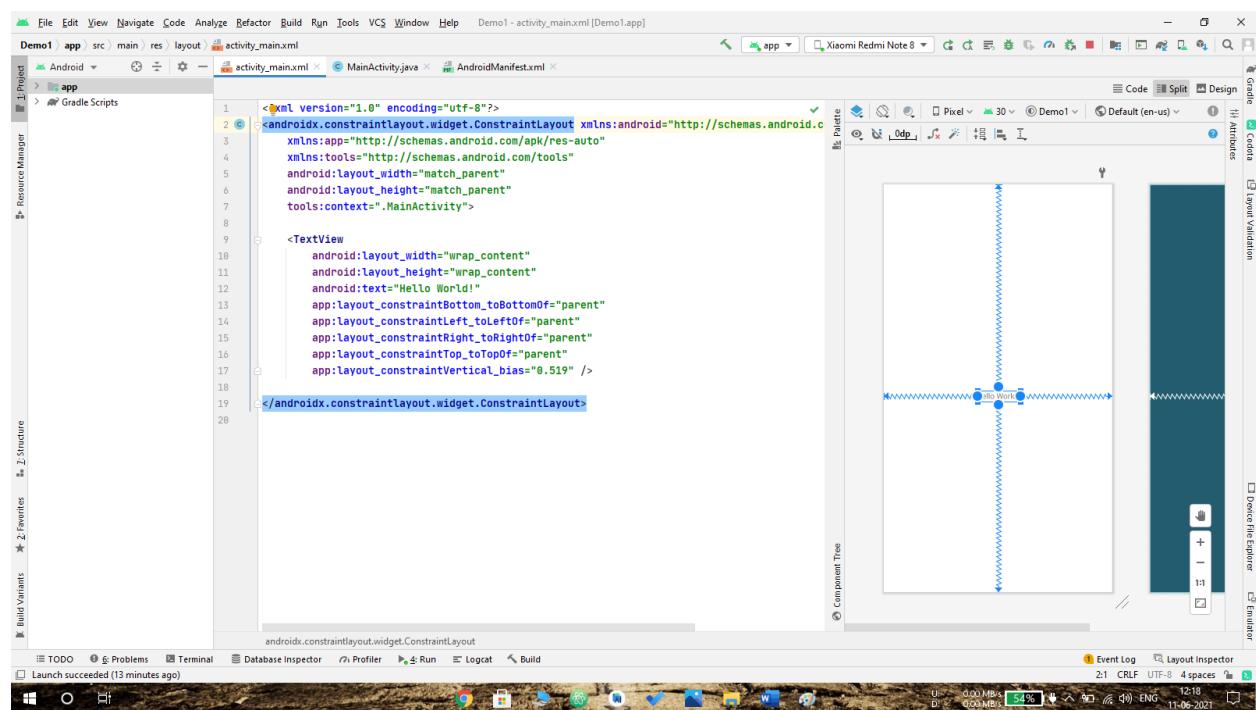
<u>Index</u>	<u>Task Detail</u>	<u>Page No.</u>
1	Demo 1	3
2	NormalCalc & Summation	5
3	Intent , LifeCycle	12
4	MyRegistration	17
5	Three Splash Screen and Splash Screen	21
6	List View-Master and Custom List View	31
7	Shared Preferences	35
8	Phone	39
9	PostOffice	42
10	Message	44
11	Player	46
12	Binge	48
	<b>Project</b>	
	VideoPlayerApp-master	50

# 1- Demo1

## Main Activity:



## Activity\_main.xml:



Output:



Hello World!

## 2 – NormalCalc:

In this two Inputs required for the operation. In this we can do 4 operation  
Summation, Subtraction, Division, Multiplication.

In this we get the data from user input like:

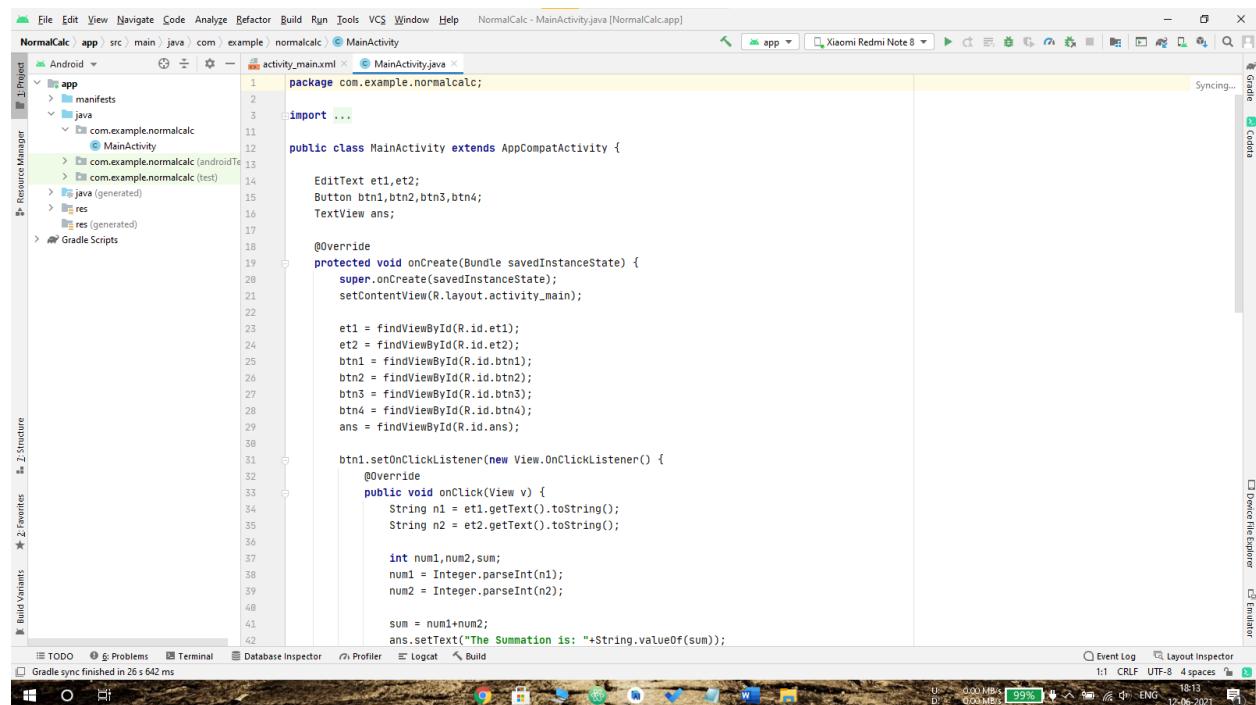
**Number1 = editText.getText().toString();**

And after operation performed, we have to set the data to the TextView like:

**textView.setText("Text");**

### Code:

#### Main Activity:



The screenshot shows the Android Studio interface with the project 'NormalCalc' open. The main window displays the 'MainActivity.java' file under the 'src/main/java/com/example/normalcalc' package. The code implements a summation operation between two integers obtained from EditText fields. The code is as follows:

```
package com.example.normalcalc;

import ...

public class MainActivity extends AppCompatActivity {

    EditText et1,et2;
    Button btn1,btn2,btn3,btn4;
    TextView ans;

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

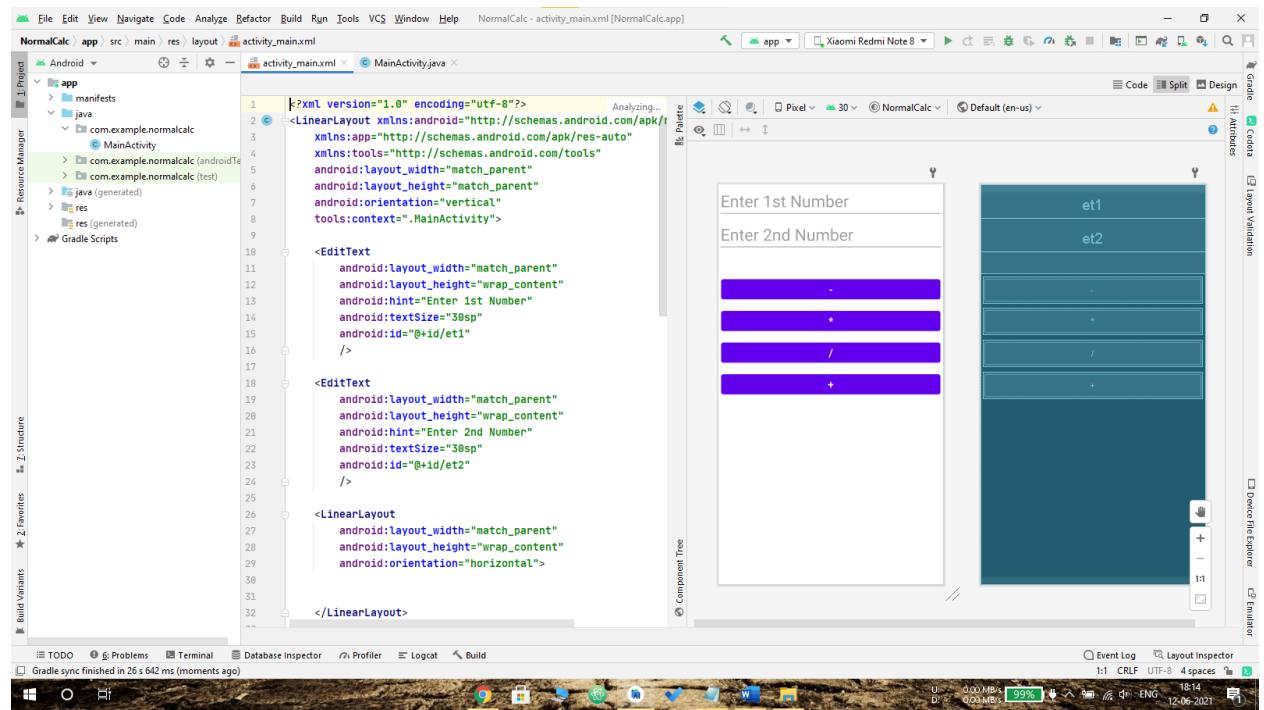
        et1 = findViewById(R.id.et1);
        et2 = findViewById(R.id.et2);
        btn1 = findViewById(R.id.btn1);
        btn2 = findViewById(R.id.btn2);
        btn3 = findViewById(R.id.btn3);
        btn4 = findViewById(R.id.btn4);
        ans = findViewById(R.id.ans);

        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String n1 = et1.getText().toString();
                String n2 = et2.getText().toString();

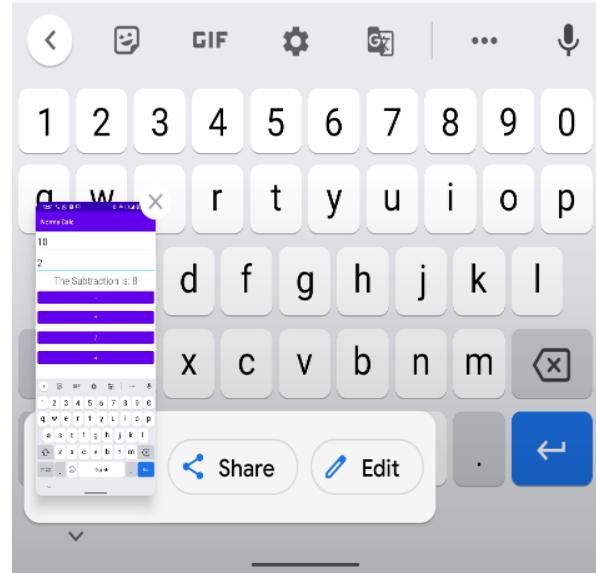
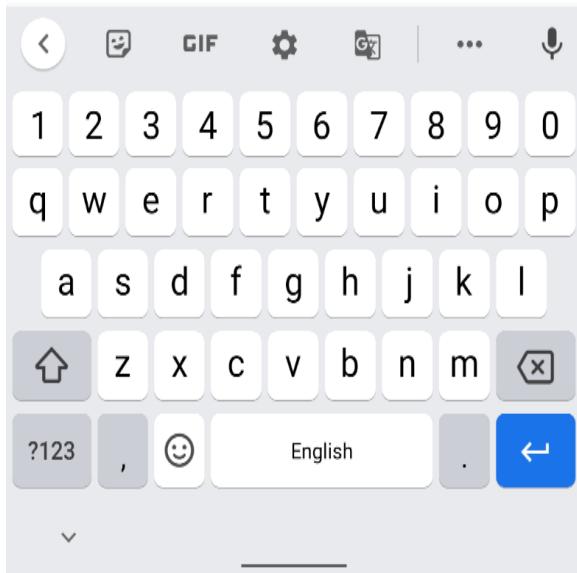
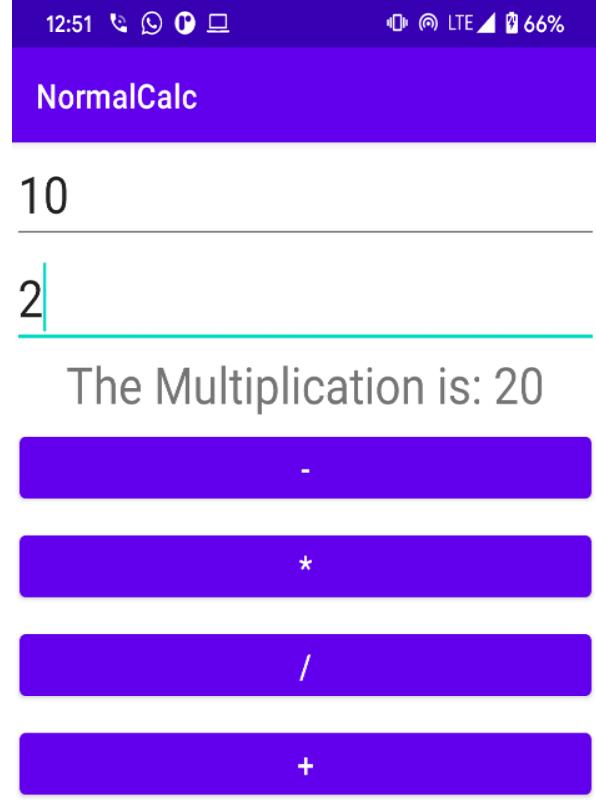
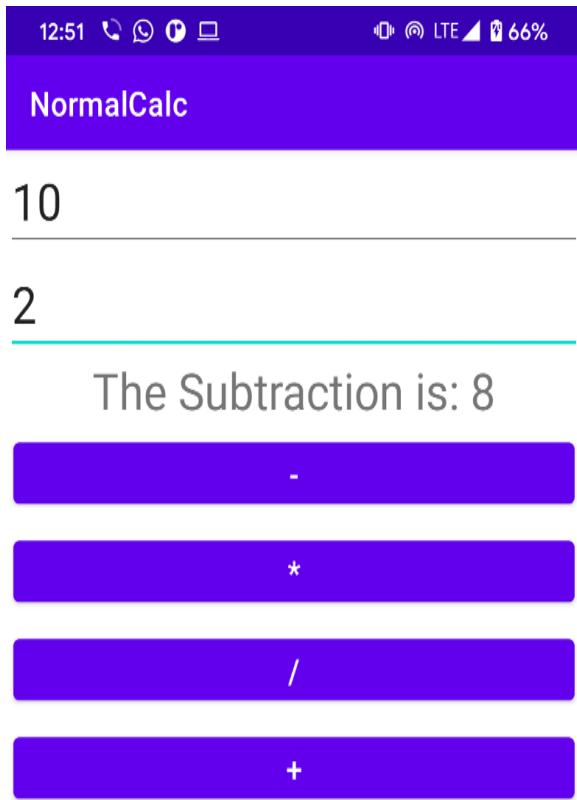
                int num1,num2,sum;
                num1 = Integer.parseInt(n1);
                num2 = Integer.parseInt(n2);

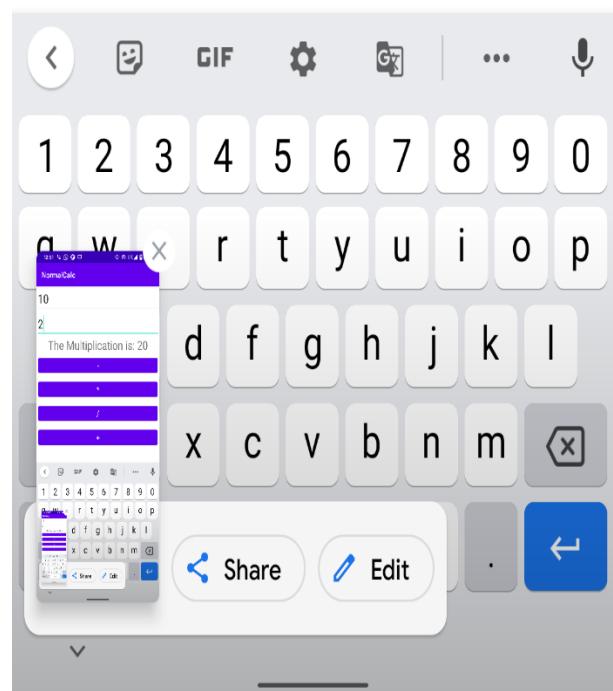
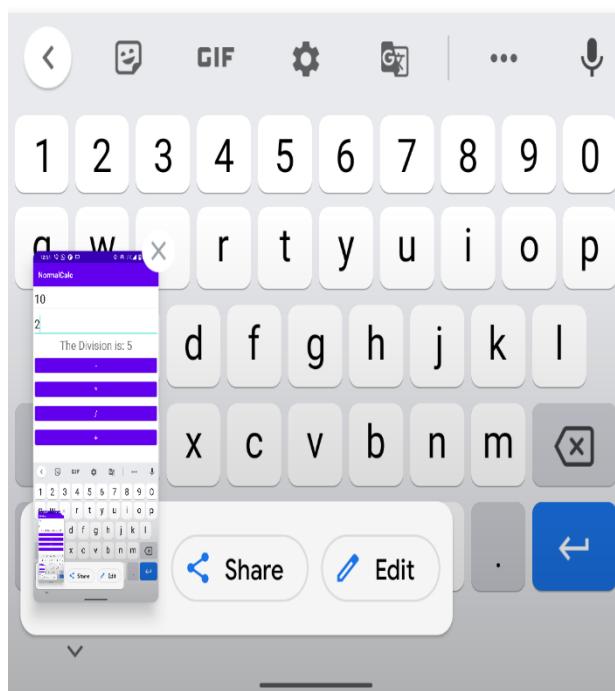
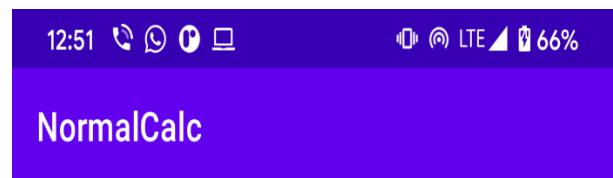
                sum = num1+num2;
                ans.setText("The Summation is: "+String.valueOf(sum));
            }
        });
    }
}
```

## Activity\_main.xml:



Output:





## 2.1 – Summation:

### Code:

#### Main Activity:

The screenshot shows the Android Studio interface with the project 'Summation' open. The code editor displays the MainActivity.java file, which contains Java code for a simple addition application. The code includes imports for EditText, Button, and TextView, and defines two EditText fields (et1, et2), a Button (btn1), and a TextView (ans). It overrides the onCreate method to set the content view and initialize the views. It then sets an onClickListener for the button that retrieves text from the EditText fields, converts them to integers, adds them together, and displays the result in the TextView.

```
package com.example.summation;
import ...;
public class MainActivity extends AppCompatActivity {
    EditText et1, et2;
    Button btn1;
    TextView ans;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et1 = findViewById(R.id.et1);
        et2 = findViewById(R.id.et2);
        btn1 = findViewById(R.id.btn1);
        ans = findViewById(R.id.ans);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String n1 = et1.getText().toString();
                String n2 = et2.getText().toString();
                int num1, num2, sum;
                num1 = Integer.parseInt(n1);
                num2 = Integer.parseInt(n2);
                sum = num1 + num2;
                ans.setText("Sum: " + sum);
            }
        });
    }
}
```

## Activity\_main.xml:

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "Summation" and contains an "app" module with "manifests", "java", and "res" directories.
- Code Editor:** The current file is "activity\_main.xml". The XML code defines a linear layout with two edit texts for entering numbers and a button to calculate the sum.
- Layout Editor:** The right side of the screen shows the visual representation of the layout. It consists of a vertical linear layout containing two edit texts labeled "et1" and "et2", and a button labeled "SUM".
- Bottom Bar:** Shows the build variants (Current File), problems (8 problems), and system status (CPU: 0.00MHz, RAM: 98%, 1847, ENG, 12-06-2021).

```
<?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"
    tools:context=".MainActivity"
    android:orientation="vertical">

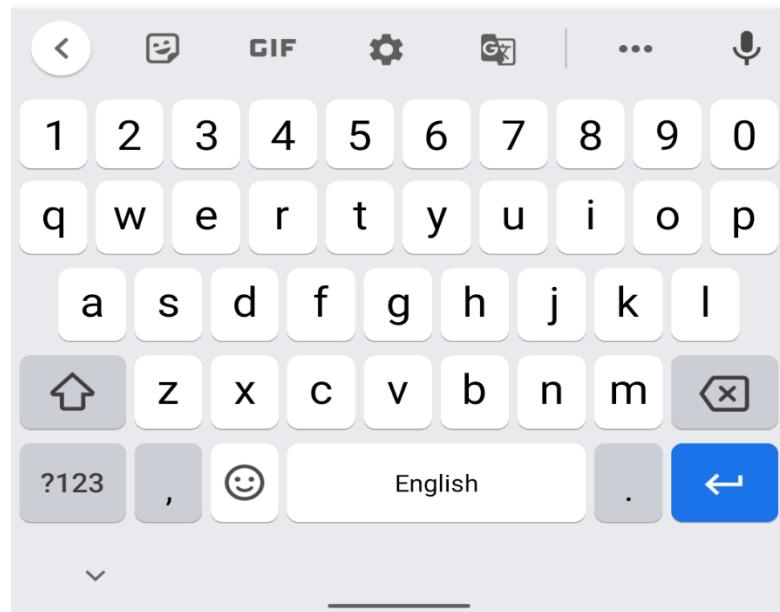
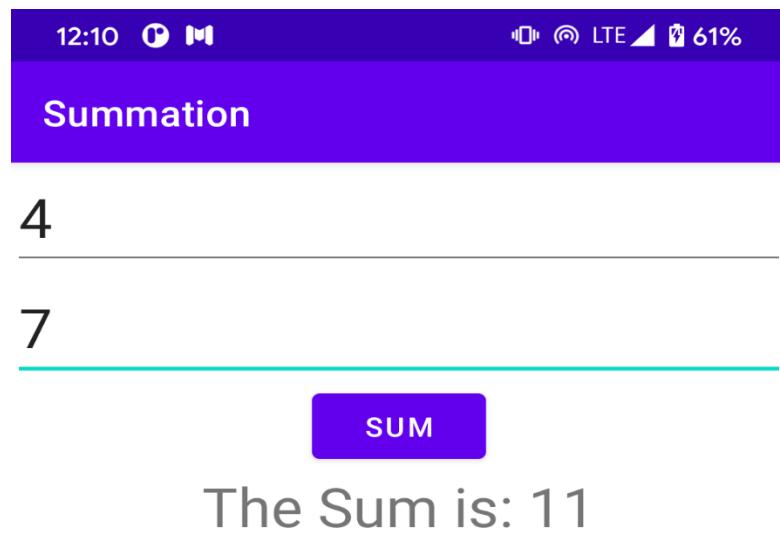
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter 1st Number"
        android:textSize="30sp"
        android:id="@+id/et1"
        />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter 2nd Number"
        android:textSize="30sp"
        android:id="@+id/et2"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="SUM"
        android:id="@+id/button1"
        />

```

Output:



### 3 - Intent:

Android app may contain one or more screens. Intent basically used to provide navigation from one activity to other activity. Intents allow us to interact with components from the same applications as well other third-party applications

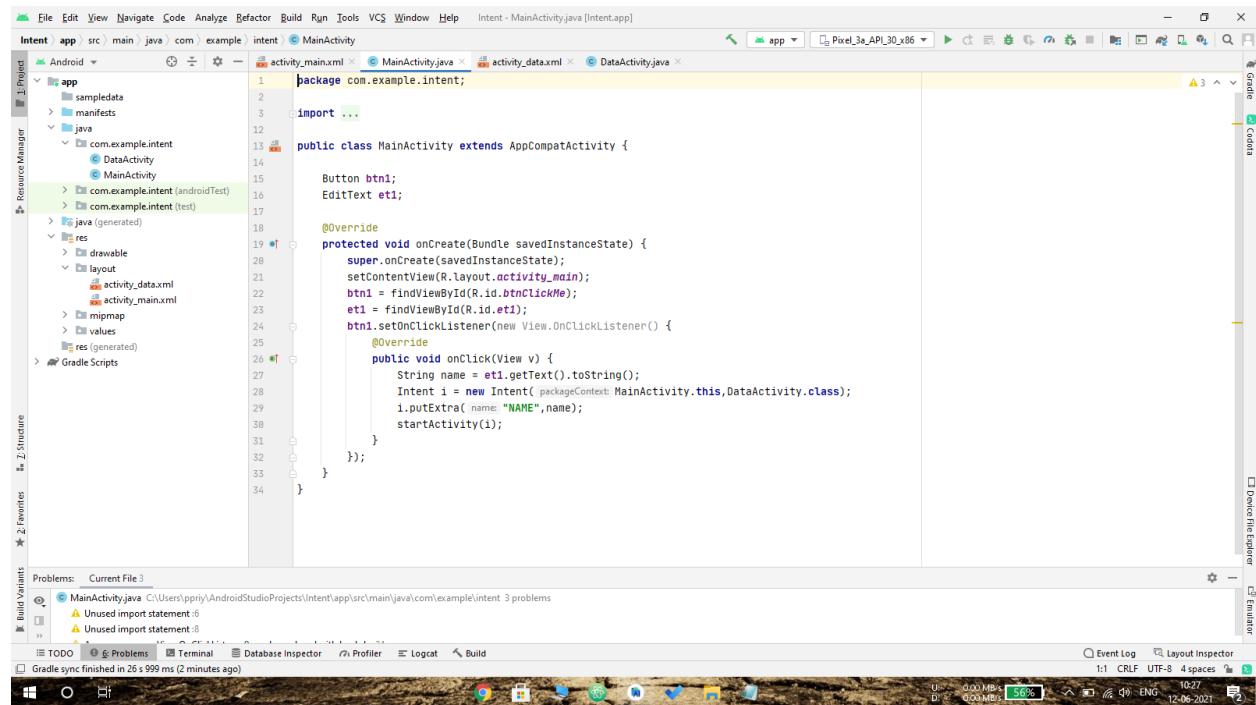
We can use intent like this:

```
Intent intent = new Intent(context,navigationActivity.class);
```

```
startActivity(intent);
```

### Code:

#### Main Activity:



The screenshot shows the Android Studio interface with the project structure and code editor. The code in MainActivity.java is as follows:

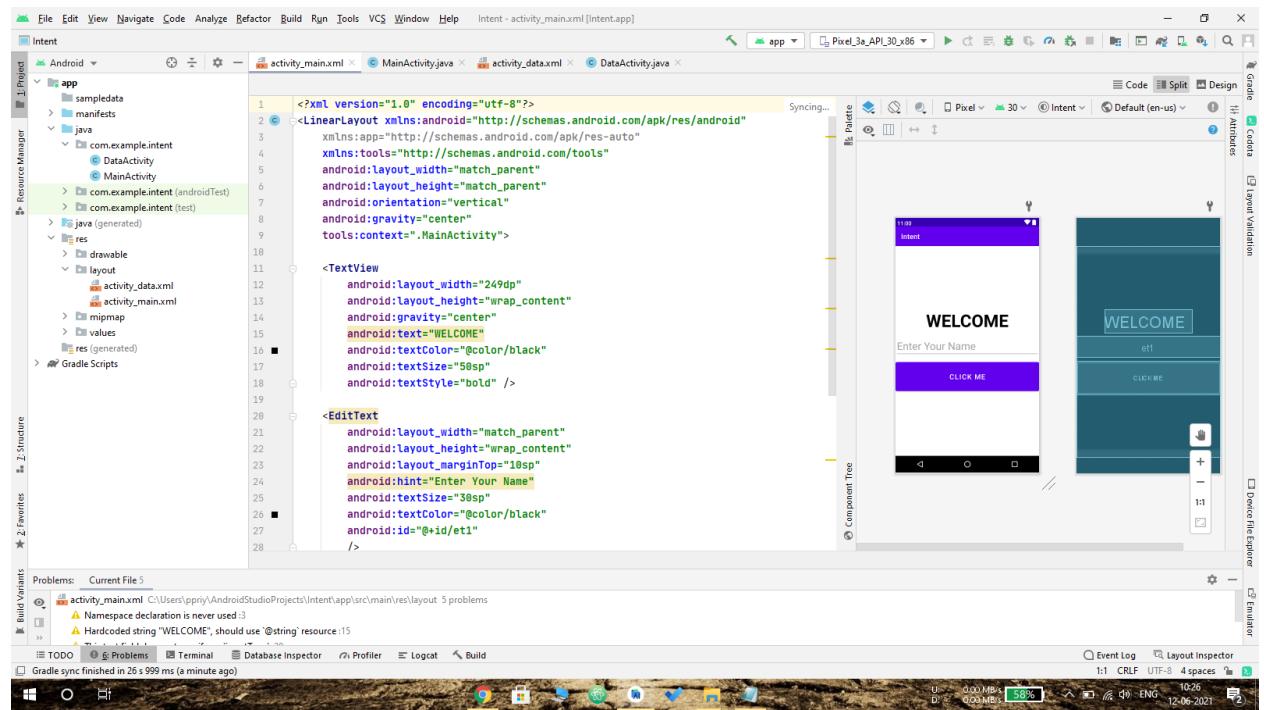
```
package com.example.intent;
import ...;

public class MainActivity extends AppCompatActivity {
    Button btn1;
    EditText et1;

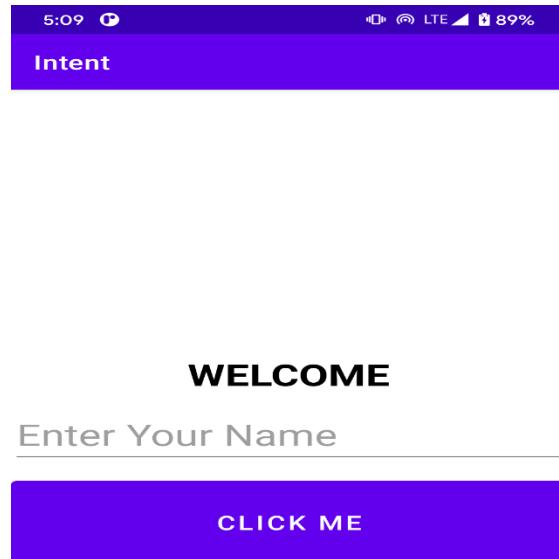
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn1 = findViewById(R.id.btnClickMe);
        et1 = findViewById(R.id.et1);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = et1.getText().toString();
                Intent i = new Intent(getApplicationContext(), DataActivity.class);
                i.putExtra("NAME",name);
                startActivity(i);
            }
        });
    }
}
```

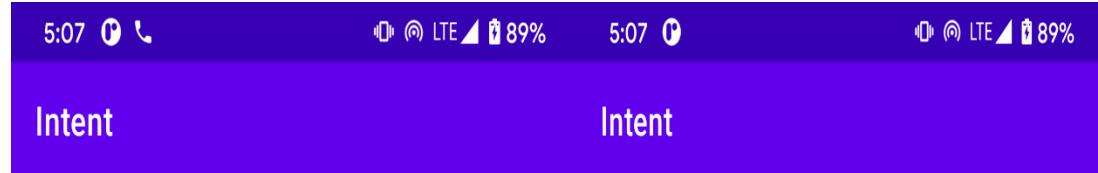
The code defines an activity that contains a button and an edit text field. When the button is clicked, it retrieves the text from the edit text, creates an intent pointing to the DataActivity, and starts it.

## Activity\_main.xml:



## Output:





**WELCOME**

Priyesh Chikhaliya

**CLICK ME**

**WELCOME PRIYESH  
CHIKHALIYA**

### 3.1 – Activity life cycle:

There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle.

Main activity:

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "Lifecycle". It contains an "app" module with "manifests", "java", and "res" directories. The "java" directory contains the "MainActivity.java" file.
- Code Editor:** The "MainActivity.java" file is open, displaying the following Java code:

```
package com.example.lifecycle;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("TAG","onCreate");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d("TAG","onStart");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d("TAG","onResume");
    }

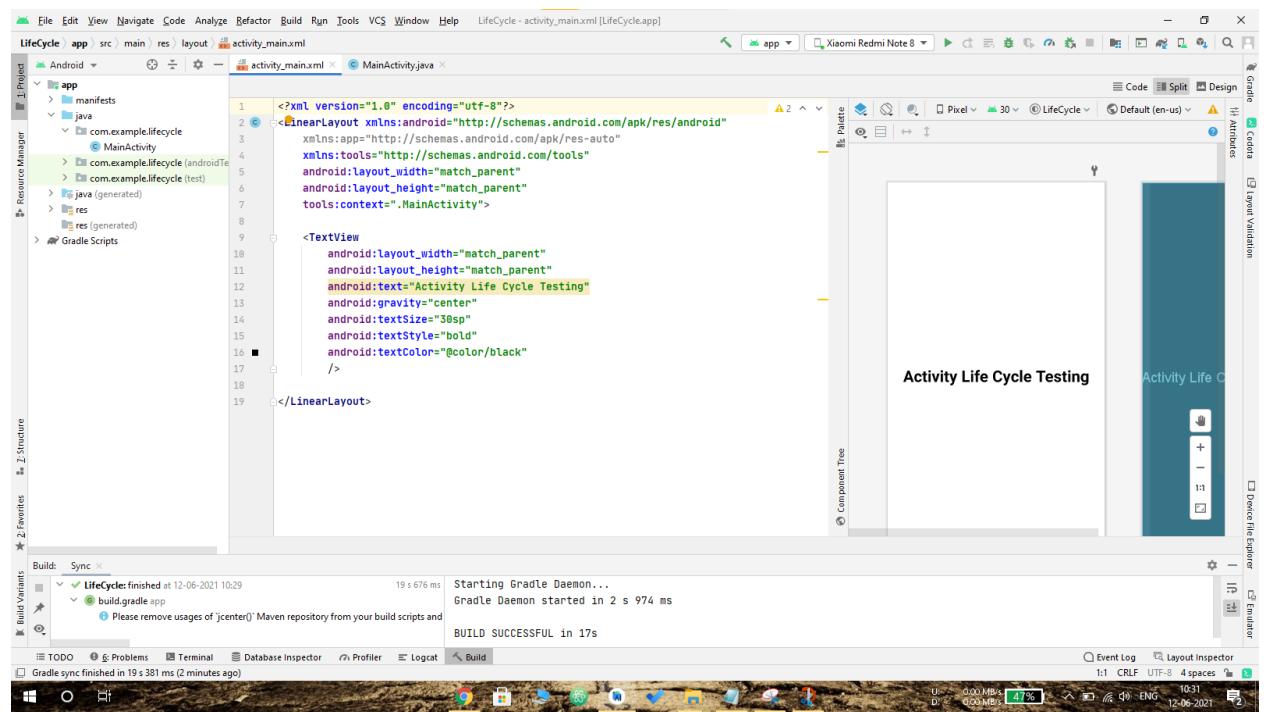
    @Override
    protected void onPause() {
        super.onPause();
        Log.d("TAG","onPause");
    }
}
```

- Build Log:** The bottom pane shows the build process:

  - Gradle sync completed at 12-06-2021 10:29.
  - Starting Gradle Daemon...
  - Gradle Daemon started in 2 s 974 ms.
  - BUILD SUCCESSFUL in 17s.

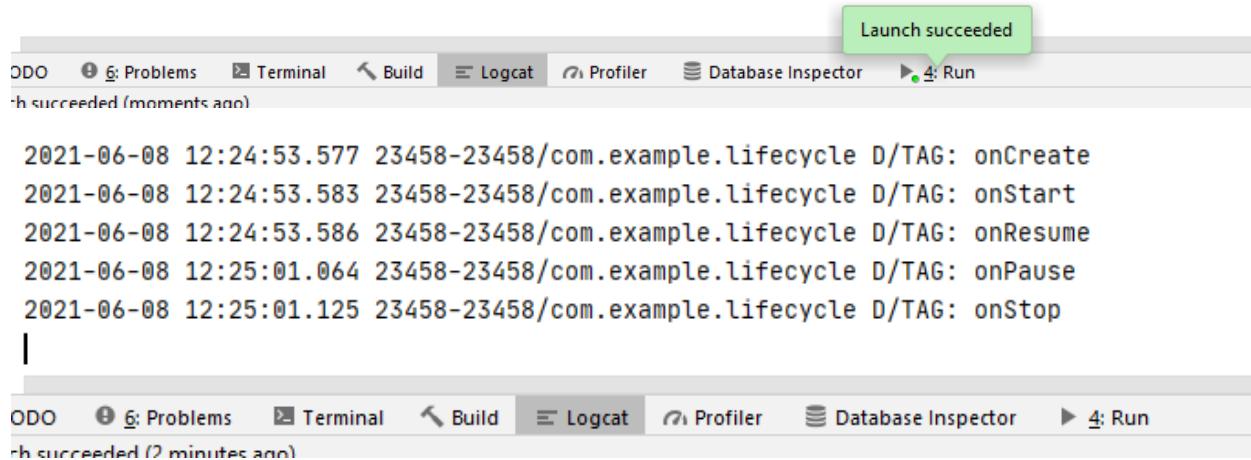
- System Status:** The taskbar at the bottom shows various system icons and the date/time.

## Activity\_main.xml:



## Logcat:

```
2021-06-08 12:23:12.186 5861-5861/com.example.lifecycle D/TAG: onCreate
2021-06-08 12:23:12.192 5861-5861/com.example.lifecycle D/TAG: onStart
2021-06-08 12:23:12.196 5861-5861/com.example.lifecycle D/TAG: onResume
```

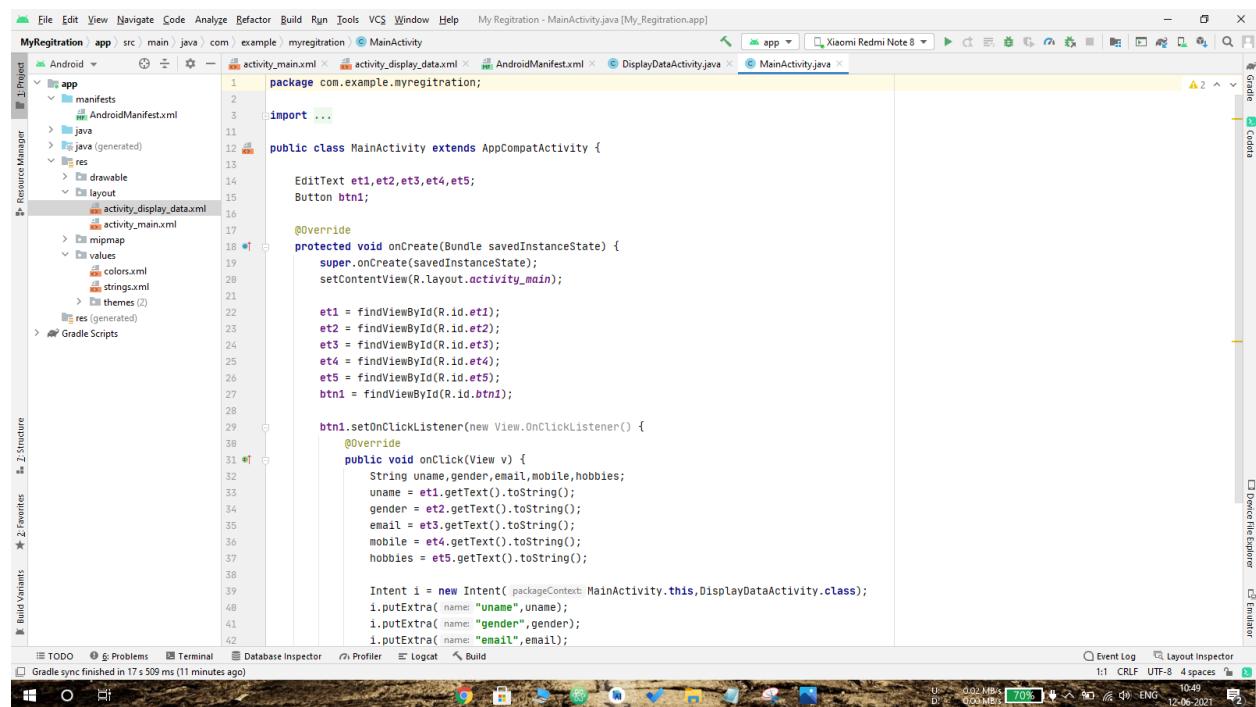


## 4 - MyRegistration

In this app it takes inputs like registration details and that details send to other activity called data activity and set that data to text view using **setText()** Method.

### Code:

#### Main Activity:



```
MyRegistration - MainActivity.java [My_Registration.app]
MyRegistration > app > src > main > java > com > example > myregistration > MainActivity
```

```
1 package com.example.myregistration;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     EditText et1,et2,et3,et4,et5;
8     Button btn1;
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_main);
14
15        et1 = findViewById(R.id.et1);
16        et2 = findViewById(R.id.et2);
17        et3 = findViewById(R.id.et3);
18        et4 = findViewById(R.id.et4);
19        et5 = findViewById(R.id.et5);
20        btn1 = findViewById(R.id.btn1);
21
22        btn1.setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                String uname,gender,email,mobile,hobbies;
26                uname = et1.getText().toString();
27                gender = et2.getText().toString();
28                email = et3.getText().toString();
29                mobile = et4.getText().toString();
30                hobbies = et5.getText().toString();
31
32                Intent i = new Intent(getApplicationContext(),DisplayDataActivity.class);
33                i.putExtra("uname",uname);
34                i.putExtra("gender",gender);
35                i.putExtra("email",email);
36
37            }
38
39        });
40
41    }
42}
```

The screenshot shows the Android Studio interface with the code editor open to the MainActivity.java file. The code implements an OnClickListener for a button, which retrieves text from five EditText fields and starts an Intent to the DisplayDataActivity. The project structure is visible on the left, and various toolbars and panels are at the bottom.

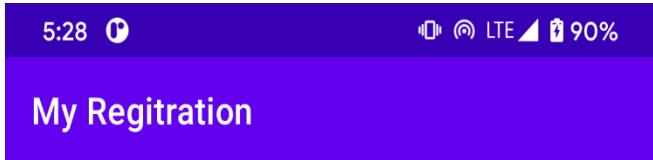
## Activity\_main.xml:

The screenshot shows the Android Studio interface with the project 'MyRegistration' open. The 'activity\_main.xml' file is selected in the layout editor. The XML code defines a vertical linear layout containing five text input fields and a submit button. The design preview on the right shows a dark-themed user interface with white text for the input fields and a purple 'SUBMIT' button. The bottom of the screen displays the Windows taskbar and system tray.

```
<?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"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Please Enter the Details"
        android:textAllCaps="true"
        android:textSize="30sp"
        android:gravity="center"
        android:paddingBottom="40sp"
        android:textColor="@color/black"
        android:textStyle="bold"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Username"
        android:textSize="30sp"
        android:id="@+id/et1"
        android:textColor="@color/black"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Password"
        android:textSize="30sp"
        android:id="@+id/et2"
        android:textColor="@color/black"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Email"
        android:textSize="30sp"
        android:id="@+id/et3"
        android:textColor="@color/black"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Mobile number"
        android:textSize="30sp"
        android:id="@+id/et4"
        android:textColor="@color/black"
        />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Hobbies(At least 4)"
        android:textSize="30sp"
        android:id="@+id/et5"
        android:textColor="@color/black"
        />
</LinearLayout>
```

Output:



## PLEASE ENTER THE DETAILS

Enter Username

Type Your Gender

Enter Email

Enter Mobile number

Enter Password

Enter Hobbies(At least 4)

SUBMIT

5:57

LTE 93%

My Registration

5:57

LTE 93%

My Registration

## PLEASE ENTER THE DETAILS

Priyeshkumar Chikhaliya

Male

cr433270@gmail.com

6353711716

Enter Password

Coding, Gaming, Watching  
Food shows

SUBMIT

Welcome  
**Priyeshkumar  
Chikhaliya**

Gender:Male

Email:cr433270@gmail.com

Mobile:6353711716

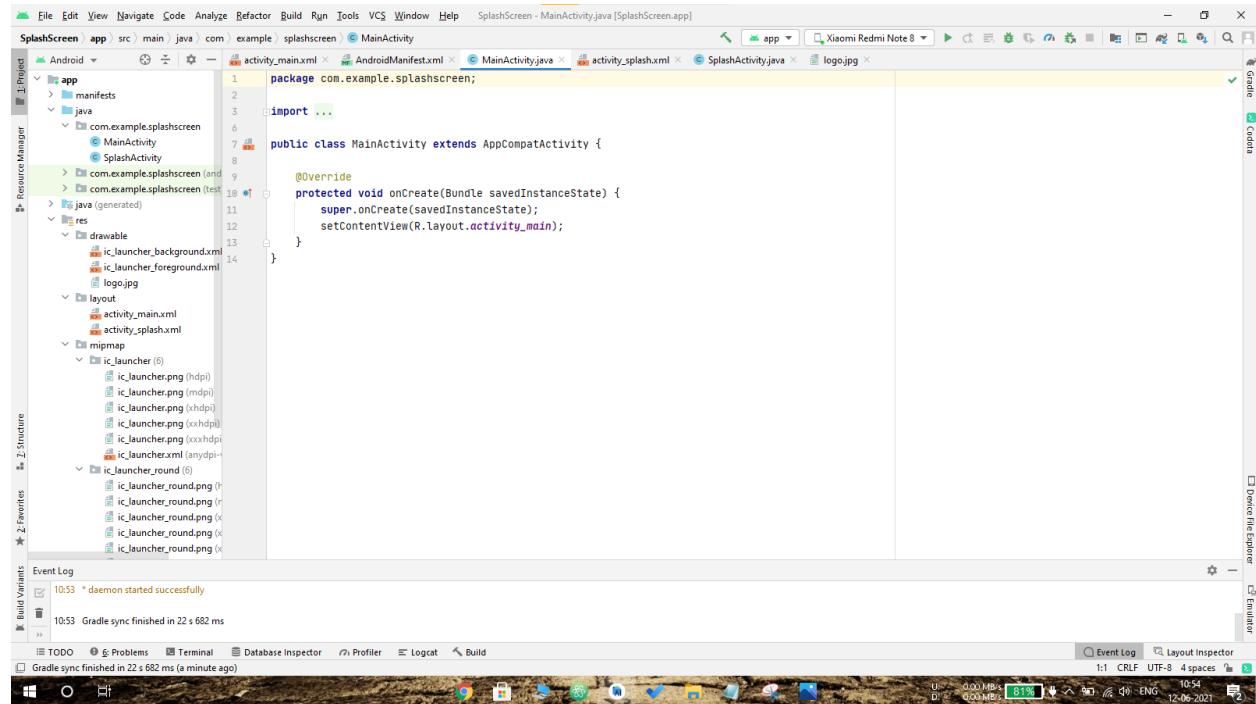
Hobbies:Coding, Gaming, Watching  
Food shows

## 5 -Splash Screen:

Android splash screen are normally used to show user some kind of progress before the app's main screen. Some people use splash screen just to show their app / company logo for a couple of second (Mostly 3 to 4 seconds).

### Code:

#### Main activity:



The screenshot shows the Android Studio interface with the project 'SplashScreen' open. The code editor displays the MainActivity.java file, which contains the following code:

```
package com.example.splashscreen;

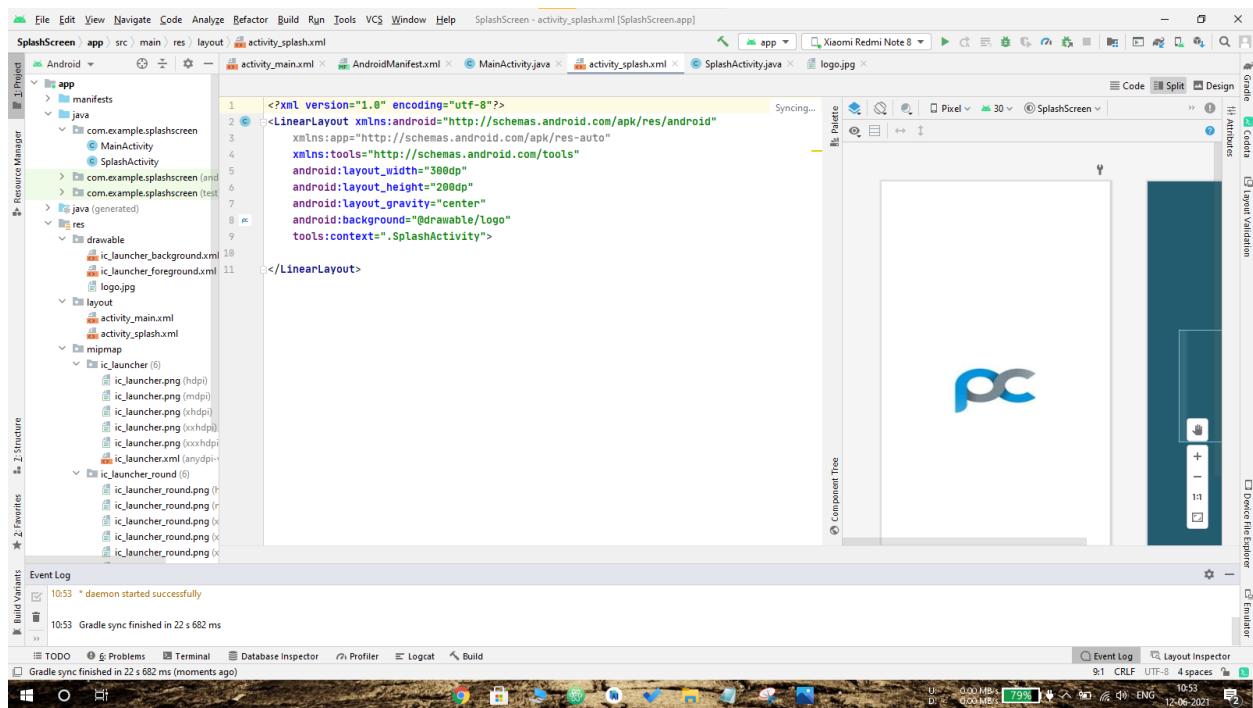
import ...

public class MainActivity extends AppCompatActivity {

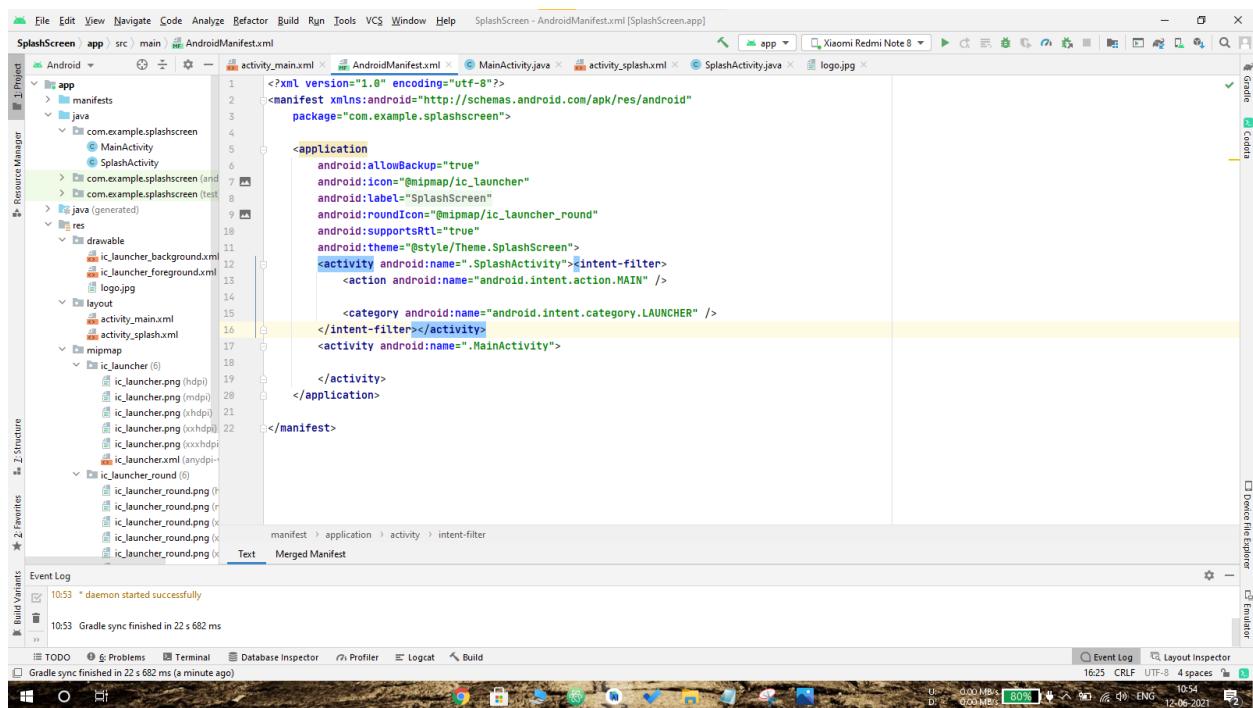
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

The Project tool window on the left shows the structure of the app, including the manifest, Java files (MainActivity.java, SplashActivity.java), resources (drawable, layout, mipmap), and XML files (activity\_main.xml, activity\_splash.xml). The Event Log at the bottom shows a successful Gradle sync.

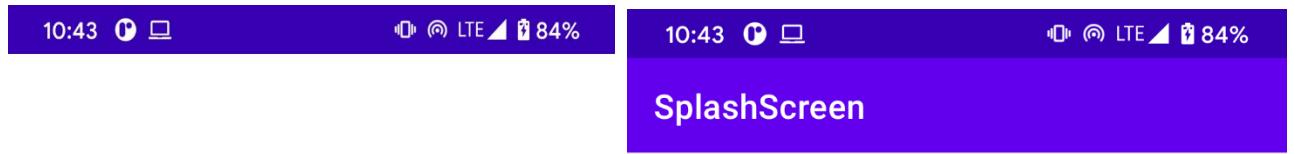
## Activity\_main.xml:



## AndroidManifest.xml:



Output:



**Application**



## 5.1 -Three Splash Screen:

### Main activity:

The screenshot shows the Android Studio interface with the project 'ThreeSplashScreen' open. The main window displays the Java code for `MainActivity.java`. The code defines a class `MainActivity` that extends `AppCompatActivity`. It overrides the `onCreate` method to set the content view to `R.layout.activity_main`. The project structure on the left shows files like `AndroidManifest.xml`, `activity_main.xml`, and various Java and XML files for other activities.

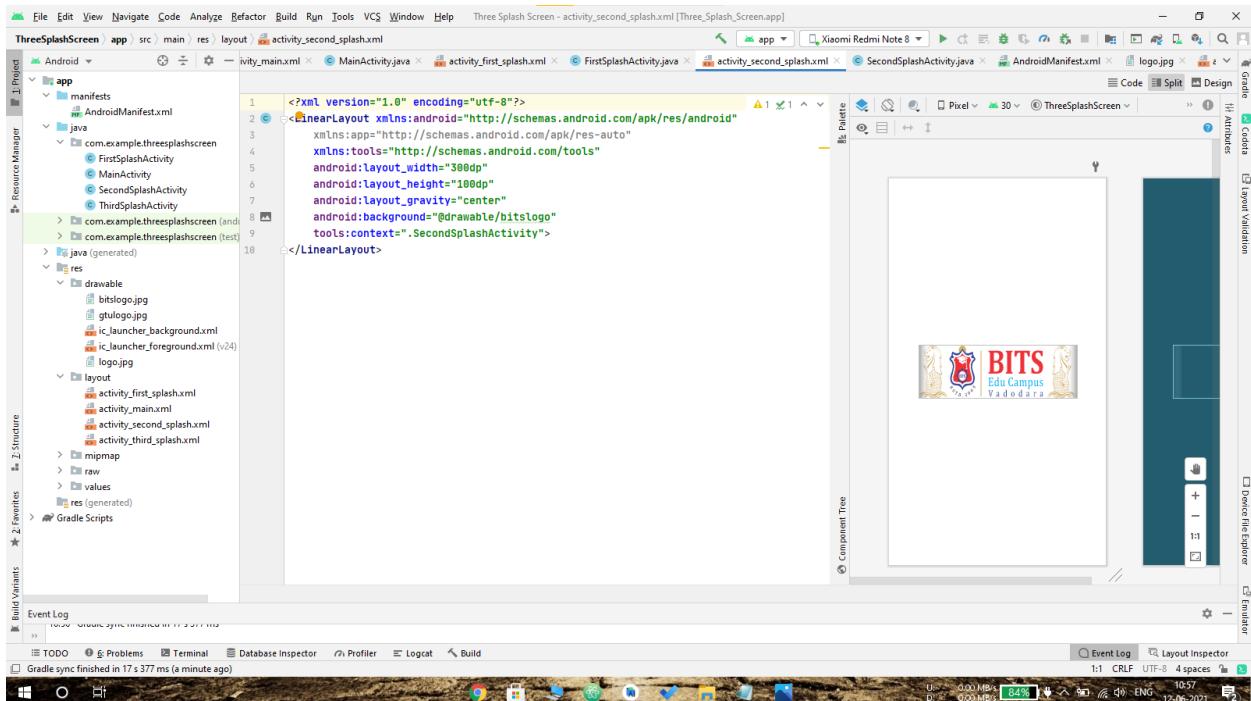
```
1 package com.example.threesplashscreen;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12}
13
14}
```

### Activity\_first\_splash.xml:

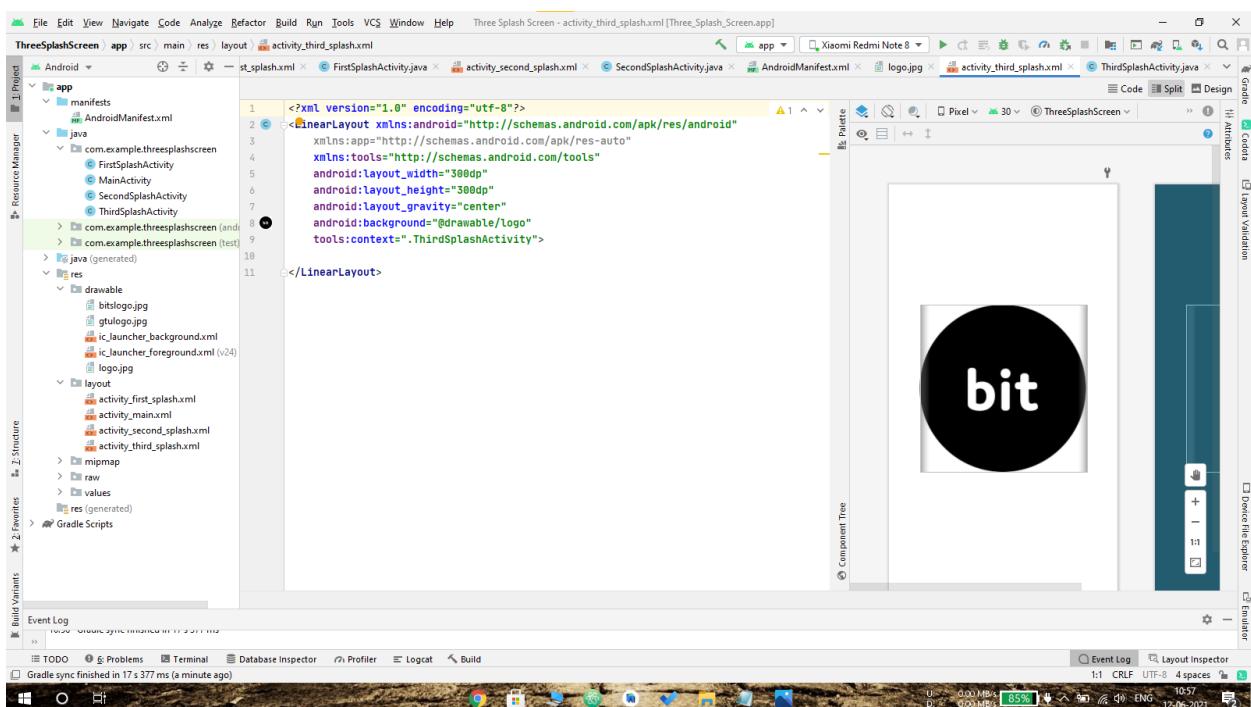
The screenshot shows the Android Studio interface with the layout file `activity_first_splash.xml` selected. The code defines a `LinearLayout` with a single child, another `LinearLayout`. This second `LinearLayout` has attributes `android:layout_width="300dp"`, `android:layout_height="300dp"`, and `android:gravity="center"`. It also includes a background reference `@drawable/gtulogo` and a tools context attribute `tools:context=".FirstSplashActivity"`. To the right, the preview pane shows a shield-shaped logo with a blue border and a red center, featuring a white rocket and some text in Devanagari script.

```
<?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="300dp"
    android:layout_height="300dp"
    android:gravity="center"
    android:background="@drawable/gtulogo"
    tools:context=".FirstSplashActivity">
</LinearLayout>
```

## Activity\_second\_splash.xml:



## Activity\_third\_splash.xml:



## Output:



10:35

LTE 83%



## 6 - List View-Master:

The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

**android:id, android:divider , android:dividerHeight** are the properties of List View.

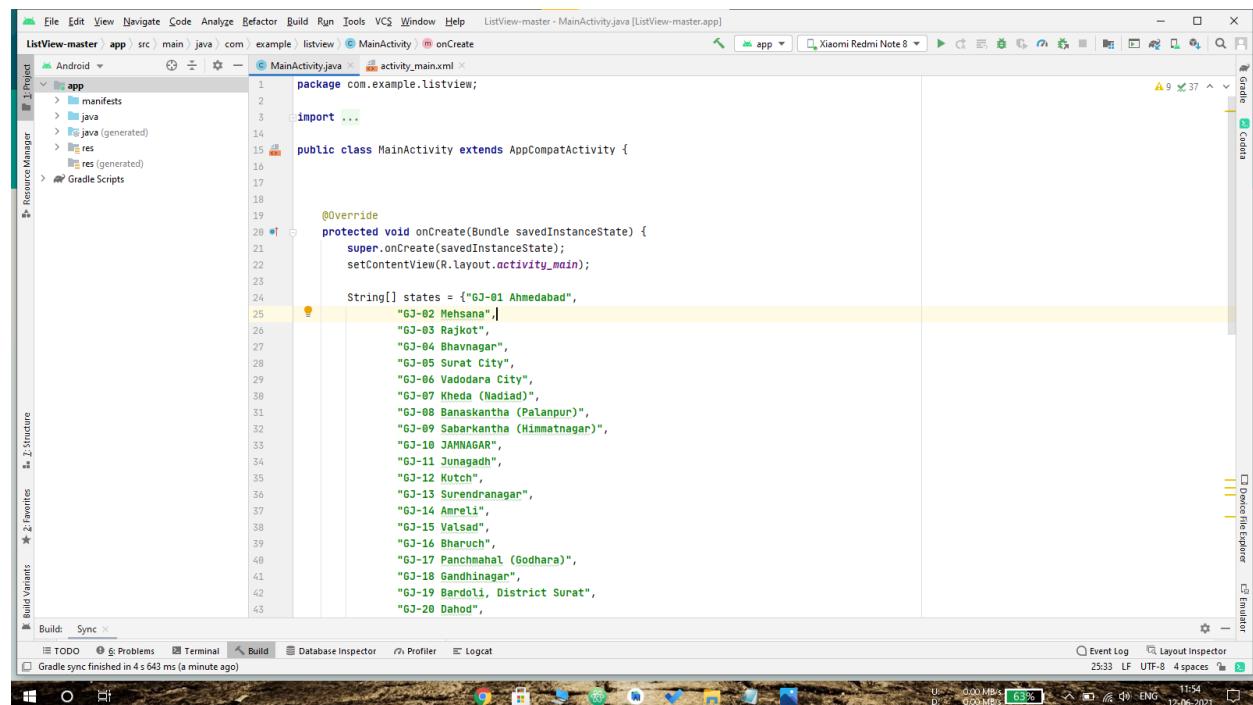
An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter holds the data and send the data to adapter view.

We can use the adapter like:

```
ArrayAdapter adapter = new ArrayAdapter(context, R.layout.ListView,
StringArray);
```

### Code:

Main activity:



A screenshot of the Android Studio IDE. The main window shows the Java code for `MainActivity.java`. The code defines a class `MainActivity` that extends `AppCompatActivity`. It overrides the `onCreate` method to set the content view to `R.layout.activity_main`. Inside this method, a string array `states` is defined, containing 20 entries representing districts of Gujarat, starting with "GJ-01 Ahmedabad" and ending with "GJ-20 Dahod". The Android Studio interface includes toolbars, a navigation bar, and various inspection windows on the right side.

```
package com.example.listview;
import ...;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    String[] states = {"GJ-01 Ahmedabad",
        "GJ-02 Mehsana",
        "GJ-03 Rajkot",
        "GJ-04 Bhavnagar",
        "GJ-05 Surat City",
        "GJ-06 Vadodara City",
        "GJ-07 Kheda (Nadiad)",
        "GJ-08 Banaskantha (Palanpur)",
        "GJ-09 Sabarkantha (Himmatnagar)",
        "GJ-10 JAMNAGAR",
        "GJ-11 Junagadh",
        "GJ-12 Kutch",
        "GJ-13 Surendranagar",
        "GJ-14 Amreli",
        "GJ-15 Valsad",
        "GJ-16 Bharuch",
        "GJ-17 Panchmahal (Godhra)",
        "GJ-18 Gandhinagar",
        "GJ-19 Bardoli, District Surat",
        "GJ-20 Dahod",
    };
}
```

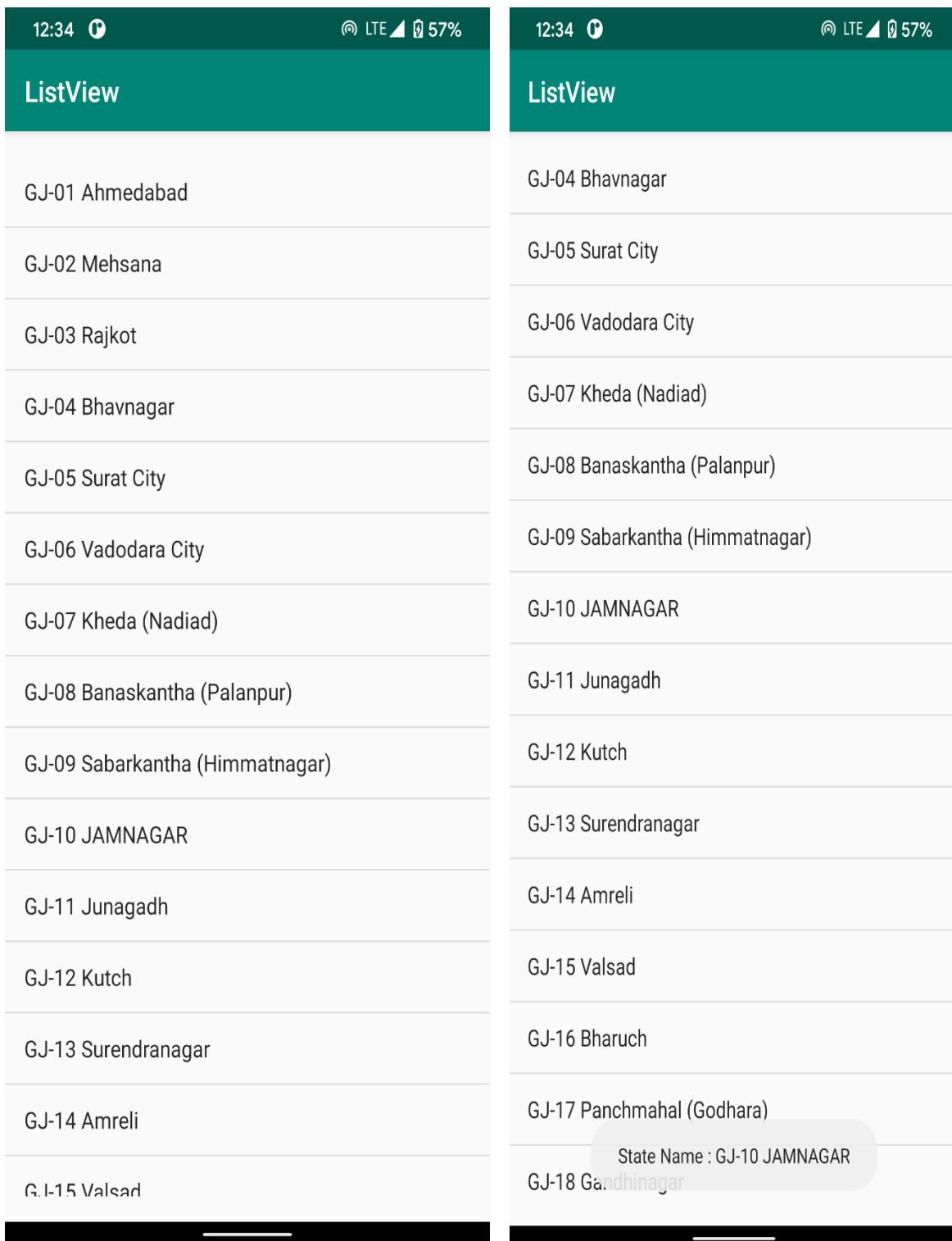
## Activity\_main.xml:

The screenshot shows the Android Studio interface with the project 'ListView-master' open. The main window displays the XML file 'activity\_main.xml' under the 'layout' directory. The XML code defines a ConstraintLayout containing a ListView with specific dimensions and margins. To the right of the code editor, a preview window shows a list of items labeled 'Item 1' through 'Item 11'. The bottom right corner of the screen shows the system status bar with battery level, signal strength, and time.

```
<?xml version="1.0" encoding="UTF-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

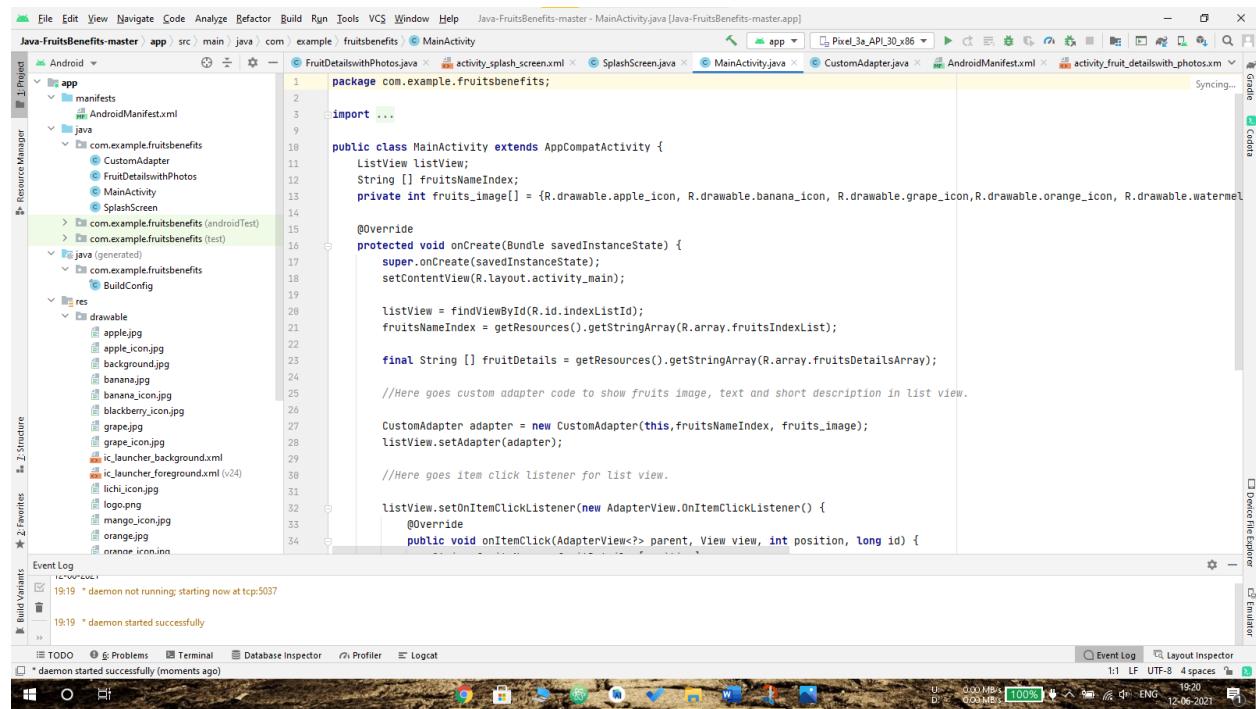
    <ListView
        android:id="@+id/useenlist"
        android:layout_width="395dp"
        android:layout_height="715dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

## Output:



## 6.1 -Custom view list:

### Main activity:



The screenshot shows the Android Studio interface with the Java code for `MainActivity.java` open. The code implements a `CustomAdapter` to display fruit details with photos in a list view. The `ListView` is defined in the XML layout file `activity_main.xml`.

```
package com.example.fruitsbenefits;

import ...

public class MainActivity extends AppCompatActivity {
    ListView listView;
    String [] fruitsNameIndex;
    private int fruits_image[] = {R.drawable.apple_icon, R.drawable.banana_icon, R.drawable.grape_icon, R.drawable.orange_icon, R.drawable.watermelon};

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

        listView = findViewById(R.id.indexListId);
        fruitsNameIndex = getResources().getStringArray(R.array.fruitsIndexList);

        final String [] fruitDetails = getResources().getStringArray(R.array.fruitsDetailsArray);

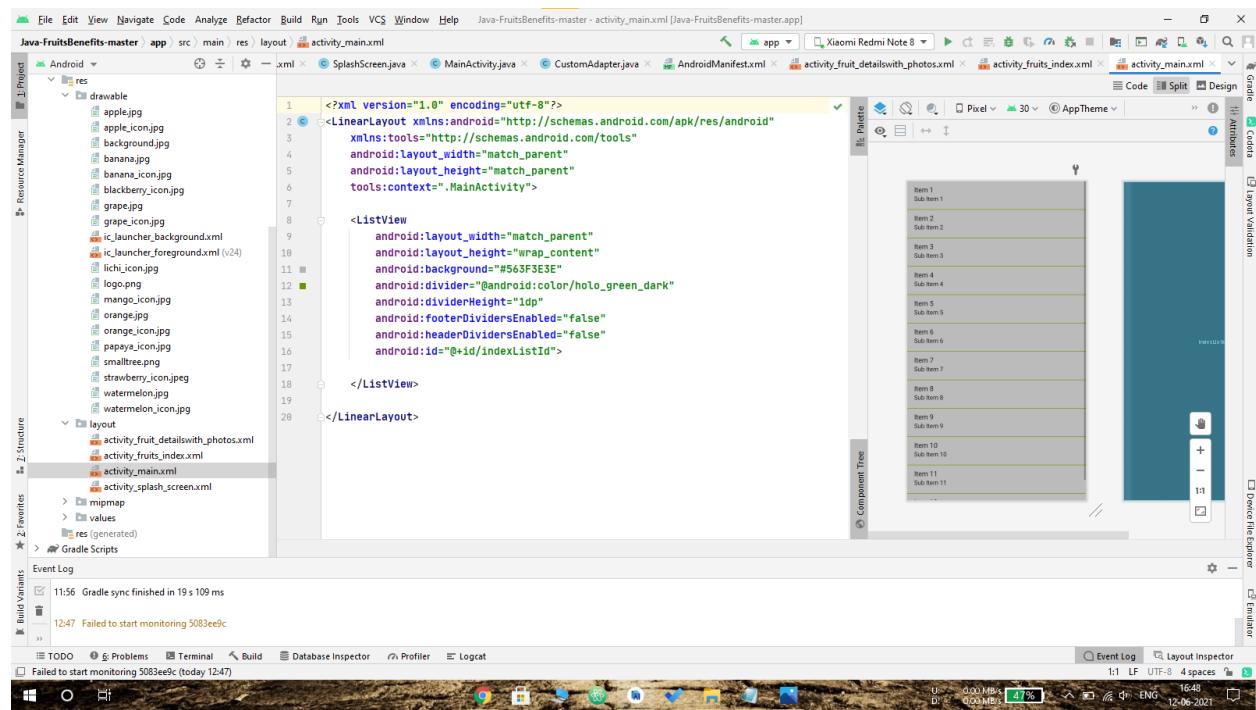
        //Here goes custom adapter code to show fruits image, text and short description in list view.

        CustomAdapter adapter = new CustomAdapter(this,fruitsNameIndex, fruits_image);
        listView.setAdapter(adapter);

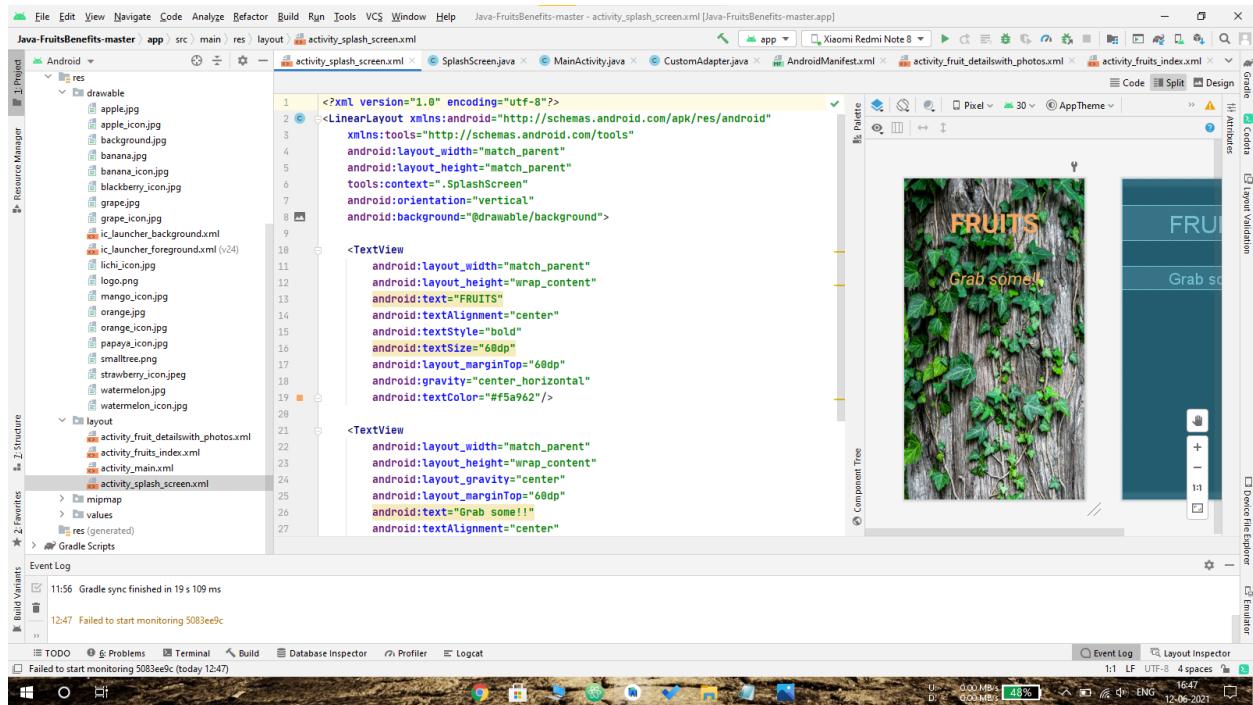
        //Here goes item click listener for list view.

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Intent intent = new Intent(getApplicationContext(),FruitDetailsWithPhotos.class);
                intent.putExtra("fruit_name",fruitsNameIndex[position]);
                intent.putExtra("fruit_details",fruitDetails[position]);
                intent.putExtra("fruit_image",fruits_image[position]);
                startActivity(intent);
            }
        });
    }
}
```

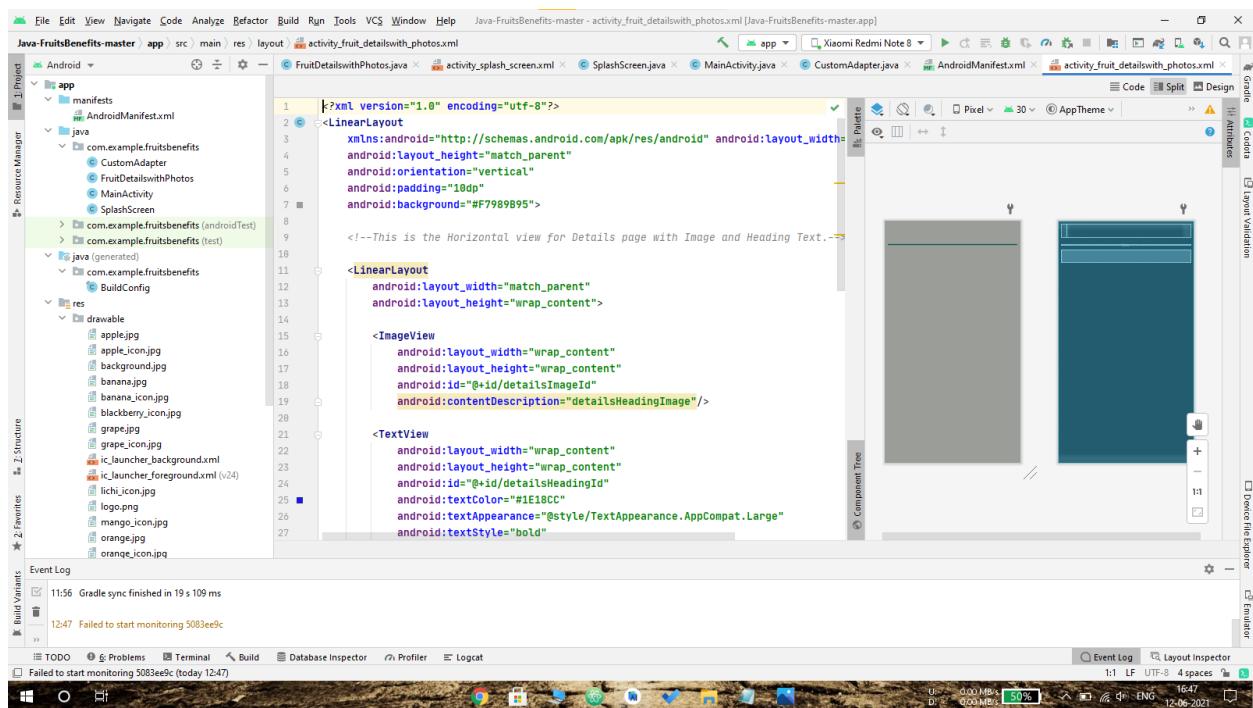
### Activity\_main.xml:



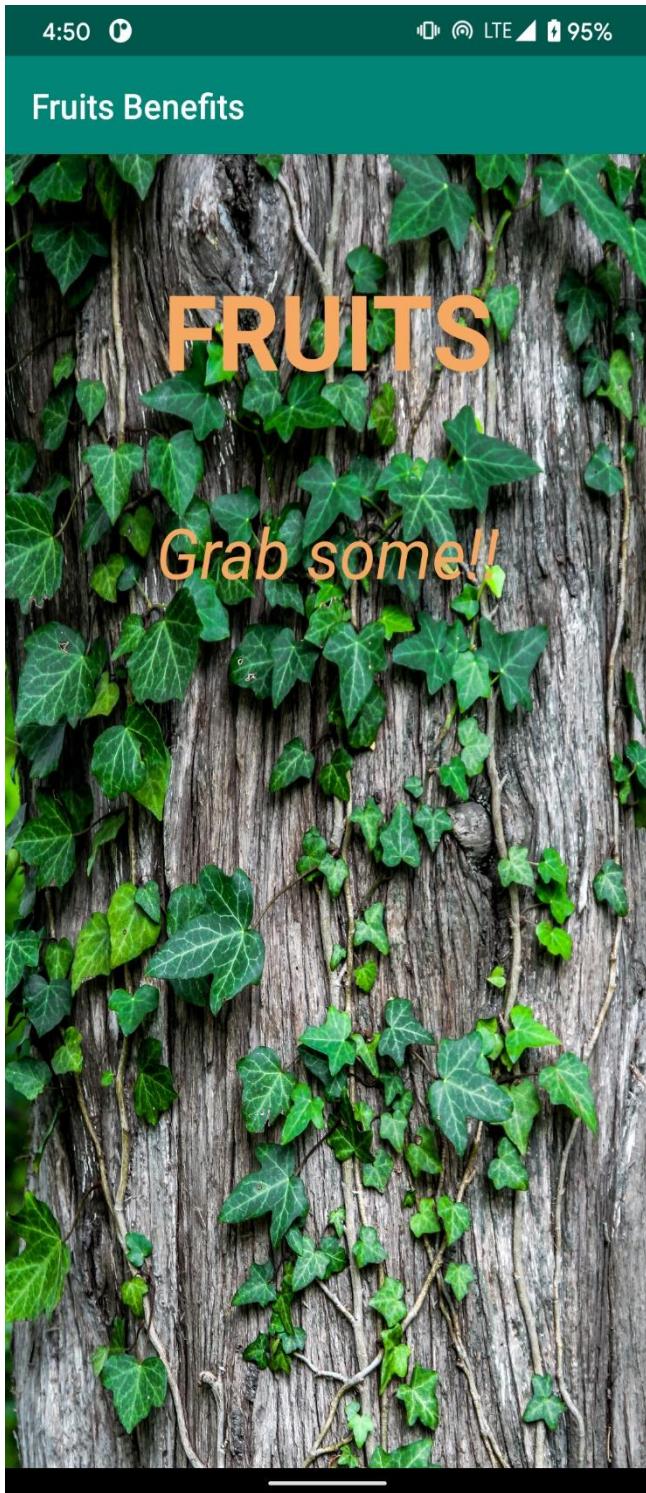
## Activity\_splash\_screen.xml:



## Activity\_fruit\_details\_with\_photos.xml:



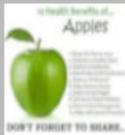
Output:



4:50

LTE 95%

## Fruits Benefits



### Apple

[Click for Details](#)

### Banana

[Click for Details](#)

### Grape

[Click for Details](#)

### Orange

[Click for Details](#)

### Watermelon

[Click for Details](#)

### Papaya

[Click for Details](#)

### Blackberry

[Click for Details](#)

### Strawberry

[Click for Details](#)

### Mango

[Click for Details](#)

4:50

LTE 95%

## Fruits Benefits



### Orange

Orange (fruit) An orange is a type of citrus fruit that people often eat or they can sniff the skin when grinded into a smooth powder. Oranges are a very good source of vitamin . Orange juice is an important part of many people's breakfast.

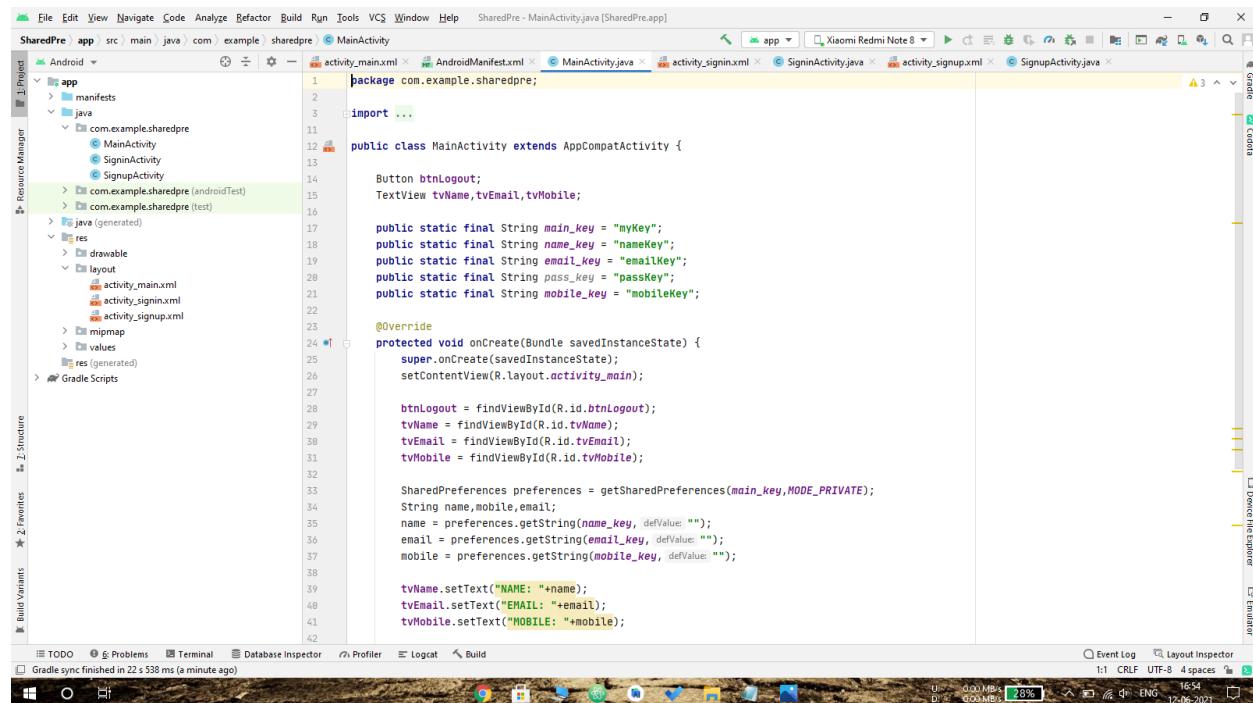
## 7 - Shared Preferences:

Using Shared Preferences, we can store the data on local device for temporary purpose if user clear the data of app, then the stored data of shared preferences are removed and users have to login again.

Using Shared Preferences, we can make user stay logged IN in the application.

### Code:

Main activity:



The screenshot shows the Android Studio interface with the project 'SharedPre' open. The 'MainActivity.java' file is selected in the editor. The code implements logic to read and set shared preferences for name, email, and mobile number. It also updates three TextViews ('tvName', 'tvEmail', 'tvMobile') based on the preference values.

```
package com.example.sharedpre;

import ...

public class MainActivity extends AppCompatActivity {

    Button btnLogout;
    TextView tvName, tvEmail, tvMobile;

    public static final String main_key = "myKey";
    public static final String name_key = "nameKey";
    public static final String email_key = "emailKey";
    public static final String pass_key = "passKey";
    public static final String mobile_key = "mobileKey";

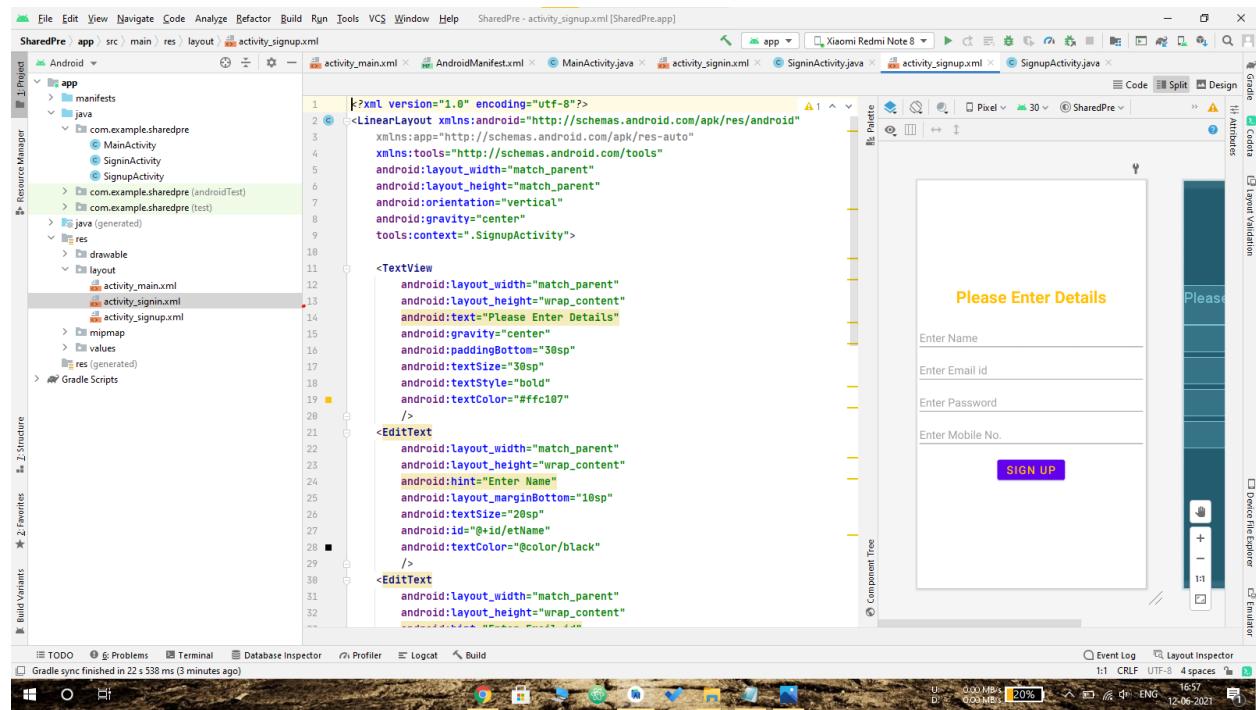
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnLogout = findViewById(R.id.btnLogout);
        tvName = findViewById(R.id.tvName);
        tvEmail = findViewById(R.id.tvEmail);
        tvMobile = findViewById(R.id.tvMobile);

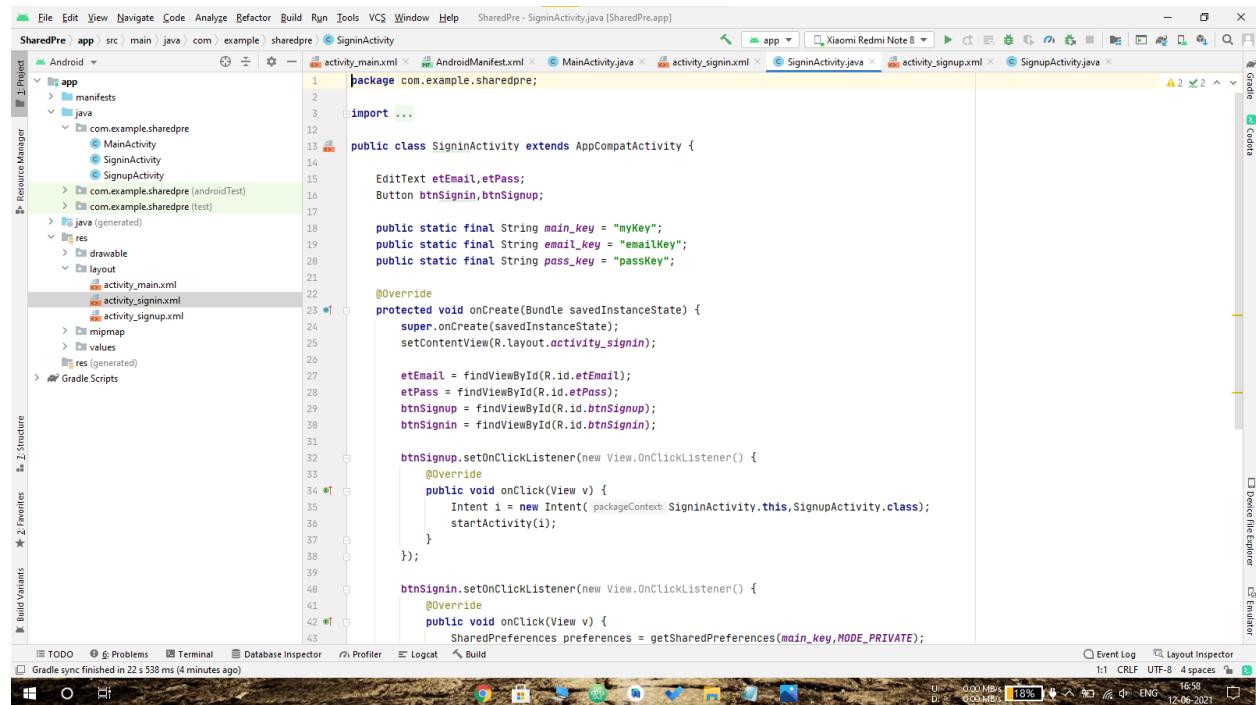
        SharedPreferences preferences = getSharedPreferences(main_key, MODE_PRIVATE);
        String name, mobile, email;
        name = preferences.getString(name_key, defaultValue: "");
        email = preferences.getString(email_key, defaultValue: "");
        mobile = preferences.getString(mobile_key, defaultValue: "");

        tvName.setText("NAME: "+name);
        tvEmail.setText("EMAIL: "+email);
        tvMobile.setText("MOBILE: "+mobile);
    }
}
```

## Activity\_signup.xml:

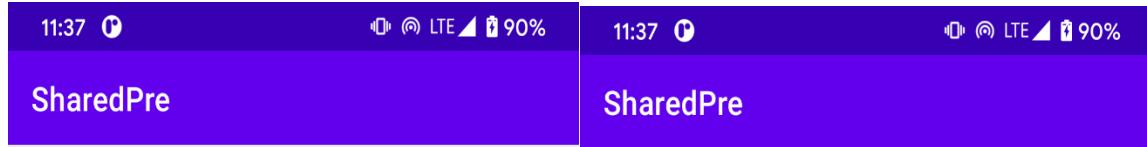


## Signin activity.java:



Output:





## USER DATA

Please Log In

cr433270@gmail.com

**NAME: PRIYESH  
CHIKHALIYA**

....

**EMAIL: cr433270@gmail  
.com**

**SIGN IN**

**MOBILE: 6353711716**

**SIGN UP**

**LOG OUT**



## 8 - Phone:

Using intent's ACTION\_CALL property we can make call from calling app.

```
Intent intent = new Intent(Intent.ACTION_CALL);
```

We have to set the mobile number to URI and it redirect to calling app.

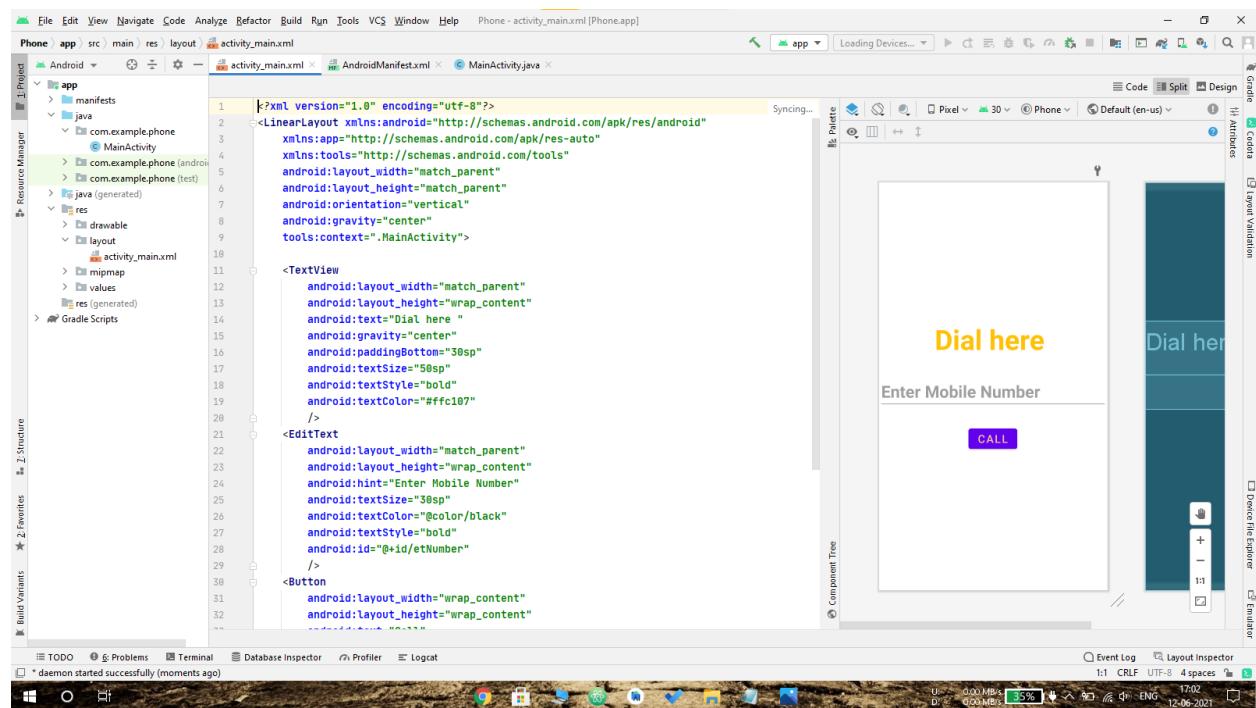
```
intent.setData(Uri.parse("number:"+phNumber));
```

and we have to add "uses-permission" for call\_phone in AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>
```

## Code:

Main activity:



## Activity\_main.xml:

The screenshot shows the Android Studio interface with the project 'Phone' open. The 'activity\_main.xml' layout file is selected in the editor. The code in the XML file is as follows:

```
<?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"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dial here"
        android:gravity="center"
        android:paddingBottom="30sp"
        android:textSize="30sp"
        android:textStyle="bold"
        android:textColor="#ffcc107" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Mobile Number"
        android:textSize="30sp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:id="@+id/etNumber" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CALL" />
</LinearLayout>
```

The preview window on the right shows a mobile application interface with a yellow header containing the text 'Dial here'. Below it is a white input field with a placeholder 'Enter Mobile Number'. At the bottom is a purple button labeled 'CALL'. The status bar at the top of the screen shows the time as 12:55, signal strength, battery level at 100%, and connectivity information.

Output:



Dial here

Enter Mobile Number

CALL



Phone



Calling...

A Yash Bro

Home 90167 55060

Dial here

9016755060



Mute



Keypad



Speaker



Add call



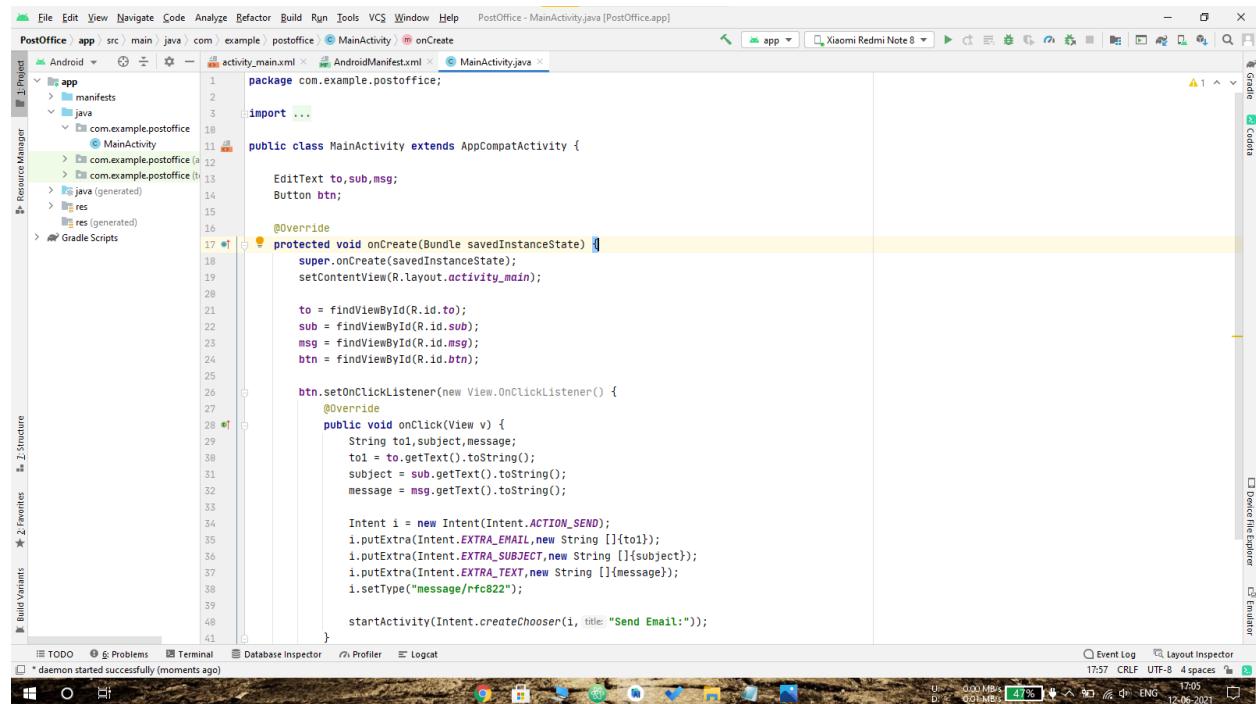
## 9 – PostOffice:

To send email from our android application we need following Intent actions.

**Intent.ACTION\_SEND, Intent.EXTRA\_EMAIL, Intent.EXTRA\_SUBJECT,  
Intent.EXTRA\_TEXT, Intent.createChooser().**

### Code:

#### Main activity:



The screenshot shows the Android Studio interface with the code editor open to MainActivity.java. The code implements an Activity that handles email sending via Intent. It includes methods for finding views by ID and setting up an OnClickListener for a button. The Intent is configured with ACTION\_SEND, EXTRA\_EMAIL, EXTRA\_SUBJECT, and EXTRA\_TEXT extra fields, and a chooser is started with the title "Send Email".

```
package com.example.postoffice;

import ...

public class MainActivity extends AppCompatActivity {

    EditText to,sub,msg;
    Button btn;

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

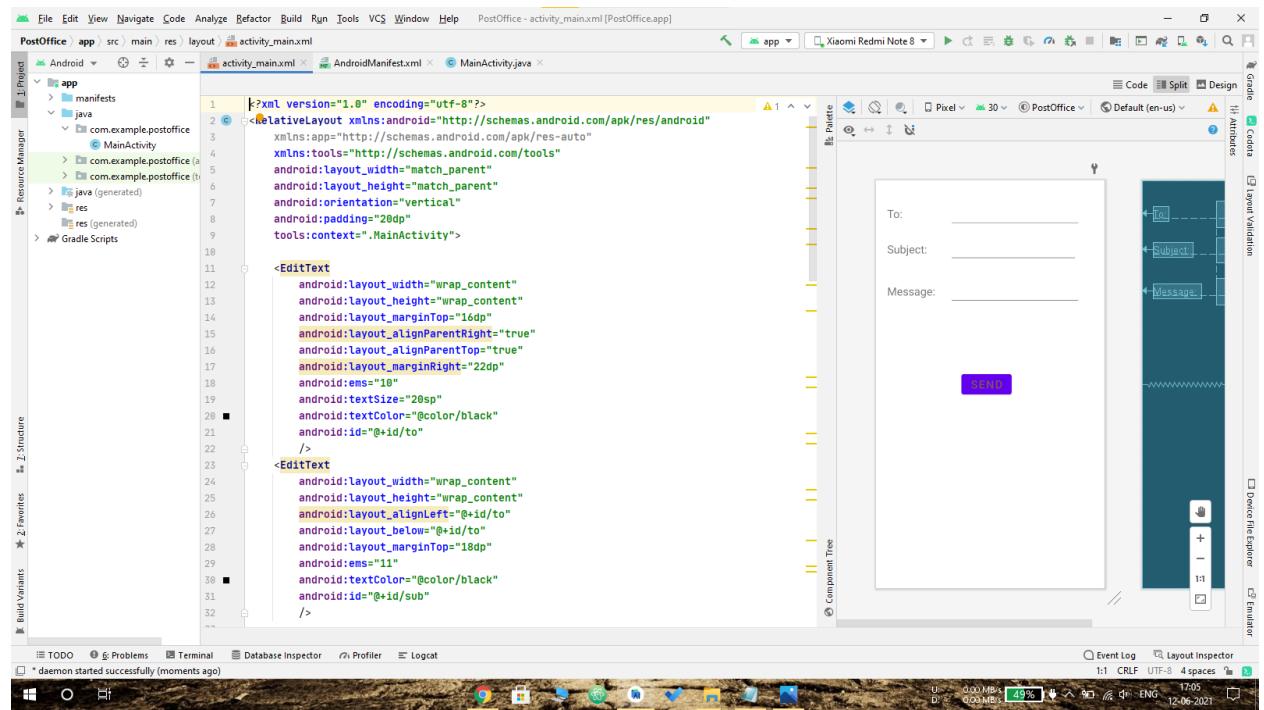
        to = findViewById(R.id.to);
        sub = findViewById(R.id.sub);
        msg = findViewById(R.id.msg);
        btn = findViewById(R.id.btn);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String to1,subject,message;
                to1 = to.getText().toString();
                subject = sub.getText().toString();
                message = msg.getText().toString();

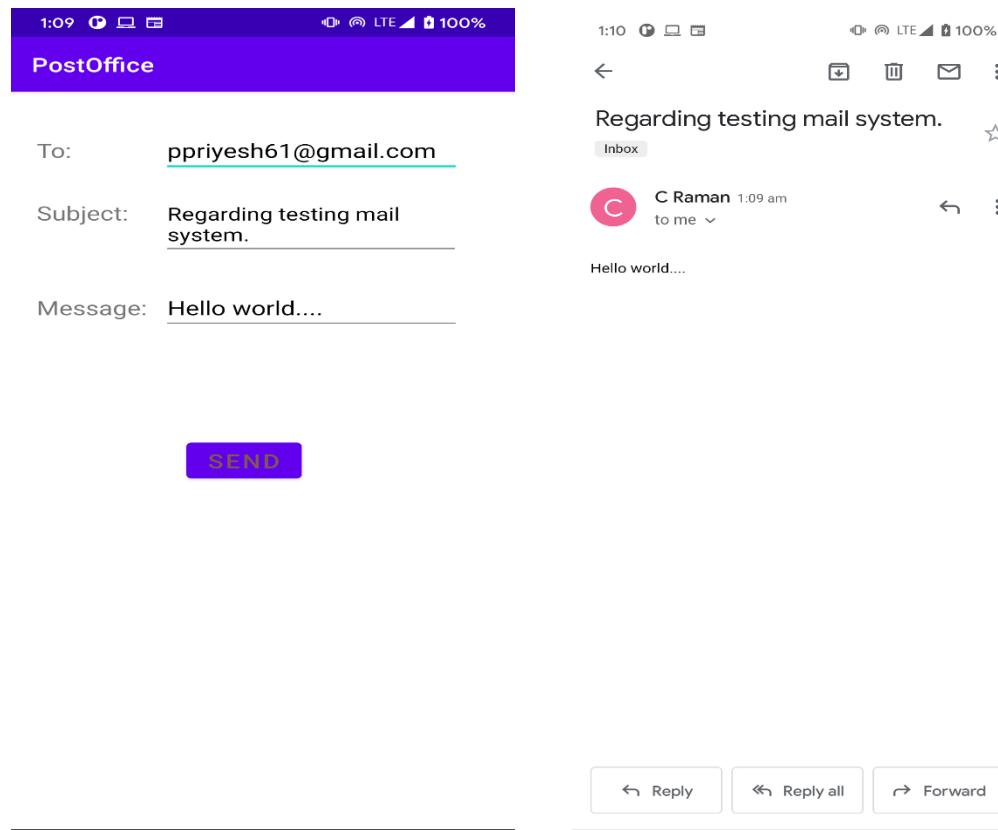
                Intent i = new Intent(Intent.ACTION_SEND);
                i.putExtra(Intent.EXTRA_EMAIL,new String []{to1});
                i.putExtra(Intent.EXTRA_SUBJECT,new String []{subject});
                i.putExtra(Intent.EXTRA_TEXT,new String []{message});
                i.setType("message/rfc822");

                startActivity(Intent.createChooser(i, title: "Send Email:"));
            }
        });
    }
}
```

## Activity\_main.xml:



## Output:

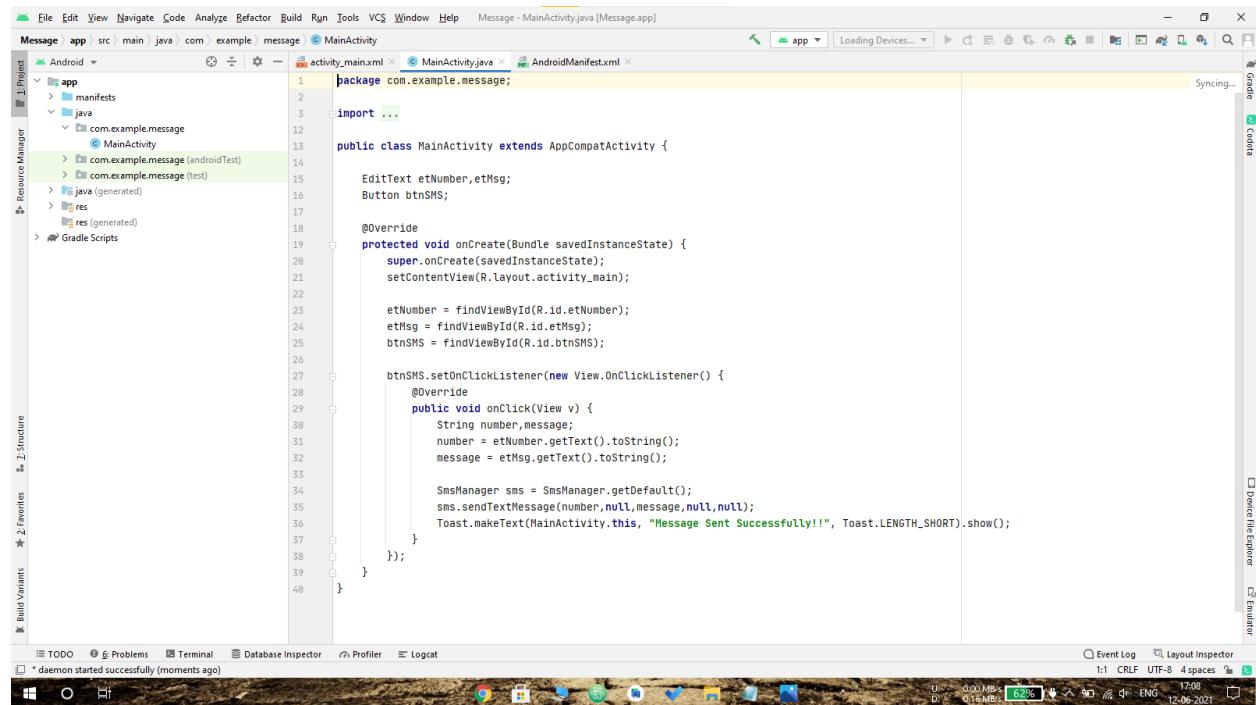


## 10 – Message:

- We can send SMS from our android application in two ways either by using SMSManager api or Intents based on our requirements.
- If we use SMSManager api, it will directly send SMS from our application.
- In case if we use Intent with proper action (ACTION\_VIEW), it will invoke built-in SMS app to send SMS from our application

### Code:

#### Main activity:



The screenshot shows the Android Studio interface with the project 'Message' open. The main window displays the 'MainActivity.java' file under the 'src/main/java/com/example/message' package. The code implements a button click listener to send an SMS using the SmsManager API. The code is as follows:

```
package com.example.message;

import ...

public class MainActivity extends AppCompatActivity {

    EditText etNumber, etMsg;
    Button btnSMS;

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

        etNumber = findViewById(R.id.etNumber);
        etMsg = findViewById(R.id.etMsg);
        btnSMS = findViewById(R.id.btnSMS);

        btnSMS.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String number, message;
                number = etNumber.getText().toString();
                message = etMsg.getText().toString();

                SmsManager sms = SmsManager.getDefault();
                sms.sendTextMessage(number, null, message, null, null);
                Toast.makeText(MainActivity.this, "Message Sent Successfully!!", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

## Activity\_main.xml:

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

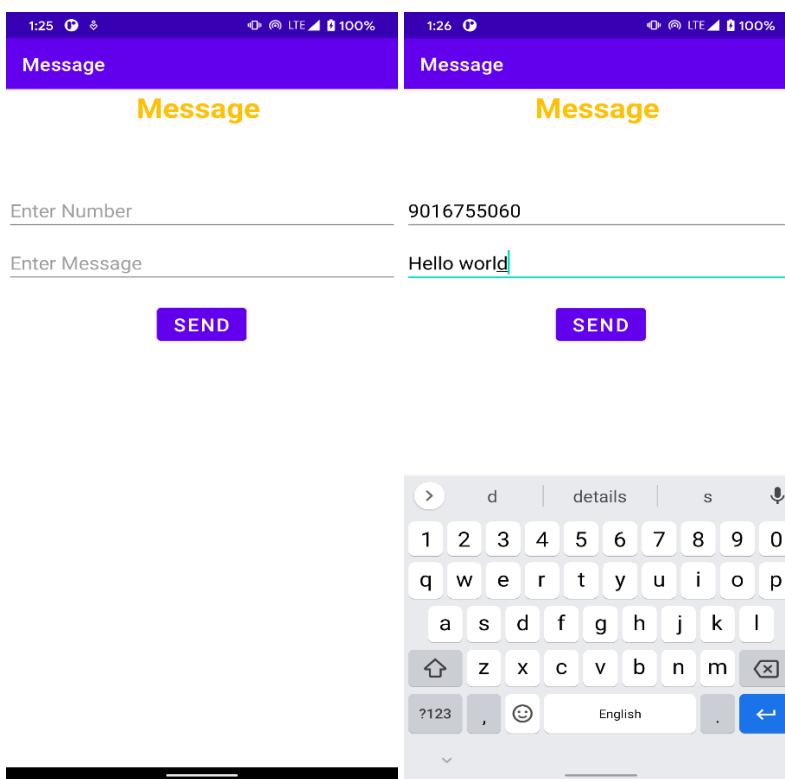
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Message"
        android:gravity="center"
        android:paddingBottom="30sp"
        android:textSize="30sp"
        android:textStyle="bold"
        android:textColor="#ffcc107"
        />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40sp"
        android:hint="Enter Number"
        android:textSize="28sp"
        android:textColor="@color/black"
        android:id="@+id/etNumber"
        />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10sp"
        />

```

## Output:



## 11 –Player

To Play music we have to define MediaPlayer:

**MediaPlayer mPlayer = MediaPlayer.create(this, R.raw.audio\_file\_name);**

After that We can start Media Player using start() Method.

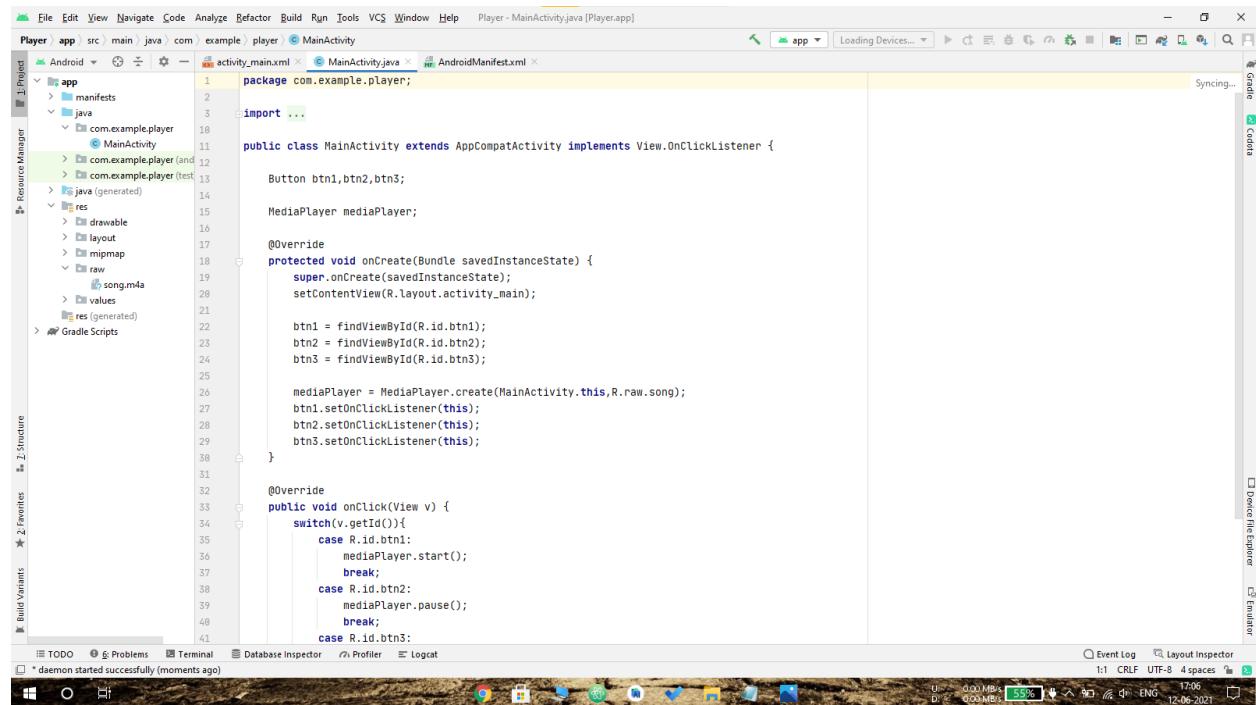
**mPlayer.start();**

**pause()** method is used to pause the Audio.

**stop()** method is used to stop the Audio.

### Code:

Main activity:



The screenshot shows the Android Studio interface with the code editor open to MainActivity.java. The code implements a MediaPlayer to play a song from raw resources when buttons are clicked.

```
package com.example.player;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    Button btn1,btn2,btn3;
    MediaPlayer mediaPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn1 = findViewById(R.id.btn1);
        btn2 = findViewById(R.id.btn2);
        btn3 = findViewById(R.id.btn3);
        mediaPlayer = MediaPlayer.create(MainActivity.this,R.raw.song);
        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);
        btn3.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch(v.getId()){
            case R.id.btn1:
                mediaPlayer.start();
                break;
            case R.id.btn2:
                mediaPlayer.pause();
                break;
            case R.id.btn3:
                mediaPlayer.stop();
                break;
        }
    }
}
```

## Activity\_main.xml:

The screenshot shows the Android Studio interface with the project 'Player' open. The left sidebar shows the project structure with 'activity\_main.xml' selected. The main area displays the XML code for the layout:

```
<?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"
    android:gravity="center"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Music Player"
        android:textColor="#3c8dad"

        android:textSize="40sp"
        android:gravity="center"
        android:textStyle="bold"
    />

    <Button
        android:layout_width="150dp"
        android:layout_marginTop="20sp"
        android:text="Start Music"
        android:layout_height="wrap_content"
        android:id="@+id/btn1"
    />
    <Button
        android:layout_width="150dp"
        android:text="Pause Music"
        android:layout_height="wrap_content"
    />

```

The right side shows the 'Design' tab of the Layout Editor with a preview of the layout. The title bar says 'Music Player'. It contains a central text view with bold, black, 40sp text reading 'Music Player'. Below it are two purple rectangular buttons: 'START MUSIC' and 'PAUSE MUSIC'. A third button, 'STOP MUSIC', is visible but currently disabled.

Output:



## Music Player

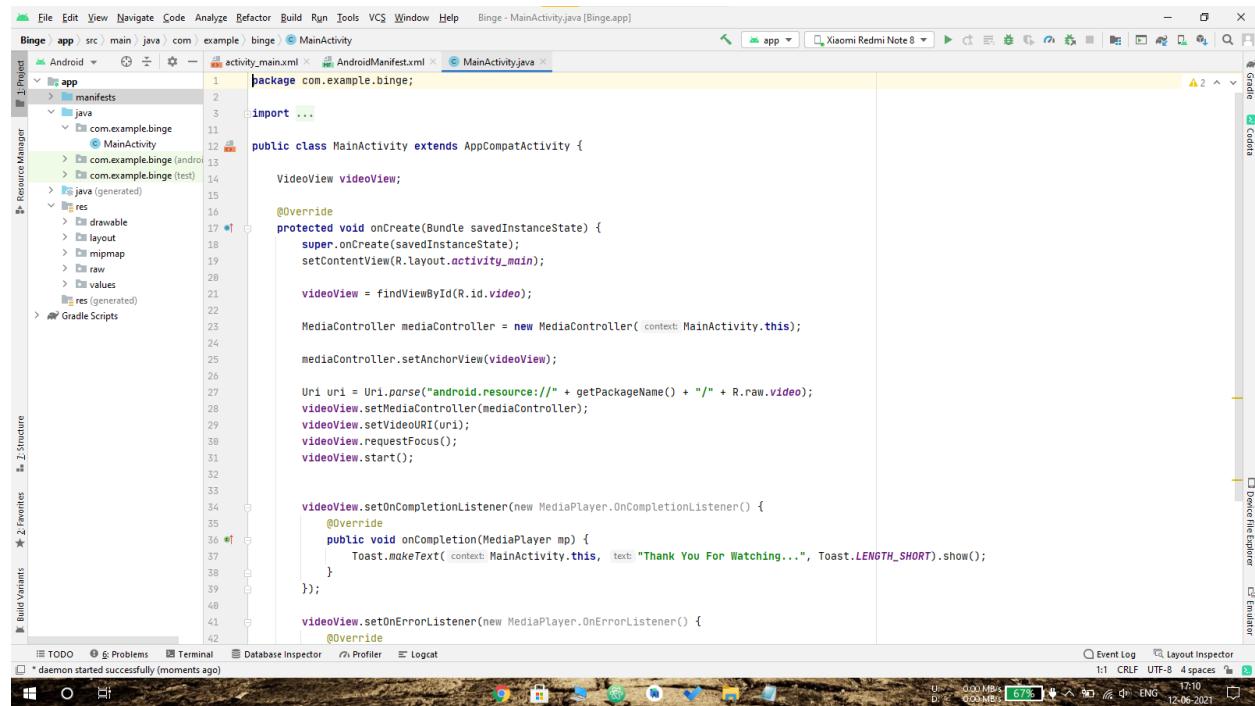
START MUSIC  
PAUSE MUSIC  
STOP MUSIC

## 12 – Binge:

In android , VideoView component and MediaController class is used to implement the video player in android applications to play the videos with multiple playback options, such as play, pause, forward, backward, etc.

### Code:

#### Main activity:



The screenshot shows the Android Studio interface with the code editor open to MainActivity.java. The code implements a video player using a VideoView and a MediaController. It includes methods for handling video completion and errors.

```
package com.example.binge;

import ...

public class MainActivity extends AppCompatActivity {

    VideoView videoView;
    MediaController mediaController = new MediaController(context: MainActivity.this);

    mediaController.setAnchorView(videoView);

    Uri uri = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.video);
    videoView.setMediaController(mediaController);
    videoView.setVideoURI(uri);
    videoView.requestFocus();
    videoView.start();

    videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            Toast.makeText(context: MainActivity.this, text: "Thank You For Watching...", Toast.LENGTH_SHORT).show();
        }
    });

    videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
        @Override
        public boolean onError(MediaPlayer mp, int what, int extra) {
            return false;
        }
    });
}
```

## Activity\_main.xml:

The screenshot shows the Android Studio interface with the 'activity\_main.xml' layout file open. The code editor displays the XML structure:

```
<?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">

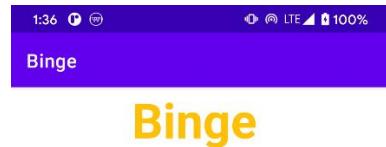
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:paddingBottom="30sp"
        android:textSize="50sp"
        android:textStyle="bold"
        android:textColor="#ffcc00"/>

    <VideoView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="30dp"
        android:id="@+id/video"/>

</LinearLayout>
```

The preview window on the right shows a dark teal background with the word 'Binge' in yellow at the top. Below it is a large gray rectangular area representing the VideoView component.

## Output:



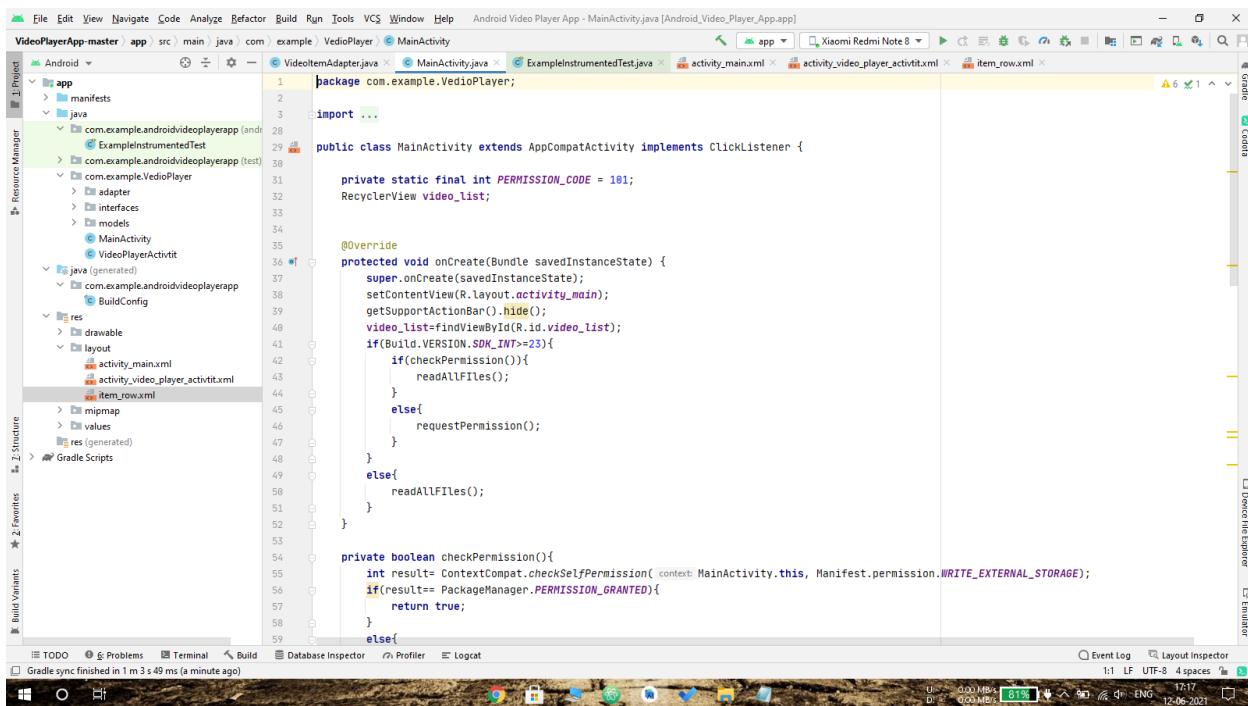
# Project

## Video Player app-master:

In this it List all .mp4 files from internal Storage and we can play video of our choice in Landscape Mode.

## Code:

### Main activity:



The screenshot shows the Android Studio interface with the code editor open to the `MainActivity.java` file. The code implements a RecyclerView to display a list of videos from internal storage. It includes methods for checking permissions and reading files, and handles different API levels for permission requests.

```
package com.example.VedioPlayer;

import ...

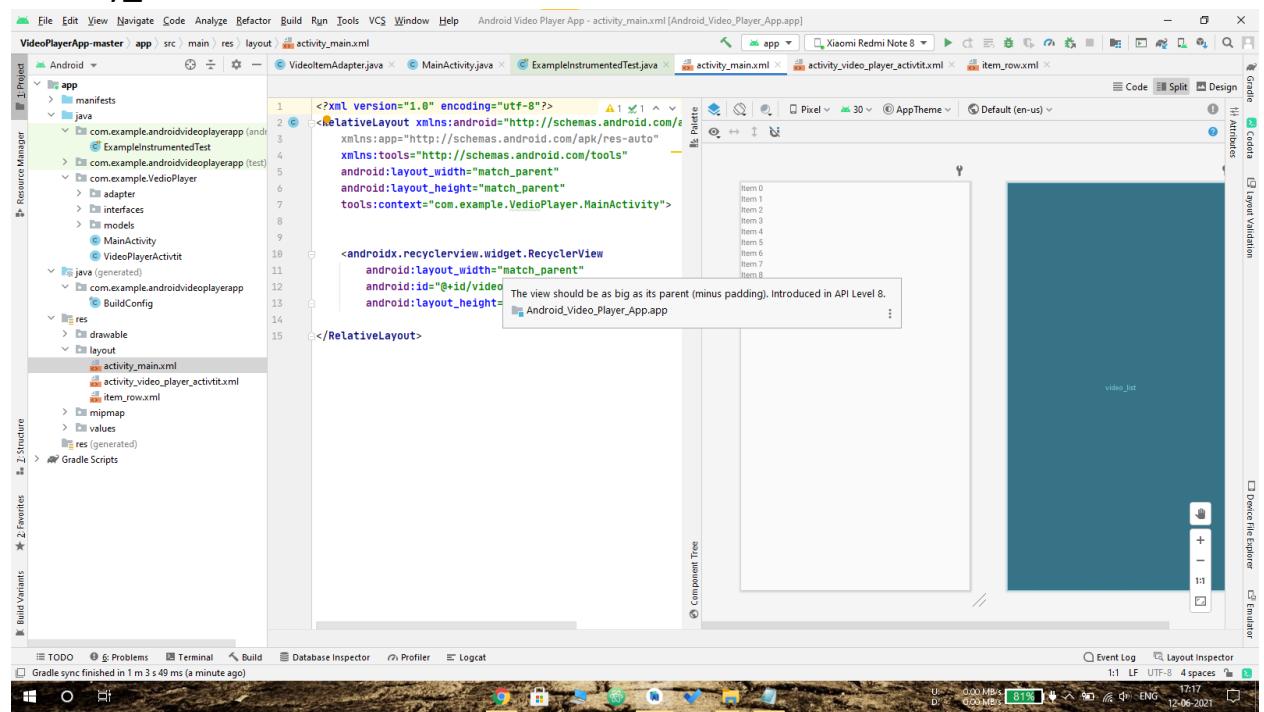
public class MainActivity extends AppCompatActivity implements ClickListener {
    private static final int PERMISSION_CODE = 101;
    RecyclerView video_list;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();
        video_list=findViewById(R.id.video_list);
        if(Build.VERSION.SDK_INT>=23){
            if(checkPermission()){
                readAllFiles();
            }
            else{
                requestPermission();
            }
        }
        else{
            readAllFiles();
        }
    }

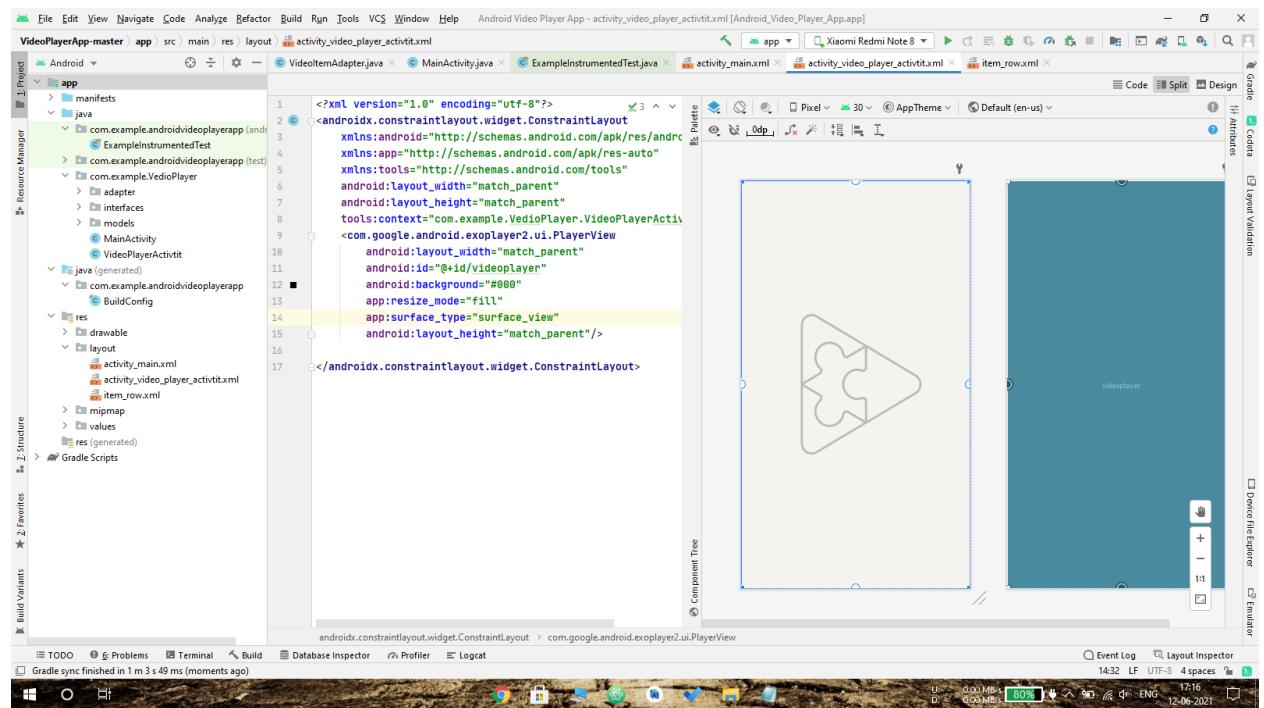
    private boolean checkPermission(){
        int result= ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.WRITE_EXTERNAL_STORAGE);
        if(result== PackageManager.PERMISSION_GRANTED){
            return true;
        }
        else{

```

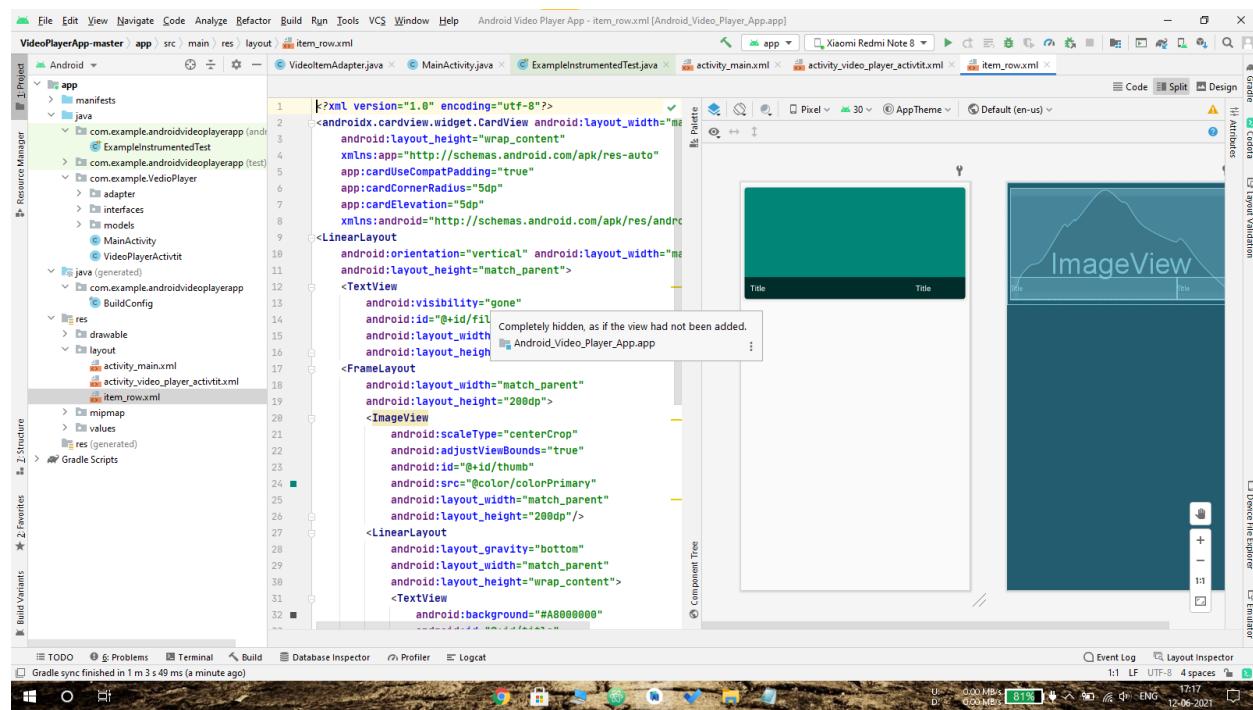
## Activity\_main.xml:



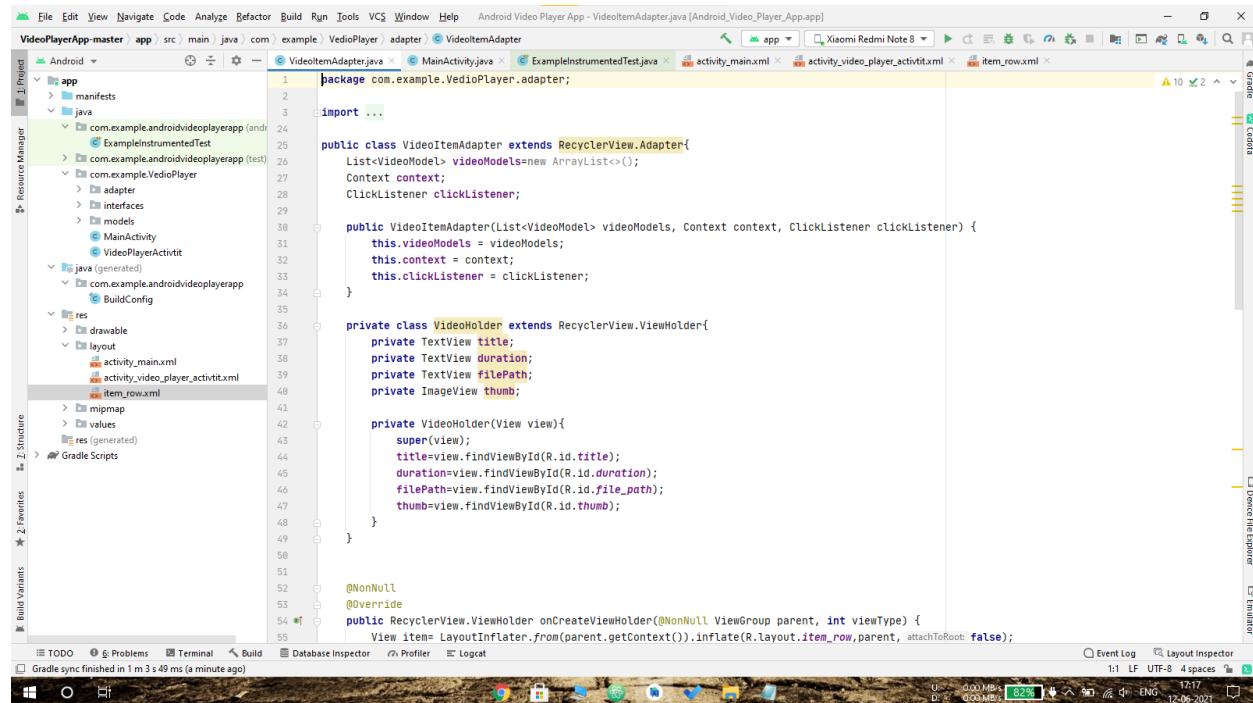
## Activity\_video\_player\_activeit.xml:



## Item\_row.xml:



## VideolItemAdapter.java:



## Output:

