**MOBILE APPLICATION DEVELOPMENT**

# LABSET PROGRAMS

1. **Creating “Hello world” Application.**
2. **Creating an application that displays message based on the screen orientation.**
3. **Create an application to develop Login window using UI controls**
4. **Create an application to implement new activity using explicit intent, implicit intent and content providers.**
5. **Create an application that displays custom designed opening screen.**
6. **Create an UI with all views.**
7. **Create menu in application.**
8. **Read/write the local data.**
9. **Create/ Read/ write data with database (SQLite).**
10. **Create an application to send SMS and receive SMS.**
11. **Create an application to send an e-mail.**
12. **Display a Map based on Current/ given location.**
13. **Create a sample application with login module (check username and password) on successful Login change Text view “Login Successful”, on login fail alert using Toast “Login fail”.**
14. **Learn to deploy android application.**

**Program 1:**

1. **Creating “Hello world” Application.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button.

4**. activity\_main.xml**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

android:textStyle="bold"

android:textColor="@color/black"

android:textSize="32sp"

android:background="#CE0EEF"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**5. MainActivity.java**

package com.example.a1stprogram;

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);

}

}

**6. Output:**

****

**Imp methods:**

* **AppCompatActivity** is a class in the Android framework that is part of the AndroidX library. It is a base class for activities that wish to use some of the newer platform features on older Android devices. By extending AppCompatActivity, you can use many of the modern Android components and features while maintaining backward compatibility with older versions of Android.
* **savedInstanceState** is a parameter in the Android activity lifecycle methods such as onCreate, onSaveInstanceState, and onRestoreInstanceState. It is a Bundle object that contains the activity's previously saved state. This is particularly useful for restoring the state of an activity when it is recreated, such as during a configuration change (e.g., screen rotation) or when the activity is temporarily destroyed by the system to free up resources.
*  **State Preservation**: savedInstanceState allows you to save and restore the state of your activity. This includes UI elements, user input, and other relevant data that you want to maintain across configuration changes or when the activity is recreated.
*  **Bundle Object**: It is a Bundle, which is a key-value store that can hold various types of data, including primitive types, strings, arrays, and other Bundle objects.
* The **setContentView method** in Android is used to set the layout for an activity. This method is called within the onCreate method of an activity to define the UI components that will be displayed to the user. By specifying the layout resource file, you tell the activity which XML file to use for its UI.

1. **Creating an application that displays message based on the screen orientation.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button.

4**.** Create two Button resources in activity\_main.xml and update the following code.

**5. activity\_main.xml**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<Button

android:id="@+id/por"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:text="Portrait"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:ignore="MissingConstraints" />

<Button

android:id="@+id/lan"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/por"

android:layout\_centerInParent="true"

android:text="Landscape"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>

**6. MainActivity.java**

package com.example.a2ndprogram;

import android.content.pm.ActivityInfo;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

Button l, p;

l=findViewById(R.id.lan);

p=findViewById(R.id.por);

l.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);

Toast.makeText(MainActivity.this, "Hey! We are in Landscape orientation", Toast.LENGTH\_SHORT).show();

}

});

p.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT);

Toast.makeText(MainActivity.this, "Hey! We are in Portrait orientation",

Toast.LENGTH\_SHORT).show();

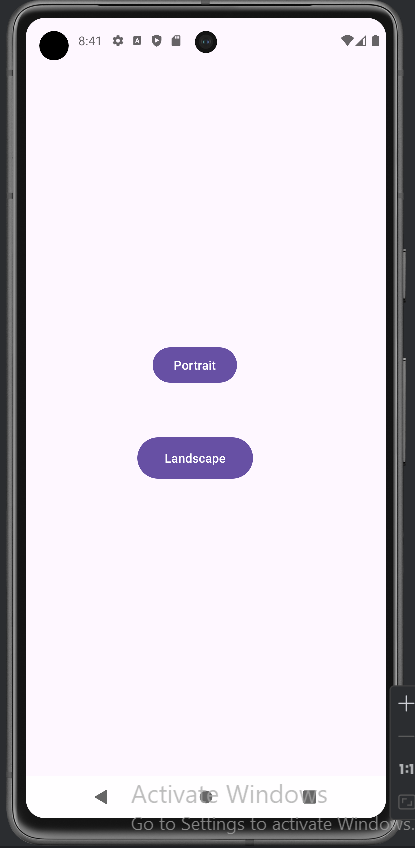
}

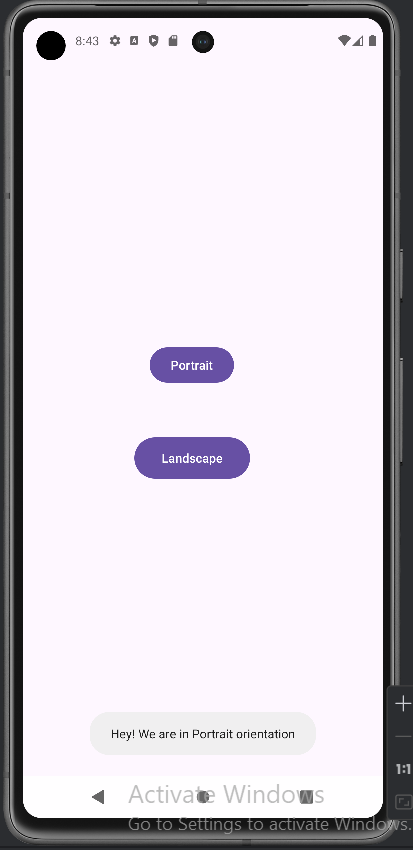
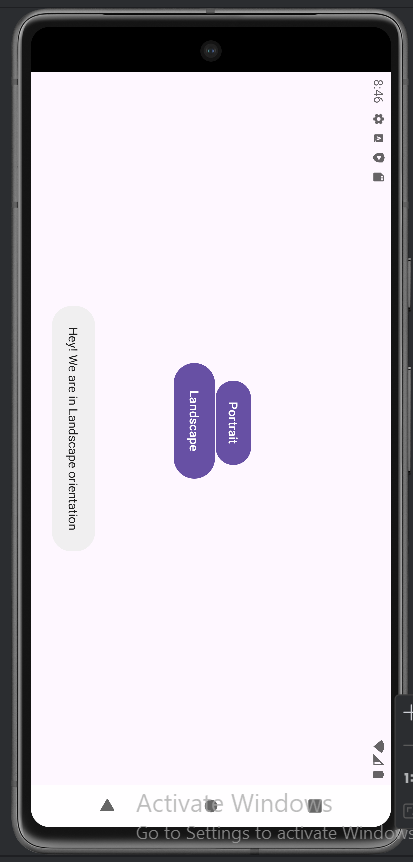
});

}

}

**7. Output:**

****

** **

**Imp methods:**

* **The findViewById method** in Android is used to find and return a view that was identified by an ID attribute in the XML layout file. This method is crucial for accessing UI components (such as buttons, text views, etc.) defined in XML so that you can manipulate them in your Java or Kotlin code.
* **The OnClickListener** interface in Android is used to define the action that occurs when a user clicks on a view (e.g., a button). To handle click events, you implement this interface and override its onClick method.
* The s**etRequestedOrientation** method in Android is used to lock an activity's orientation to a specific one, such as portrait or landscape. This is particularly useful when you want to control the screen orientation of your app to ensure a consistent user experience.
* The **Toast class** in Android is used to display a short message to the user. A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive.
* The **Toast.LENGTH\_SHORT** constant in Android is used to specify the duration for which a toast message is displayed. There are two predefined duration constants in the Toast class:
* **Toast.LENGTH\_SHORT**: Displays the toast for a short duration, usually around 2 seconds.
* **Toast.LENGTH\_LONG**: Displays the toast for a longer duration, usually around 3.5 seconds.
* The show method in Android is used to display a Toast message on the screen. The Toast class provides a simple way to show a quick message to the user. This method is called after creating and configuring the Toast object.

1. **Create an application to develop Login window using UI controls.**

1. Click **New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then click Next.

3. Specify the **Name of your project**, Select the Language as **Java**, and **Select the Minimum SDK as API 24 (“Nougat”, Android 7.0)**. Click Finish Button.

4. Create **background resources** (**bg\_outer.xml**)

a. To create resource file click **app→res→drawable**. Right click **drawable→New→ Drawable Resource File**. The New Resource File dialog box appears.

b. Set **filename as bg\_outer.xml**, root element as **shape** and then click ok. Modify the bg\_outer.xml file.

**5. bg\_outer.xml**

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="12dp" />

<gradient

android:startColor="#02D9F4"

android:endColor="#FBF9F8"

android:angle="100"/>

</shape>

6. Create **background resources** (**bg\_inner.xml**)

a. To create resource file click **app→res→drawable**. Right click **drawable→New→ Drawable Resource File**. The New Resource File dialog box appears.

b. Set **filename as bg\_inner.xml**, root element as **shape** and then click ok. Modify the bg\_inner.xml file.

**7. bg\_inner.xml**

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<gradient

android:startColor="#C9DA5884"

android:endColor="#4905FA"

android:angle="100"/>

<corners android:radius="20dp"/>

</shape>

8. Create **two EditText box** and **a Button** resource in **activity\_main.xml** and update the following code

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:background="@drawable/bg\_outer">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:background="@drawable/bg\_inner"

android:gravity="center"

android:orientation="vertical"

android:padding="30dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:ignore="MissingConstraints">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:fontFamily="sans-serif-condensed-medium"

android:paddingBottom="20dp"

android:text="LOGIN PAGE"

android:textColor="@color/black"

android:textSize="32sp"

android:textStyle="bold"

tools:ignore="MissingConstraints" />

<EditText

android:id="@+id/editTextUsername"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="16dp"

android:hint="Username"

tools:ignore="MissingConstraints" />

<EditText

android:id="@+id/editTextPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/editTextUsername"

android:layout\_marginBottom="16dp"

android:hint="Password"

android:inputType="textPassword"

tools:ignore="MissingConstraints" />

<Button

android:id="@+id/buttonLogin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/editTextPassword"

android:text="Login"

tools:ignore="MissingConstraints" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

9. Create two **EditText** and a **Button** object, create **clickListener**, **onClick** event for button object and update the following code in **MainActivity.java**

package com.example.a3rdprogram;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

private EditText editTextUsername,editTextPassword;

private Button buttonLogin;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

editTextUsername = findViewById(R.id.editTextUsername);

editTextPassword = findViewById(R.id.editTextPassword);

buttonLogin = findViewById(R.id.buttonLogin);

buttonLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String username = editTextUsername.getText().toString().trim();

String password = editTextPassword.getText().toString().trim();

if(username.equals("shivaleela") && password.equals("admin")){

Toast.makeText(MainActivity.this, "Login successful",

Toast.LENGTH\_SHORT).show();

}

else {

Toast.makeText(MainActivity.this, "Invalid username or password",

Toast.LENGTH\_SHORT).show();

}

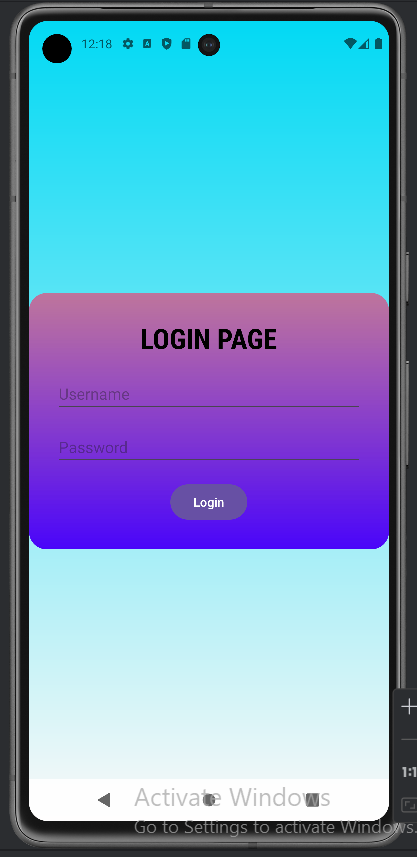
}

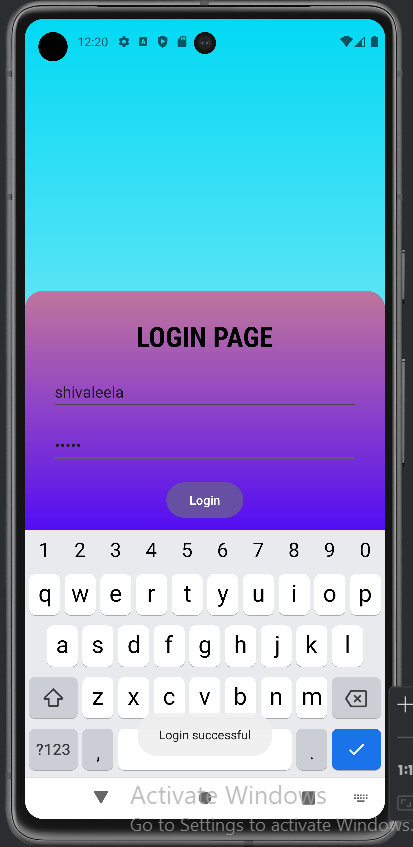
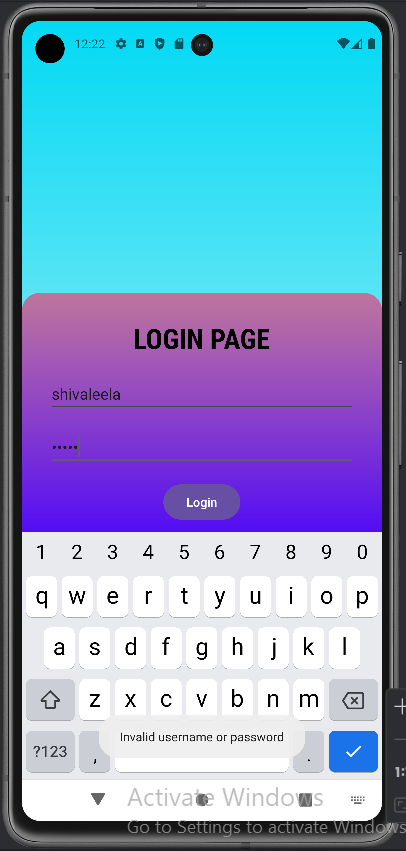
});

}

}

**10. Outputs:**



**Imp methods:**

* **The getText method** in Android is commonly used to retrieve text from a TextView or its subclasses such as EditText or Button. This method returns a CharSequence which can then be used as needed in your code.
* **The toString method** in Android is used to convert various data types and objects to their string representation. In the context of retrieving text from views like TextView or EditText, after using the getText method, you often convert the returned CharSequence to a String using toString.
* **The trim method** in Android, which is a part of the String class, is used to remove any leading and trailing whitespace characters from a string. This method is particularly useful when processing user input to ensure that there are no unnecessary spaces around the text.

1. **Create an application to implement new activity using explicit intent, implicit intent and content providers.**

1. Click **New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then click Next.

3. Specify the **Name of your project**, Select the Language as **Java**, and **Select the Minimum SDK as API 24 (“Nougat”, Android 7.0)**. Click Finish Button.

4. To create another activity for **Explicit Intent,** Click **File****New****Activity** **Empty Views Activity.** A **New Android Activity** dialog box appears, Specify the **Name** of the activity as **NewActivity** then click **Finish**.

5. Create one **Text View** resource in **activity\_new.xml** and update the following code.

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:andr[oid="htt](http://schemas.android.com/apk/res/android)p:/[/sc](http://schemas.android.com/apk/res/android)he[mas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android)

xmlns:app=["http://sc](http://schemas.android.com/apk/res-auto)h[emas.android.com/apk/res-auto"](http://schemas.android.com/apk/res-auto) xmlns:t[ools="ht](http://schemas.android.com/tools)tp:/[/sc](http://schemas.android.com/tools)he[mas.android.com/tools"](http://schemas.android.com/tools) android:id="@+id/main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" tools:context=".NewActivity">

<TextView

android:id="@+id/textView" android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome to Explicit Intent" android:textSize="28sp" app:layout\_constraintBottom\_toBottomOf="parent" app:layout\_constraintEnd\_toEndOf="parent" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintTop\_toTopOf="parent"

/>

</androidx.constraintlayout.widget.ConstraintLayout>

6. Add two events named as **onImplicitButtonClicked, onExplicitButtonClicked** and update the following code in **MainActivity.java**

package com.example.a4thprogram;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

}

public void onImplicitButtonClicked(View view) {

Uri url=Uri.parse("https://www.google.com");

Intent i=new Intent(Intent.ACTION\_VIEW, url);

startActivity(i);

}

public void onExplicitButtonClicked(View view )

{

Intent i=new Intent(MainActivity.this, NewActivity.class);

startActivity(i);

}

}

7. Add **two Button** resource in **activity\_main.xml** and update the following code.

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

tools:context=".MainActivity">

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onImplicitButtonClicked"

android:text="Implicit Intent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.76"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.437"

/>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onExplicitButtonClicked"

android:text="Explicit Intent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.163"

app:layout\_constraintStart\_toStartOf="parent"

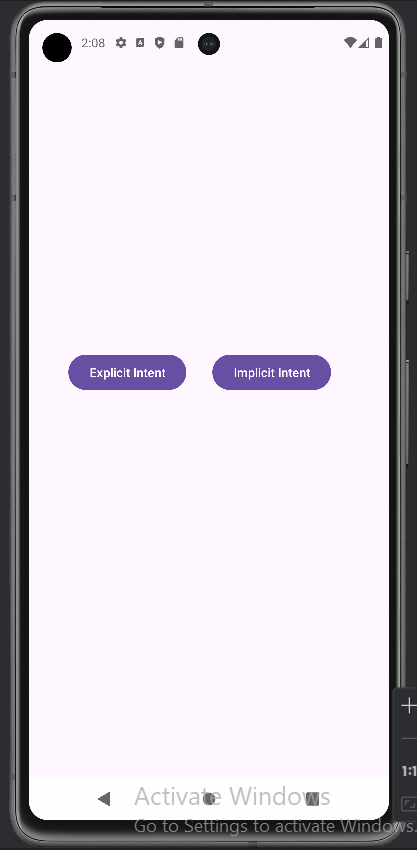
app:layout\_constraintTop\_toTopOf="parent"

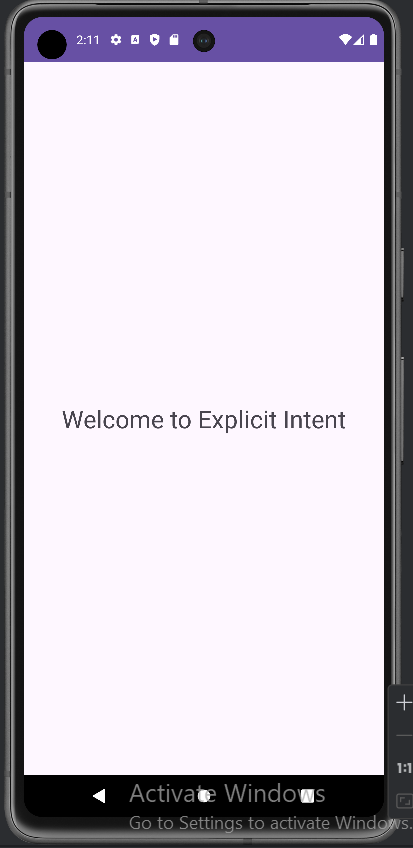
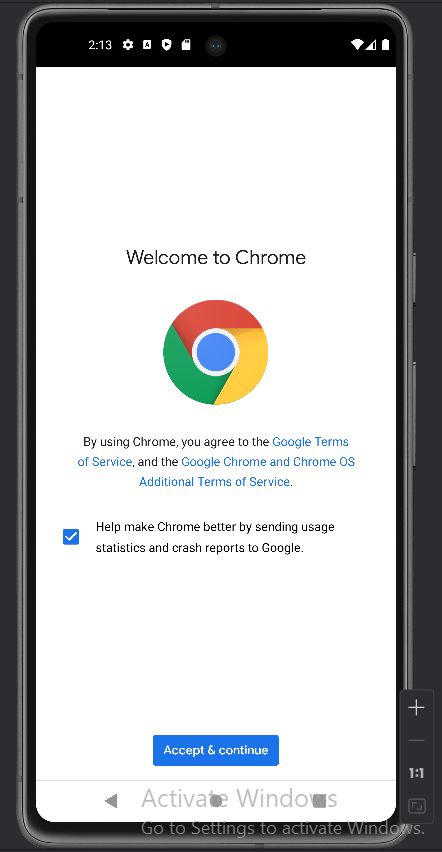
app:layout\_constraintVertical\_bias="0.437"

/>

</androidx.constraintlayout.widget.ConstraintLayout>

**8. Outputs:**

****

** **

**Imp methods:**

* **Implicit intents** do not specify the component to start. Instead, they declare a general action to perform, and it is up to the Android system to find the appropriate component (activity, service, etc.) that can handle the intent. Implicit intents are used when you want to perform an action without needing to know which component will handle it.
* **Explicit intents** specify the exact component (activity, service, etc.) to start by using the component's class name. These are typically used within the same application when you know the exact class you want to start.
* **The Uri.parse method** in Android is used to convert a string representation of a URI (Uniform Resource Identifier) into a Uri object. This method is commonly used when you need to work with URIs, such as when you're opening a web page, sending an email, or making a phone call.

1. **Create an application that displays custom designed opening screen.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button.

4. To create another activity for **Home Page,** Right Click **App****New****Activity** **Empty Views Activity.** A **New Android Activity** dialog box appears, Specify the **Name** of the activity as **mainScreen** then click **Finish**.

5. Create one **TextView** resource in **activity\_mainScreen.xml** and update the following code.

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".mainScreen"

android:gravity="center"

android:background="#09E8D3">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome to home Page"

android:textStyle="bold"

android:textSize="32sp"

android:textColor="@color/black"

tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>

6.  **To add an ImageView resource:** Copy an image and paste it into **drawable folder (**Right- click **Drawable** **Paste** the image**[dora.jpg]**).

7. Set an **image** as **src** in **activity\_main.xml** and update the following code.

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:gravity="center"

>

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/dora"

tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>

**8. MainActivity.java**

package com.example.a5thprogram;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.view.WindowManager;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

private static final int SPLASH\_SCREEN\_TIME\_OUT = 8000;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN,

WindowManager.LayoutParams.FLAG\_FULLSCREEN);

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

Intent i = new Intent(MainActivity.this, mainScreen.class);

startActivity(i);

finish();

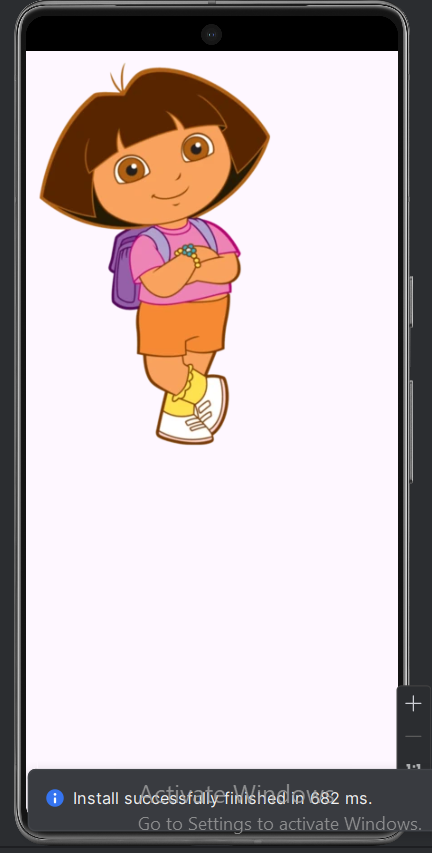
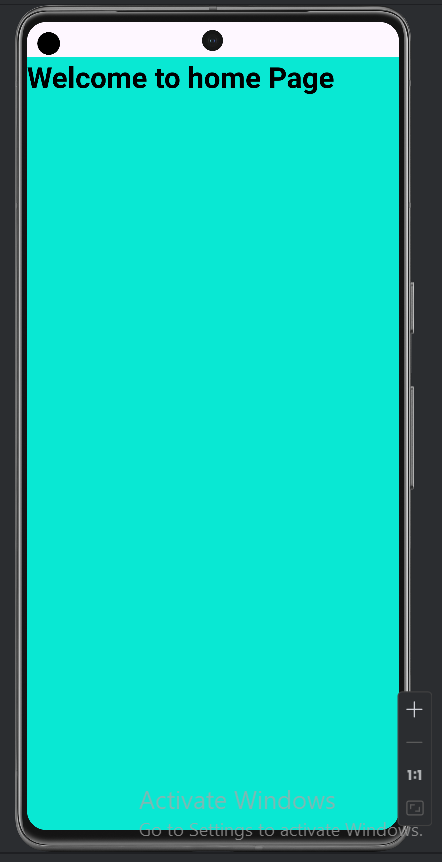
}

}, SPLASH\_SCREEN\_TIME\_OUT);

}

}

**9. Output:**

** **

**Imp methods:**

* **A splash screen** is a common UI element in mobile applications. It's typically shown when the app is launched and serves as an initial screen while the app is loading. In Android, creating a splash screen involves a combination of XML layout design and Java or Kotlin code.
* **The getWindow** method in Android is used to access the current Window of an activity. This method allows you to perform various window-level operations, such as modifying window attributes, hiding the status bar, setting full-screen mode, and more.
* **The setFlags** method in Android is used to set window flags that modify the behavior and appearance of an activity's window. This method is typically called on the Window object, which can be retrieved using the getWindow method. Window flags can control aspects such as whether the window is full-screen, whether the status bar is visible, and whether the window is kept on while the screen is off.
* In Android, **a Handler is a class** that is used to schedule and run tasks (runnables) on the UI thread or other threads. It allows you to schedule tasks to be executed at a later time or to repeatedly execute tasks at a fixed interval. Handlers are often used in conjunction with threads to manage communication between threads or to update the UI from a background thread.
* In Android, **postDelayed is a method** provided by the Handler class that allows you to schedule a Runnable to be executed after a specified delay. This is particularly useful when you want to perform an action after a certain amount of time, such as updating UI elements, executing background tasks, or any other operation that needs to be delayed.
* In Android, **WindowManager.LayoutParams.FLAG\_FULLSCREEN** is a constant flag used within the LayoutParams class of the WindowManager to make an activity or window full-screen. This flag is typically used when you want your activity to occupy the entire screen without showing the status bar (the bar at the top of the screen that displays notifications, battery life, etc.).

1. **Create an UI with all views.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button.

4. Create **background** resources(**bg\_outer.xml**, **bg\_inner.xml, bg.xml**)

* + 1. To create resource file click **app****res****drawable.** Right click **drawable****New** **Drawable Resource File.** The **New Resource File** dialog box appears.
    2. Set **filename** as **bg\_outer.xml**, **root element** as **shape** and then click **ok.**

Modify the bg\_outer.xml file

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<gradient android:startColor="#64EFAE"

android:endColor="#84FFFF"

android:angle="120"

android:gradientRadius="5dp"/>

<corners android:radius="20dp"/>

</shape>

c. Create another background resource for inner layout. Set **filename** as **bg\_inner.xml**, **root element** as **shape** and then click **ok.** Modify the bg\_inner.xml file

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<gradient android:startColor="#64F194"

android:endColor="#B242C5"

android:angle="120"

android:gradientRadius="5dp"/>

<corners android:radius="20dp"

android:topLeftRadius="70dp"

android:bottomRightRadius="70dp"/>

</shape>

d. Likewise, create another background resource for view. Set **filename** as **bg.xml**, **root element** as **shape** and then click **ok.** Modify the bg..xml file

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<solid android:color="#2860F367"/>

<corners android:radius="30dp"/>

<stroke android:color="#00BFA5"

android:width="2dp"/>

</shape>

5. Create a **TextView, EditText, ToggleButton, ImageView, RadioGroup, RadioButton, spinner** and **a Button** resource in **activity\_main.xml** and update the following code.

<?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:id="@+id/main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:gravity="center"

android:orientation="vertical"

android:padding="30dp"

android:background="@drawable/bg\_outer">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="User Information"

android:textColor="#26389C"

android:textSize="30sp"

android:textStyle="bold" />

<ImageView

android:layout\_width="397dp"

android:layout\_height="154dp"

android:src="@drawable/download" />

<ToggleButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOn="Active"

android:textOff="Inactive"/>

<View

android:layout\_width="wrap\_content"

android:layout\_height="20dp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="30dp"

android:paddingBottom="30dp"

android:paddingLeft="5dp"

android:paddingRight="5dp"

android:orientation="vertical"

android:background="@drawable/bg\_inner">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:padding="5dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name"

android:textSize="20sp"

android:textStyle="bold"

android:textColor="#26389C"

android:padding="15dp"/>

<EditText

android:layout\_width="match\_parent"

android:layout\_height="60dp"

android:id="@+id/name"

android:background="@drawable/bg"

android:padding="15dp"/>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:padding="5dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="E-mail"

android:textSize="20sp"

android:textStyle="bold"

android:textColor="#26389C"

android:padding="15dp"/>

<EditText

android:id="@+id/email"

android:layout\_width="match\_parent"

android:layout\_height="60dp"

android:ems="10"

android:inputType="textEmailAddress"

android:background="@drawable/bg"

android:padding="15dp"/>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:padding="5dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Sex"

android:textSize="20sp"

android:textStyle="bold"

android:textColor="#26389C"

android:padding="15dp"

android:paddingEnd="40dp"

/>

<RadioGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/bg"

android:orientation="horizontal"

android:id="@+id/sex">

<RadioButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/male"

android:padding="15dp"

android:text="Male"

android:textColor="#26389C"

android:textSize="20sp"

android:textStyle="bold"/>

<RadioButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/female"

android:padding="15dp"

android:text="Female"

android:textColor="#26389C"

android:textSize="20sp"

android:textStyle="bold"/>

</RadioGroup>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:padding="5dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Country"

android:textSize="20sp"

android:textStyle="bold"

android:textColor="#26389C"

android:padding="15dp"

android:paddingEnd="5dp"/>

<Spinner

android:layout\_width="match\_parent"

android:layout\_height="60dp"

android:id="@+id/country"

android:padding="15dp"

android:background="@drawable/bg"/>

</LinearLayout>

</LinearLayout>

<View

android:layout\_width="match\_parent"

android:layout\_height="40dp"/>

<Button

android:layout\_width="210dp"

android:layout\_height="wrap\_content"

android:id="@+id/submit"

android:background="@drawable/bg"

android:padding="15dp"

android:text="Submit"

android:textColor="#26389C"

android:textSize="20sp"

android:textStyle="bold"/>

</LinearLayout>

6. Create two **EditText** and a **Button** object, create **clickListener**, **onClick** event for button object and update the following code in **MainActivity.java**

package com.example.program6;

import android.app.Dialog;

import android.content.Context;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.Spinner;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

Button sub;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

Button sub = findViewById(R.id.submit);

sub.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showMessage(MainActivity.this, "User Information", "Successfully completed");

}

});

String[] item = new String[]{"India", "Pakisthan", "China", "America", "England"};

ArrayAdapter adapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_item,item);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

Spinner spinner = findViewById(R.id.country);

spinner.setAdapter(adapter);

}

public void showMessage(Context con,String t,String msg)

{

AlertDialog.Builder builder=new AlertDialog.Builder(con);

builder.setTitle(t);

builder.setMessage(msg);

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.dismiss();

}

});

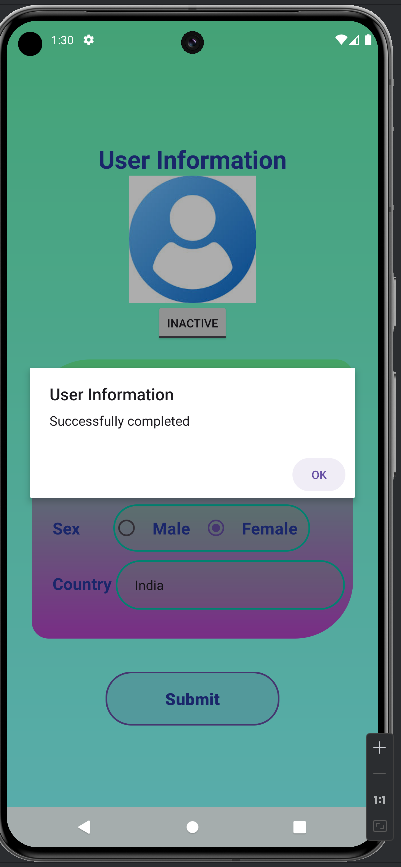
builder.show();

}

}

**7. Outputs:**



**Imp methods:**

* In Android development, **an ArrayAdapter** is a type of adapter that allows you to bind an Array or List of items to a ListView, GridView, Spinner, or similar UI components. It acts as a bridge between the data source (the array or list) and the AdapterView (the UI component).
* In Android, **AlertDialog.Builder** is a class used to create dialog windows with customizable content and behavior. It allows developers to create dialogs that can display messages, take user inputs, show lists, and more.
* In Android, **DialogInterfac**e is an interface that defines a set of methods that allow interaction with a dialog, such as AlertDialog. This interface provides callbacks for handling button clicks and dialog dismissals.
* In Android, when you create an AlertDialog using AlertDialog.Builder, you can use **setPositiveButton()** to define a positive button in the dialog. The positive button is typically used for actions that confirm or proceed with the dialog's purpose.

1. **Create menu in application.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button

* 1. To create another activity for **Home Page,** Right Click **App****New****Activity** **Empty Views Activity.** A **New Android Activity** dialog box appears, Specify the **Name** of the activity as **HomeScreen** then click **Finish**.
  2. **To create a Menu Resource File:**

Right-click on the **res** directory in your Android project, navigate to **New > Android Resource File**, and **name** the file **menus.xml**, Root element as **Menu** and update the following content.

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

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item android:id="@+id/php"

android:title="PHP"/>

<item android:id="@+id/java"

android:title="JAVA"/>

<item android:id="@+id/csharp"

android:title="C#"/>

</menu>

**4. activity\_main.xml:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

</androidx.constraintlayout.widget.ConstraintLayout>

**5. MainActivity.java:**

package com.example.lab7;

import android.os.Bundle;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

}

@Override

public boolean onCreatePanelMenu(int featureId, @NonNull Menu menu) {

MenuInflater inflater=getMenuInflater();

inflater.inflate(R.menu.menu,menu);

return true;

}

@Override

public boolean onOptionsItemSelected(@NonNull MenuItem item) {

if(item.getItemId()==R.id.php) {

Toast.makeText(this, "Php Page", Toast.LENGTH\_SHORT).show();

}

if(item.getItemId()==R.id.java) {

Toast.makeText(this, "Java Page", Toast.LENGTH\_SHORT).show();

}

if(item.getItemId()==R.id.csharp) {

Toast.makeText(this, "C# Page", Toast.LENGTH\_SHORT).show();

}

return super.onOptionsItemSelected(item);

}

}

6. **AndroidManifest.xml:**

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools">

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

<application

android:allowBackup="true"

android:dataExtractionRules="@xml/data\_extraction\_rules"

android:fullBackupContent="@xml/backup\_rules"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/Theme.AppCompat.Light"

tools:targetApi="31">

<activity

android:name=".MainActivity"

android:exported="true">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

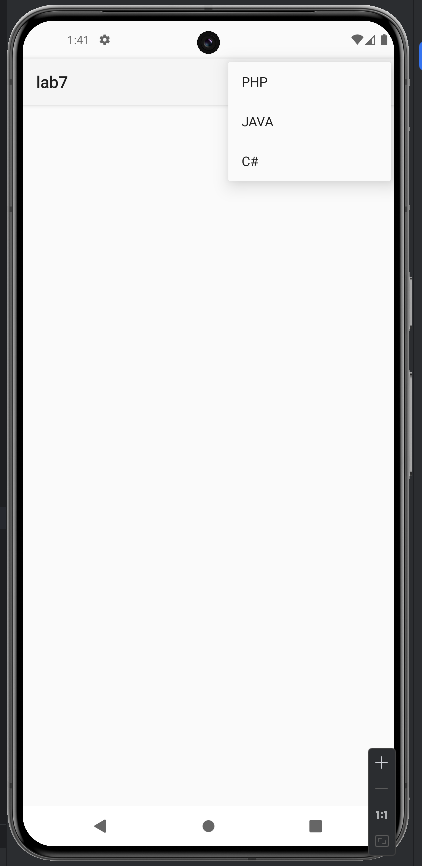
</intent-filter>

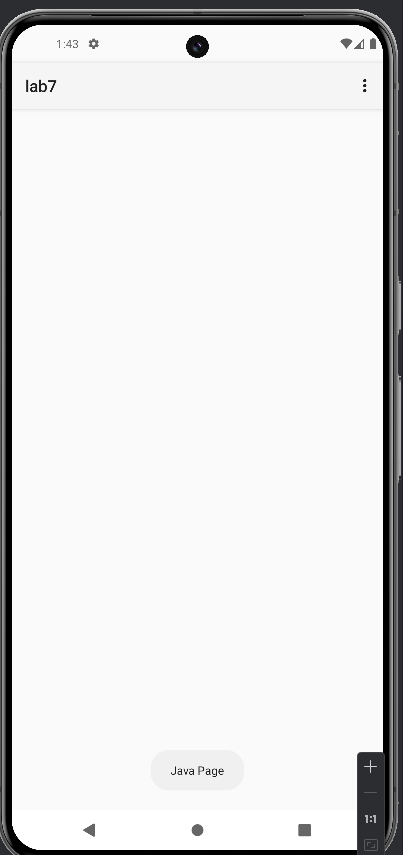
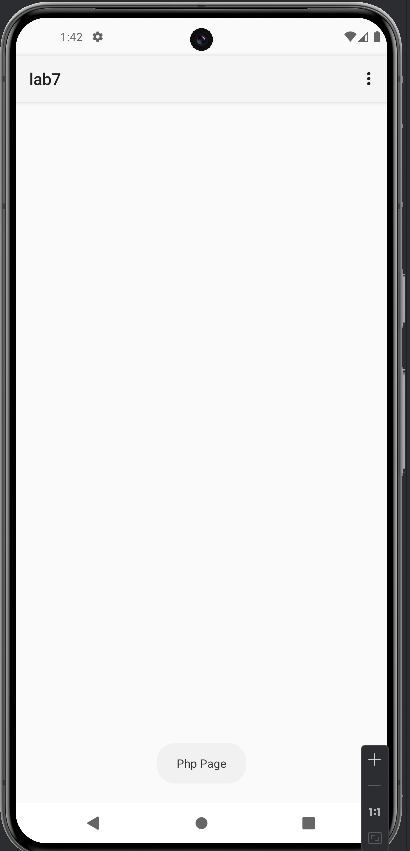
</activity>

</application>

</manifest>

**7. Outputs:**







**Imp methods:**

The **AndroidManifest.xml** file is a crucial component in every Android application. It serves several key purposes:

* **Declaration of Components**: It declares the components of the application, such as activities, services, broadcast receivers, and content providers. Each component that the application uses must be declared in this file.
* **Permissions Declaration**: It specifies the permissions that the application needs in order to access certain features or data on the device. This includes permissions like accessing the internet, reading external storage, etc.
* **Application Metadata**: It contains metadata about the application itself, such as its name, icon, version, and the minimum API level required.
* **Intent Filters**: For activities, services, and broadcast receivers, it specifies intent filters. Intent filters define the types of intents (broadcasts, requests for activities to start, etc.) that each component can respond to.
* **App Components Configuration**: It provides configuration information for each component, such as the activities' layout, the services' properties, etc.
* In Android development, the **onCreatePanelMenu()** method is a callback method that is part of the Activity class. Its primary purpose is to initialize the options menu (also known as the action bar menu) when it is first displayed.
* The **onCreatePanelMenu()** method is responsible for creating the options menu (action bar menu) in an Android activity. It is called when the system needs to initialize the menu for the first time or when the user requests to open the menu.
* **MenuInflater** is a class in Android that is used to instantiate menu objects from XML files. It takes a menu XML file and creates the corresponding Menu object (android.view.Menu). This Menu object can then be displayed in various ways, such as in the options menu (action bar menu) of an Activity or as a context menu.
* **getMenuInflater()** is a method available in the Activity and Fragment classes that returns a MenuInflater object. This MenuInflater object is used to inflate menu XML resources into Menu objects
* In Android development, the **onOptionsItemSelected(MenuItem item)** method is a callback method that is invoked when the user selects an item from the options menu (or action bar menu) of an Activity or Fragment. The **onOptionsItemSelected(MenuItem item**) method is called when the user selects a menu item from the options menu. It allows you to perform actions or respond to user interaction based on which menu item was clicked
* In Android development, **item.getItemId()** is a method provided by the MenuItem class. It is used to retrieve the unique identifier assigned to a menu item defined in the menu XML resource file.
* **The getItemId()** method is used to distinguish between different menu items when handling user interaction with menus, such as the options menu (action bar menu) or context menus.

1. **Read/write the local data.**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button

4. To create another activity for **Home Page,** Right Click **App****New****Activity** **Empty Views Activity.** A **New Android Activity** dialog box appears, Specify the **Name** of the activity as **Mainactivity2** then click **Finish**.

5. Create one **TextView** resource in **activity\_main2.xml** and update the following code.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity2"

android:orientation="vertical">

<Button

android:id="@+id/btnFetch"

android:layout\_width="168dp"

android:layout\_height="wrap\_content"

android:text="Fetch" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="User Name" >

</TextView>

<EditText

android:id="@+id/etUserName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

</EditText>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Password">

</TextView>

<EditText

android:id="@+id/etPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

</EditText>

</LinearLayout>

**4. MainActivity2.java:**

package com.example.program8;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

public class MainActivity2 extends AppCompatActivity {

Button btnFetch;

EditText etUserName,etPassword;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main2);

btnFetch = (Button) findViewById(R.id.btnFetch);

etUserName = (EditText)findViewById(R.id.etUserName);

etPassword = (EditText)findViewById(R.id.etPassword);

btnFetch.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View view)

{

SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE);

String username = sharedPreferences.getString("username", "");

String password = sharedPreferences.getString("password", "");

etUserName.setText(username);

etPassword.setText(password);

}

});

}}

**6. Activity\_main.xml:**

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="User Name"/>

<EditText

android:id="@+id/etUserName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

</EditText>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Password">

</TextView>

<EditText

android:id="@+id/etPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

</EditText>

<Button

android:id="@+id/btnsave"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Save" />

<Button

android:id="@+id/btnnext"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Next" />

</LinearLayout>

**7. MainActivity.java:**

package com.example.program8;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

Button btnsave, btnnext;

EditText etUserName, etPassword;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

btnsave = (Button) findViewById(R.id.btnsave);

btnnext = (Button) findViewById(R.id.btnnext);

etUserName = (EditText) findViewById(R.id.etUserName);

etPassword = (EditText) findViewById(R.id.etPassword);

btnsave.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putString("username", etUserName.getText().toString());

editor.putString("password", etPassword.getText().toString());

editor.apply();

Toast.makeText(getApplicationContext(), "Saved successfully", Toast.LENGTH\_LONG).show();

}

});

btnnext.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(getApplicationContext(), MainActivity2.class);

startActivity(intent);

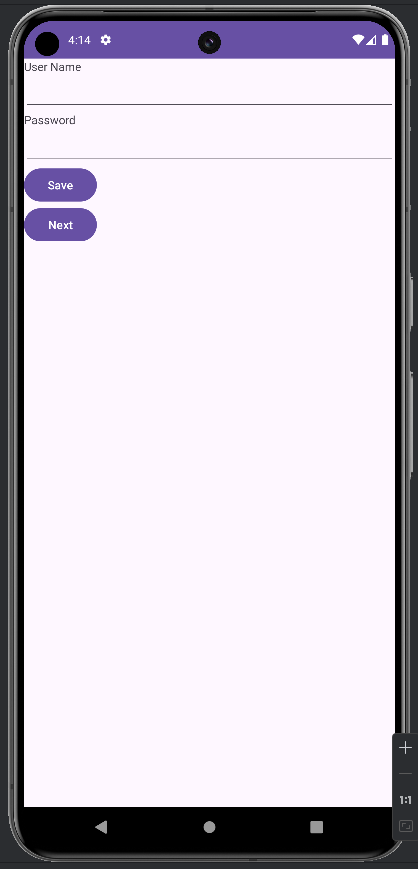
}

});

}

}

**8. Output:**





**Imp methods:**

* In Android development, the line SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE); is used to obtain a reference to a SharedPreferences object associated with a specific file name ("MyPrefs" in this case) and a privacy mode (Context.MODE\_PRIVATE)
* **SharedPreferences** is an interface provided by Android for storing and retrieving simple data in key-value pairs. It allows you to save primitive data types (like boolean, int, String, etc.) persistently across various sessions of an application lifecycle.
* **getSharedPreferences()** is a method available in the Context class (which activities and services extend) that retrieves an instance of SharedPreferences based on a specific file name and privacy mode.

**File Name ("MyPrefs")**:

* This parameter specifies the name of the file where the shared preferences will be stored. If the file does not exist, Android will create it.
* Each shared preferences file is private to the application by default, meaning other applications cannot access this file directly.

**Privacy Mode (Context.MODE\_PRIVATE)**:

* This parameter specifies the access mode for the shared preferences file. In this case, MODE\_PRIVATE indicates that the file is accessible only to the calling application.
* Other available modes include MODE\_WORLD\_READABLE and MODE\_WORLD\_WRITEABLE, but these are deprecated due to security concerns.
* **Obtain a SharedPreferences instance**: Use the getSharedPreferences() method to get an instance of the SharedPreferences object. This method takes two parameters: the name of the SharedPreferences file and the mode in which the file should be opened.
* **Get an Editor instance**: Use the edit() method on the SharedPreferences object to get a SharedPreferences.Editor instance. This editor is used to make changes to the SharedPreferences data.
* **Add data to SharedPreferences**: Use the appropriate put methods (e.g., putString(), putInt(), putBoolean(), etc.) on the Editor to add data.
* **Apply or commit the changes**: Use apply() or commit() to save the changes. apply() saves the changes asynchronously, while commit() saves them synchronously.
* **Retrieve data from SharedPreferences**: Use the appropriate get methods (e.g., getString(), getInt(), getBoolean(), etc.) on the SharedPreferences object to retrieve data.
* In Android, **the editor.putString** method is used to save a String value in the SharedPreferences. This is part of the process for storing simple key-value pairs in the device's persistent storage.
* In Android, **getApplicationContext()** is a method that returns the context of the entire application. It is an instance of Context that is tied to the lifecycle of the application, rather than the lifecycle of any specific activity or component. This is particularly useful when you need a context that will live as long as your application is running,

1. **Create/ Read/ write data with database (SQLite).**

1. Click Start →**Android Studio**, a Welcome to Android Studio dialog box will appear. **Click New Project**, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then clicks next.

3. Specify the **Name of your project**, Select the **Language as Java**, and Select the **Minimum SDK as API 24 (“Nougat”, Android 7.0).** Click Finish Button

4. Update the following code in activity\_main.xml, activity\_view.xml, MainActivity.java, ViewActvity.java

5. Create a class file **right click app- new- java class name it as student** and update the following code in student.java

6. To create another activity right click on app – new **activity-Empty views Activity**.

7. Update the following code in **activity\_edit.xml** and **EditActivity.java**. 8. Click Run App or Shift+F10 to execute the application.

8**. activity\_main.xml:**

<?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">  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Course Registation"  
 android:textSize="30dp"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Name"  
 />

<EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:id="@+id/name"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Course"  
 />  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:id="@+id/course"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Fee"  
 />  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:id="@+id/fee"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:id="@+id/bt1"  
 android:text="Ok"  
 />  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:id="@+id/bt2"  
 android:text="View"  
 />  
 </LinearLayout>  
</LinearLayout>

**9. MainActivity.java:**

package com.example.lab\_9;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.Intent;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteStatement;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
public class MainActivity extends AppCompatActivity {  
 EditText ed1,ed2,ed3;  
 Button b1,b2;  
 @SuppressLint("MissingInflatedId")  
 @Override  
 protected void onCreate(Bundle savedInstanceState)  
 {super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 ed1 = findViewById(R.id.*name*);  
 ed2 = findViewById(R.id.*course*);  
 ed3 = findViewById(R.id.*fee*);  
 b1 = findViewById(R.id.*bt1*);  
 b2 = findViewById(R.id.*bt2*);  
 b2.setOnClickListener(new View.OnClickListener()  
 {@Override  
 public void onClick(View v)  
 {  
 Intent i = new Intent(getApplicationContext(),activity\_view.class);  
 startActivity(i);  
 }  
 });  
 b1.setOnClickListener(new View.OnClickListener() {@Override  
 public void onClick(View v) {  
 insert();  
 }  
 });  
 }  
 public void insert()  
 {  
 try  
 {  
 String name = ed1.getText().toString();  
 String course = ed2.getText().toString();  
 String fee = ed3.getText().toString();  
 SQLiteDatabase db = openOrCreateDatabase("SliteDb",  
 Context.*MODE\_PRIVATE*, null);  
 db.execSQL("CREATE TABLE IF NOT EXISTS records(id INTEGER PRIMARY KEY AUTOINCREMENT,name VARCHAR,course VARCHAR,fee VARCHAR)");  
 String sql = "insert into records(name,course,fee)values('" +name + "','" +  
 course + "','" + fee + "')";  
 SQLiteStatement statement = db.compileStatement(sql);  
 statement.execute();  
 Toast.*makeText*(this,"Record addded",Toast.*LENGTH\_LONG*).show();  
 ed1.setText("");  
 ed2.setText("");  
 ed3.setText("");  
 ed1.requestFocus();  
 }  
 catch (Exception ex)  
 {  
 Toast.*makeText*(this,"RecordFail",Toast.*LENGTH\_LONG*).show();  
 }  
 }  
}

**10. activity\_view.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".activity\_view"  
 android:orientation="vertical">  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/lst1"  
 />  
</LinearLayout>

**11. ViewActivity.java:**

package com.example.lab\_9;  
  
 import androidx.appcompat.app.AppCompatActivity;  
 import android.content.Context;  
 import android.content.Intent;  
 import android.database.Cursor;  
 import android.database.sqlite.SQLiteDatabase;  
 import android.os.Bundle;  
 import android.view.View;  
 import android.widget.AdapterView;  
 import android.widget.ArrayAdapter;  
 import android.widget.ListView;  
 import java.util.ArrayList;  
  
public class activity\_view extends AppCompatActivity {  
 ListView lst1;  
 ArrayList<String> titles = new ArrayList<String>();  
 ArrayAdapter arrayAdapter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_view*);  
 SQLiteDatabase db = openOrCreateDatabase("SliteDb",Context.*MODE\_PRIVATE*,null);  
 lst1 = findViewById(R.id.*lst1*);  
 final Cursor c = db.rawQuery("select \* from records",null);  
 int id = c.getColumnIndex("id");  
 int name = c.getColumnIndex("name");  
 int course = c.getColumnIndex("course");  
 int fee = c.getColumnIndex("fee");  
 titles.clear();  
 arrayAdapter = new ArrayAdapter(this,  
 androidx.appcompat.R.layout.*support\_simple\_spinner\_dropdown\_item*,titles);  
 lst1.setAdapter(arrayAdapter);  
 final ArrayList<Student> stud = new ArrayList<Student>();  
 if(c.moveToFirst())  
 {  
 do {  
 Student stu = new Student();  
 stu.id = c.getString(id);  
 stu.name = c.getString(name);  
 stu.course = c.getString(course)  
 ;stu.fee = c.getString(fee);  
 stud.add(stu);  
 titles.add(c.getString(id) + " \t " + c.getString(name) + "\t " + c.getString(course) + " \t " + c.getString(fee) );  
 } while(c.moveToNext());  
 arrayAdapter.notifyDataSetChanged();  
 lst1.invalidateViews();  
 }  
 lst1.setOnItemClickListener(new AdapterView.OnItemClickListener()  
 {@Override  
 public void onItemClick(AdapterView parent, View view, int position, long id) {  
 String aa = titles.get(position).toString();  
 Student stu = stud.get(position);  
 Intent i = new  
 Intent(getApplicationContext(),EditActivity.class);  
 i.putExtra("id",stu.id);  
 i.putExtra("name",stu.name);  
 i.putExtra("course",stu.course);  
 i.putExtra("fee",stu.fee);  
 startActivity(i);  
 }  
 });  
 }  
}

**12. Student.java:**

package com.example.lab\_9;  
  
public class Student {  
 String id;  
 String name;  
 String course;  
 String fee;  
 String titles;  
}

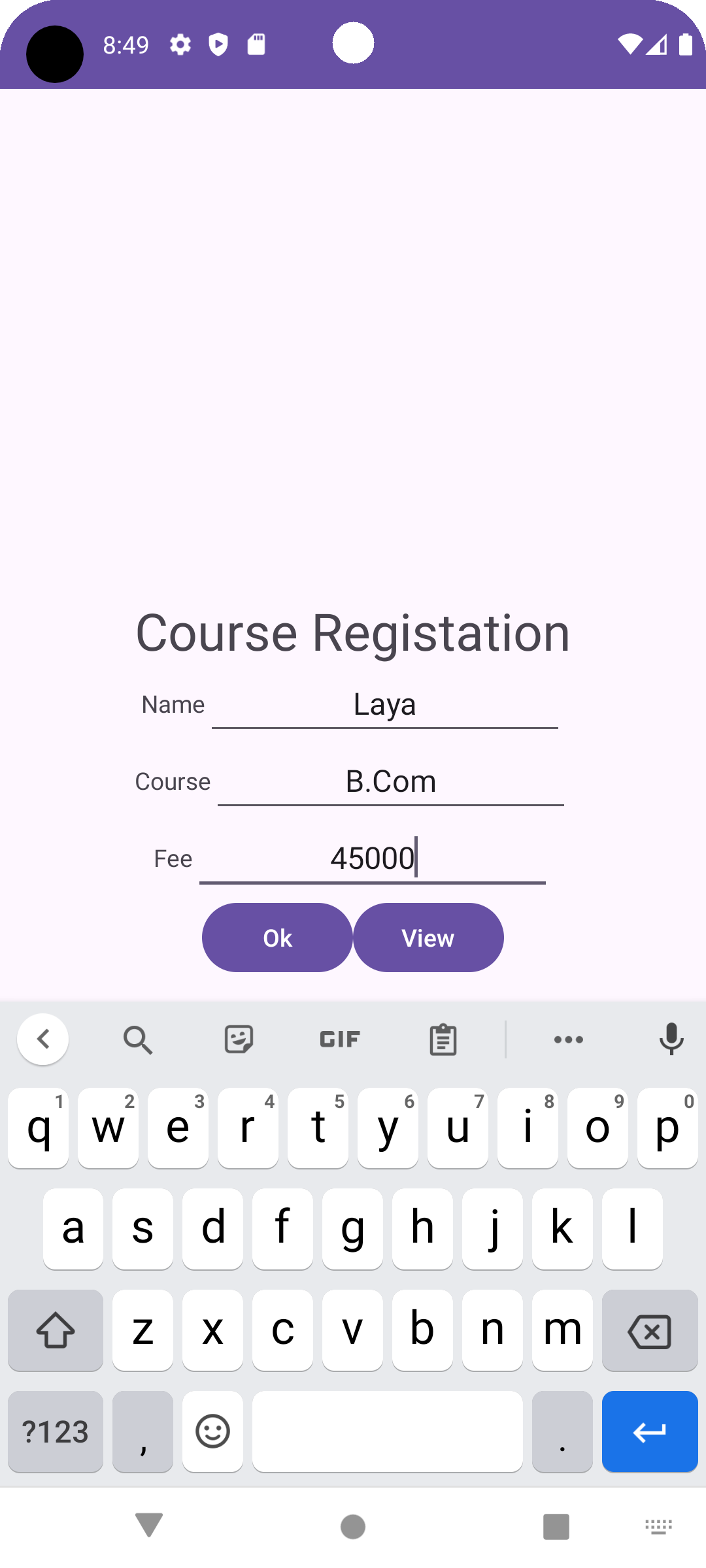
13. **EditActivity.java:**

package com.example.lab\_9;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.content.Context;  
import android.content.Intent;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteStatement;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import  
 android.widget.EditText;  
import android.widget.Toast;  
public class EditActivity extends AppCompatActivity {  
 EditText ed1,ed2,ed3,ed4;  
 Button b1,b2,b3;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_edit*);  
 ed1 = findViewById(R.id.*name*);  
 ed2 = findViewById(R.id.*course*);  
 ed3 = findViewById(R.id.*fee*);  
 ed4 = findViewById(R.id.*id*);  
 b1 = findViewById(R.id.*bt1*);  
 b2 = findViewById(R.id.*bt2*);  
 b3 = findViewById(R.id.*bt3*);  
 Intent i = getIntent();  
 String t1 = i.getStringExtra("id").toString();  
 String t2 = i.getStringExtra("name").toString();  
 String t3 = i.getStringExtra("course").toString();  
 String t4 = i.getStringExtra("fee").toString();  
 ed4.setText(t1);  
 ed1.setText(t2);  
 ed2.setText(t3);  
 ed3.setText(t4);  
 b2.setOnClickListener(new View.OnClickListener()  
 {@Override  
 public void onClick(View v) {  
 try  
 {  
 String id = ed4.getText().toString();SQLiteDatabase db =  
 openOrCreateDatabase("SliteDb",Context.*MODE\_PRIVATE*,null);  
 String sql = "delete from records where id = " + id + "";  
 SQLiteStatement statement = db.compileStatement(sql);  
 statement.execute();  
 Toast.*makeText*(EditActivity.this,"RecordDeleted",Toast.*LENGTH\_LONG*).show();  
 ed1.setText("");  
 ed2.setText("");  
 ed3.setText("");  
 ed1.requestFocus();  
 }  
 catch (Exception ex)  
 {  
 Toast.*makeText*(EditActivity.this,"Record Fail",Toast.*LENGTH\_LONG*).show();  
 }  
 }  
 });  
 b3.setOnClickListener(new View.OnClickListener() {@Override  
 public void onClick(View v) {  
 Intent i = new Intent(getApplicationContext(),activity\_view.class);  
 startActivity(i);  
 }  
 });  
 b1.setOnClickListener(new View.OnClickListener() {@Override  
 public void onClick(View v) {  
 try {  
 String name = ed1.getText().toString();  
 String course = ed2.getText().toString();  
 String fee = ed3.getText().toString();  
 String id = ed4.getText().toString();  
 SQLiteDatabase db = openOrCreateDatabase("SliteDb",Context.*MODE\_PRIVATE*, null);  
 String sql = "update records set name = '" + name + "',course='" +course +  
 "',fee='" + fee + "' where id= " + id + "";  
 SQLiteStatement statement = db.compileStatement(sql);  
 statement.execute();  
 Toast.*makeText*(EditActivity.this, "Record Updated",  
 Toast.*LENGTH\_LONG*).show();  
 ed1.setText("");  
 ed2.setText("");  
 ed3.setText("");  
 ed1.requestFocus();  
 } catch (Exception ex) {  
 Toast.*makeText*(EditActivity.this, "Record Fail",  
 Toast.*LENGTH\_LONG*).show();  
 }  
 }  
 });  
 } }

**14. activity\_edit.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent" android:layout\_height="match\_parent"  
 tools:context=".EditActivity" android:orientation="vertical">  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Course Registation"  
 android:textSize="30dp"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal" android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="ID"  
 />  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:ems="10"  
 android:id="@+id/id"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal" android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Name"  
 />  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:ems="10"  
 android:id="@+id/name"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal" android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Course"/>  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:ems="10"  
 android:id="@+id/course"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Fee"  
 />  
 <EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:id="@+id/fee"  
 android:textAlignment="center"  
 />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal" android:gravity="center">  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:id="@+id/bt1"  
 android:text="Edit"  
  
 />  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:id="@+id/bt2"  
 android:text="Delete"  
 />  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1" android:id="@+id/bt3"  
 android:text="Back"  
 />  
 </LinearLayout>  
</LinearLayout>

**Outputs:**

**Imp methods:**

* In Android, **the requestFocus()** method is used to request focus for a specific view. When a view requests focus, it becomes the target for user input events, such as keyboard input. This method is commonly used in situations where you want a specific view (like an EditText) to be ready for user input immediately when the activity starts or when a certain event occurs.
* In Android, the **notifyDataSetChanged()** method is used in conjunction with an ArrayAdapter (or any adapter that extends BaseAdapter) to inform the adapter that the underlying data has changed and the view should be refreshed to reflect this change. This method is particularly useful when you modify the data in the adapter (such as adding or removing items) and want the associated view (like a ListView or Spinner) to update accordingly
* In Android**, invalidateViews()** is a method used to refresh the entire list view by invalidating all the views in the list. This is useful when you need to update the display of a ListView but do not want to change the underlying data. It essentially forces the ListView to redraw all its child views.

1. **Create an application to send SMS and receive SMS**

1. Click **Start- Android Studio**, a **Welcome to Android Studio dialog box will appear**. Click New Project, the **New Project Dialog box appears.**

2. Choose **Empty Views Activity** then click Next.

3. Specify the Name of your project, Select the **Language as Java**, and Select the **SDK as API 24(“Nougat”,Android 7.0)**.Click Finish Button.

4. Update the following code in **activity\_main.xml** and **MainActivity.java**

5. Click Run app or shift+F10 to execute the application.

6. **activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <EditText  
 android:id="@+id/editTextPhoneNumber"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter phone number"  
 android:layout\_margin="16dp"/>  
 <EditText  
 android:id="@+id/editTextMessage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter message"  
 android:layout\_below="@id/editTextPhoneNumber"  
 android:layout\_margin="16dp"/>  
 <Button  
 android:id="@+id/buttonSend"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Send"  
 android:layout\_below="@id/editTextMessage"  
 android:layout\_alignParentEnd="true"  
 android:layout\_marginEnd="16dp"  
 android:onClick="sendMessage"  
 tools:ignore="UsingOnClickInXml" />  
 <TextView  
 android:id="@+id/textViewReceivedMessages"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/buttonSend"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="16dp"  
 android:textColor="@color/black" />  
</RelativeLayout>

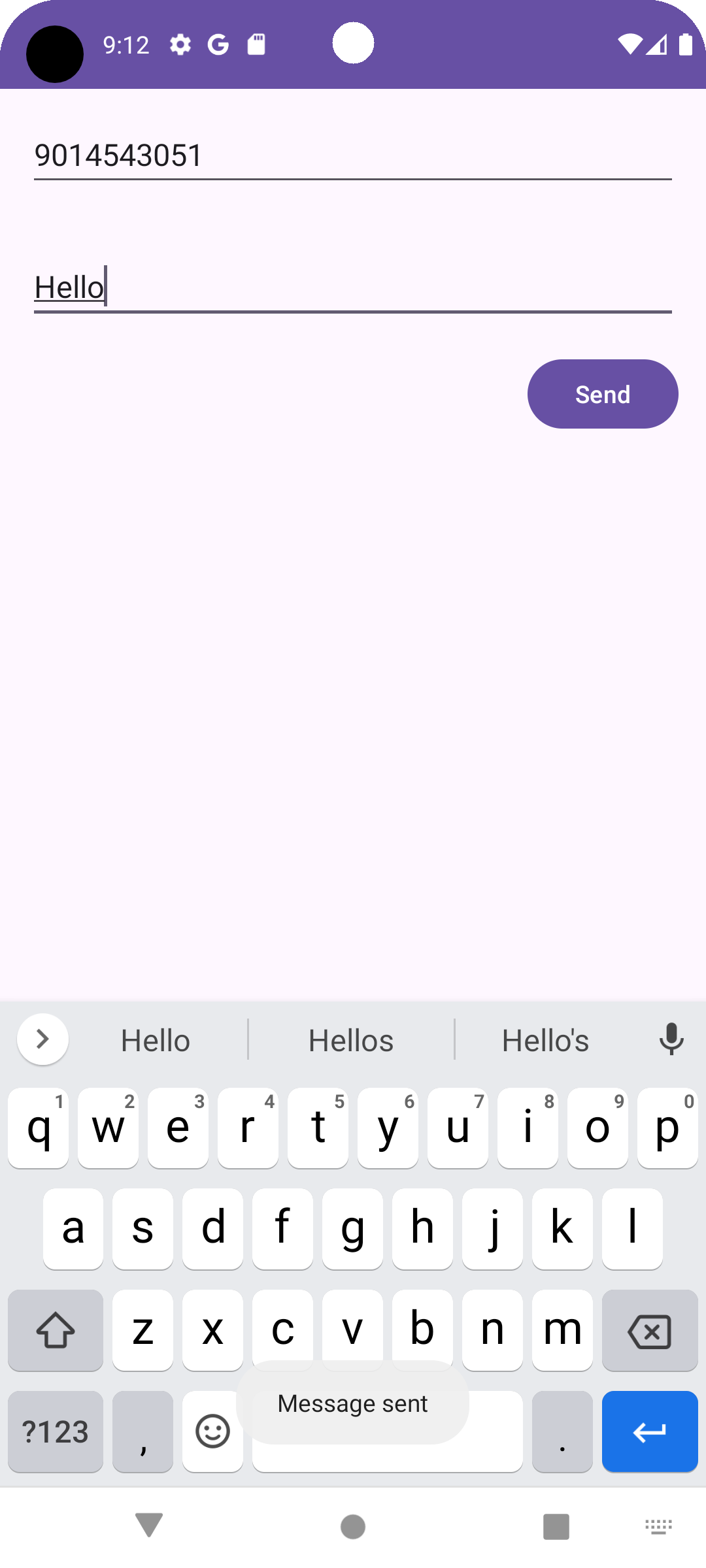
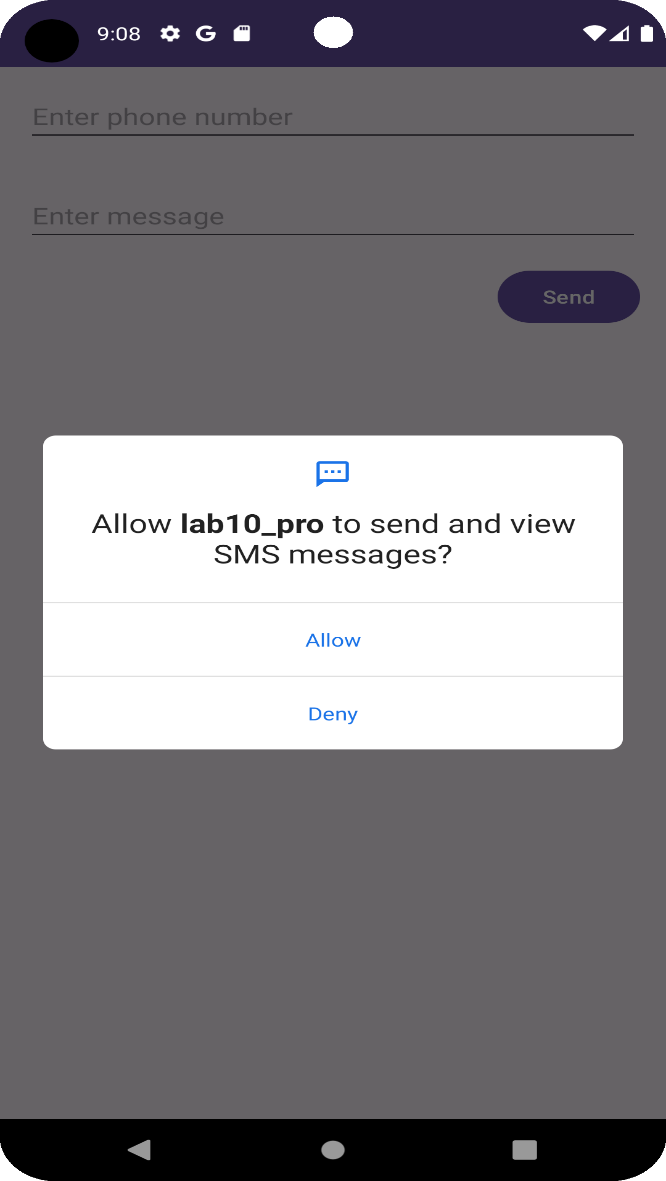
**7. Mainactivity.java:**

package com.example.lab10\_pro;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.telephony.SmsManager;  
import android.telephony.SmsMessage;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import android.widget.Toast;  
import android.Manifest;  
public class MainActivity extends AppCompatActivity {  
 private static final int *SMS\_PERMISSION\_CODE* = 101;  
 private EditText editTextPhoneNumber;  
 private EditText editTextMessage;  
 private TextView textViewReceivedMessages;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 editTextPhoneNumber = findViewById(R.id.*editTextPhoneNumber*);  
 editTextMessage = findViewById(R.id.*editTextMessage*);  
 textViewReceivedMessages =  
 findViewById(R.id.*textViewReceivedMessages*);  
 // Request SMS permissions if not granted  
  
 if (!checkSMSPermission()) {  
 requestSMSPermission();  
 }  
 // Register SMS receiver  
 IntentFilter intentFilter = new IntentFilter();  
 intentFilter.addAction("android.provider.Telephony.SMS\_RECEIVED");  
 registerReceiver(smsReceiver, intentFilter);  
 }  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 unregisterReceiver(smsReceiver);  
 }  
 // Button click listener for sending SMS  
 public void sendMessage(View view) {  
 String phoneNumber =  
 editTextPhoneNumber.getText().toString().trim();  
 String message = editTextMessage.getText().toString();  
  
 if (phoneNumber.isEmpty()) {  
 Toast.*makeText*(this, "Please enter a valid phone number",  
 Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 try {  
 SmsManager smsManager = SmsManager.*getDefault*();  
 smsManager.sendTextMessage(phoneNumber, null, message, null, null);  
 Toast.*makeText*(this, "Message sent", Toast.*LENGTH\_SHORT*).show();  
 } catch (IllegalArgumentException e) {  
 Toast.*makeText*(this, "Invalid phone number format",  
 Toast.*LENGTH\_SHORT*).show();  
 } catch (Exception e) {  
 Toast.*makeText*(this, "Failed to send message",  
 Toast.*LENGTH\_SHORT*).show();  
 e.printStackTrace();  
 }  
 }  
 // Check if SMS permission is granted  
  
 private boolean checkSMSPermission() {  
 return ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*SEND\_SMS*) == PackageManager.*PERMISSION\_GRANTED*;  
 }  
 // Request SMS permission  
 private void requestSMSPermission() {  
 ActivityCompat.*requestPermissions*(this, new  
 String[]{Manifest.permission.*SEND\_SMS*}, *SMS\_PERMISSION\_CODE*);  
 }  
 // SMS receiver  
 private final BroadcastReceiver smsReceiver = new BroadcastReceiver() {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 Bundle bundle = intent.getExtras();  
 if (bundle != null) {  
 Object[] pdus = (Object[]) bundle.get("pdus");  
 if (pdus != null) {  
 for (Object pdu : pdus) {  
 SmsMessage smsMessage = SmsMessage.*createFromPdu*((byte[]) pdu);  
  
 String senderPhoneNumber = smsMessage.getDisplayOriginatingAddress();  
 String messageBody = smsMessage.getMessageBody();  
 textViewReceivedMessages.append("From: " +  
 senderPhoneNumber + "\n");  
 textViewReceivedMessages.append("Message: " +  
 messageBody + "\n\n");  
 }  
 }  
 }  
 }  
 };  
}

**8. AndroidManifest.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <uses-feature  
 android:name="android.hardware.telephony"  
 android:required="false"/>  
  
 <uses-permission android:name="android.permission.SEND\_SMS"/>  
 <uses-permission android:name="android.permission.RECEIVE\_SMS"/>  
 <uses-permission android:name="android.permission.READ\_SMS"/>  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Lab10\_pro"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**9. Outputs:**



**Imp methods:**

* In Android, when requesting runtime permissions, such as for sending or reading SMS, you need to handle the permission request and its result. The SMS\_PERMISSION\_CODE is a request code used to identify the permission request in the onRequestPermissionsResult callback. The specific value of 101 is arbitrary and can be any integer that uniquely identifies the permission request within the application.
* The request code is used to distinguish between different permission requests, especially when your application might request multiple permissions at different times. When the user responds to the permission request, the system calls onRequestPermissionsResult with the same request code that you used in the permission request, allowing you to identify which permission was responded to
* **Check SMS Permission**: Define a method to check if the SMS permission is already granted.
* **Request SMS Permission**: Define a method to request the SMS permission if it's not already granted.
* In Android, an **IntentFilter** is used to specify the types of intents that a component, such as an Activity, Service, or BroadcastReceiver, can respond to. An IntentFilter declares the capabilities of its parent component — what types of intents the component is interested in receiving. This is useful for defining both implicit intent receivers and broadcast receivers
* In Android, the **onDestroy()** method is a lifecycle callback that is called when an activity is being destroyed. This can happen due to various reasons such as the activity being finished, the system needing to free up resources, or the user explicitly closing the activity. It is an appropriate place to clean up resources that the activity has been using, such as stopping services, unregistering broadcast receivers, closing database connections, etc.
* In Android, if you dynamically registered a BroadcastReceiver using **registerReceiver()** (typically done in **onResume()** or a similar lifecycle method), you should unregister it to prevent memory leaks and unnecessary processing when your component (such as an Activity) is no longer active. This is crucial to avoid issues where the receiver continues to receive broadcasts even when the activity is not in the foreground, which can lead to unexpected behavior or crashes.
* **printStackTrace()** is a method that belongs to the Throwable class. It is used to print the stack trace of an exception to the console or standard error stream. The stack trace provides information about the sequence of method calls that led up to the exception being thrown. This is particularly useful for debugging and understanding the flow of your program when an unexpected exception occurs.
* In Android development, particularly when dealing with **SMS messaging, pdus refers** to a parameter used to handle SMS messages in a raw byte format. Here's an explanation of how pdus is typically used and what the condition pdus != null signifies:
* When an Android device receives an SMS, the system processes it into a format known as a **Protocol Data Unit (PDU). Multiple PDUs** can represent parts of a single concatenated SMS message (for long messages split into segments).
* In Android development, broadcasting and receiving broadcasts are fundamental mechanisms for communication between different components within your application or between applications. Let's delve into the key concepts and methods involved: sendBroadcast(), onReceive(), and BroadcastReceiver.

### **1. sendBroadcast(Intent)**

* The sendBroadcast(Intent) method is used to send a broadcast Intent to the Android system. This broadcast can be received by other components within your application or even by components in other applications that have registered to receive that particular broadcast.

 **BroadcastReceiver**: A subclass of BroadcastReceiver is created to handle specific broadcast events.

 **onReceive()**: Override this method to define the behavior when the broadcast is received.

 **Context**: Provides the context in which the broadcast is received.

*  **Intent**: Contains the data associated with the broadcast, including any extras added when the broadcast was sent.

### **2.onReceive(Context, Intent)**

The onReceive(Context, Intent) method is part of the BroadcastReceiver class. It is called by the Android system when a broadcast Intent matching the IntentFilter (registered action) defined in the BroadcastReceiver is received.

### **3. BroadcastReceiver**

A BroadcastReceiver is a component that enables your application to listen for system-wide broadcast announcements. It can respond to broadcasts from other applications or from the system itself.

1. **Create an application to send an Email.**

1. **Click Start- Android Studio**, a Welcome to **Android Studio dialog box will appear**. Click New Project, the **New Project Dialog box appears**.

2. Choose **Empty Views Activity** then click Next.

3. Specify the Name of your project, Select the **Language as Java**, and Select the **SDK as API 24(“Nougat”,Android 7.0)**.Click Finish **Button**.

4. Update the following code in **activity\_main.xml** and **MainActivity.java**

5. Click Run app or shift+F10 to execute the application.

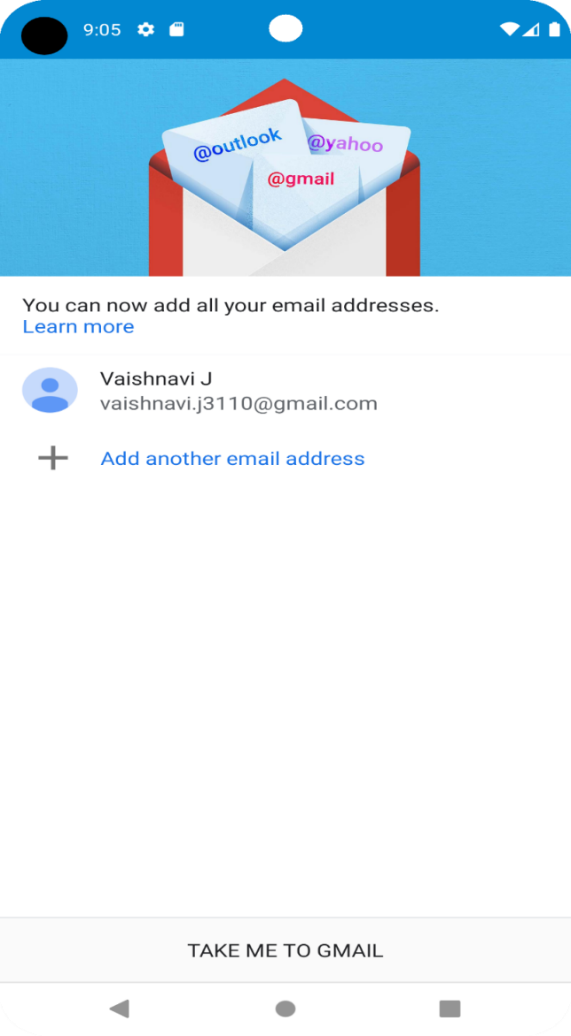
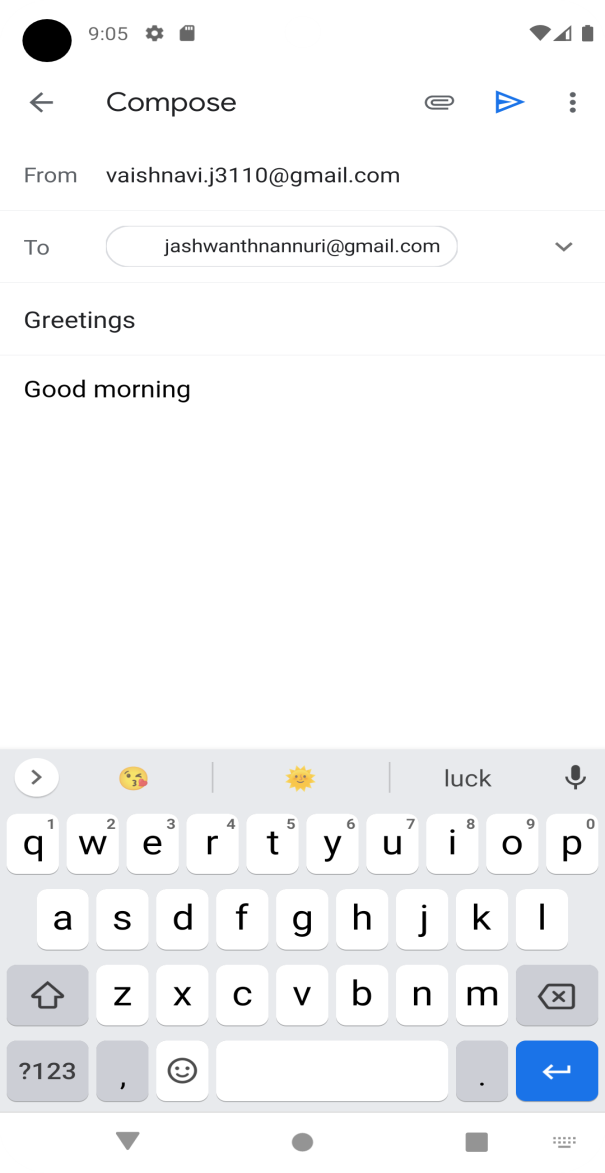
6. **activity\_main.xml**

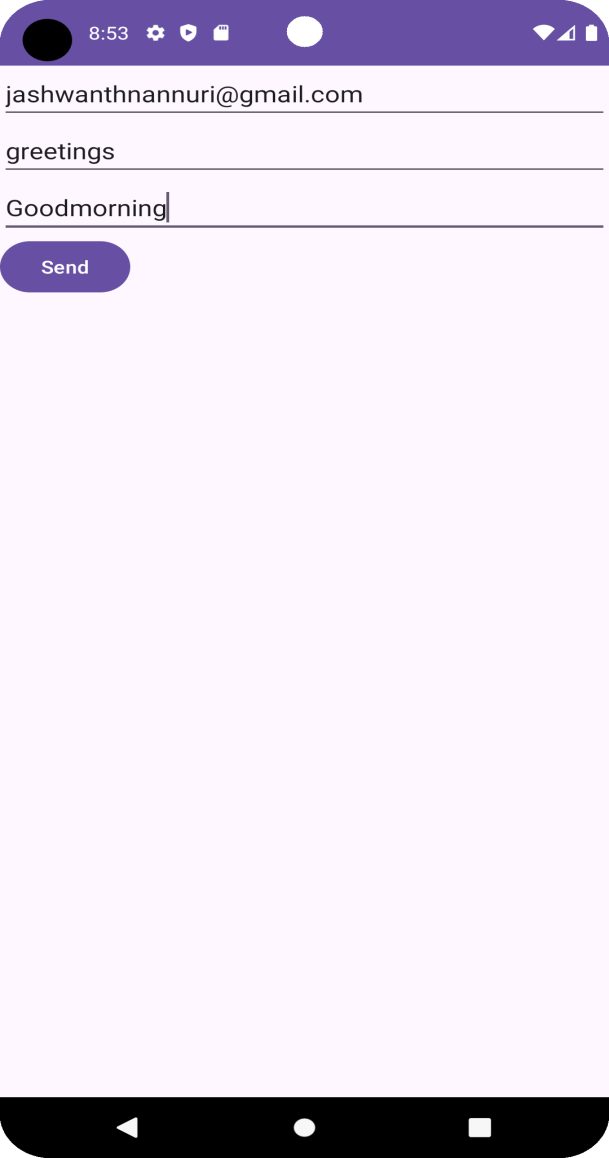
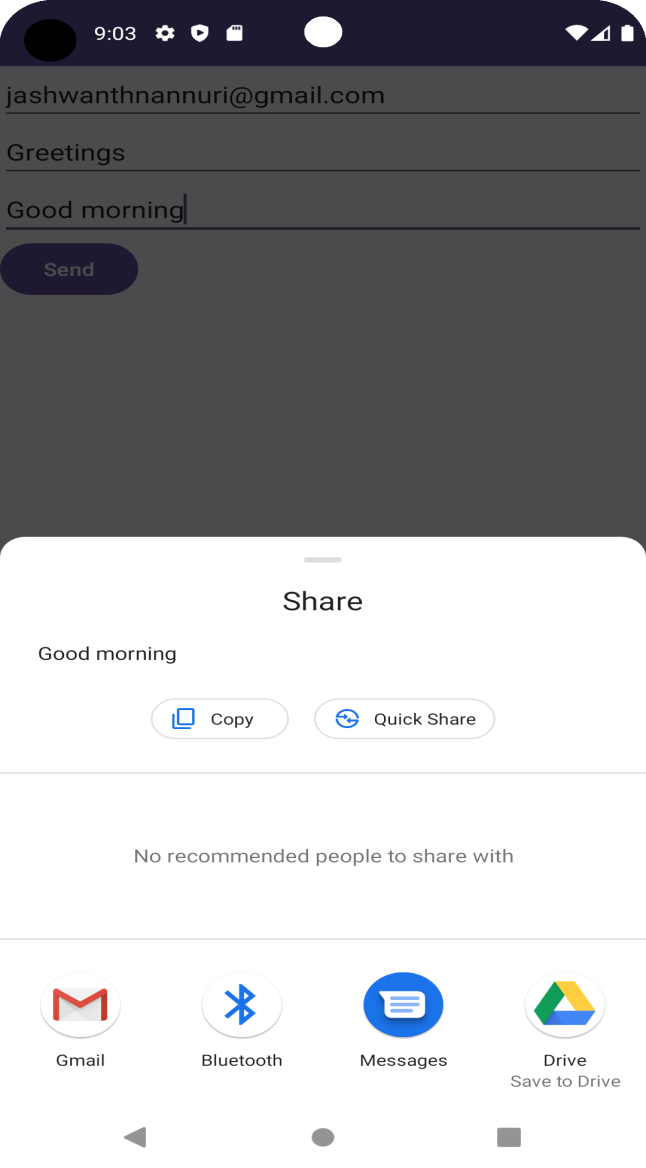
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <EditText  
  
 android:id="@+id/editTextTo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="To" />  
  
 <EditText  
 android:id="@+id/editTextSubject"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/editTextTo"  
 android:hint="Subject" />  
  
 <EditText  
 android:id="@+id/editTextMessage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/editTextSubject"  
 android:hint="Message"/>  
  
 <Button  
 android:id="@+id/buttonSend"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/editTextMessage"  
 android:text="Send" />  
</RelativeLayout>

**6. MainActivity.java:**

package com.example.lab\_11pro;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.annotation.SuppressLint;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
public class MainActivity extends AppCompatActivity {  
 EditText editTextTo, editTextSubject, editTextMessage;  
 Button buttonSend;  
 @Override  
 protected void onCreate(Bundle savedInstanceState)  
 {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 editTextTo = findViewById(R.id.*editTextTo*);  
 editTextSubject = findViewById(R.id.*editTextSubject*);  
 editTextMessage = findViewById(R.id.*editTextMessage*);  
 buttonSend = findViewById(R.id.*buttonSend*);  
 buttonSend.setOnClickListener(new View.OnClickListener()  
 {  
 @Override  
 public void onClick(View v)  
 {  
 sendEmail();  
 }  
 });  
 }  
 @SuppressLint("QueryPermissionsNeeded")  
 private void sendEmail() {  
 String to = editTextTo.getText().toString().trim();  
 String subject = editTextSubject.getText().toString().trim();  
 String message = editTextMessage.getText().toString().trim();  
 Intent intent = new Intent(Intent.*ACTION\_SEND*);  
 intent.setType("text/plain");  
 intent.putExtra(Intent.*EXTRA\_EMAIL*, new String[]{to});  
 intent.putExtra(Intent.*EXTRA\_SUBJECT*, subject);  
 intent.putExtra(Intent.*EXTRA\_TEXT*, message);  
 if (intent.resolveActivity(getPackageManager()) != null)  
 {  
 startActivity(Intent.*createChooser*(intent, "Choose an email client"));  
 }  
 }  
}

**Outputs:**





**Imp methods:**

* **@SuppressLint("QueryPermissionsNeeded")**: This annotation is used to suppress lint warnings specifically related to the need for permissions ("QueryPermissionsNeeded"). It tells the lint tool to ignore warnings related to missing permissions checks for this particular method or statement.
* Android lint checks are static code analysis tools provided by Android Studio that help developers identify potential issues, best practices violations, and optimizations in their code. These checks are helpful for maintaining code quality and ensuring adherence to Android development guidelines.
* When you perform certain operations in Android that require permissions, lint might issue warnings if you haven't explicitly declared these permissions in your **AndroidManifest.xml file.** For example, querying a content provider or accessing sensitive information may trigger such warnings.
* **sendEmail**() In Android, sending an email programmatically involves using an Intent with appropriate data and MIME types. This approach leverages the user's chosen email client to compose and send the email.
* Create an Intent with **action Intent.ACTION\_SENDTO**. This action is suitable for sending data to someone else.

**Use intent.setData(Uri.parse("mailto:"))** to specify that this intent is for sending an email. This ensures that only email apps handle this intent.\

* **Use intent.putExtra(Intent.EXTRA\_EMAIL, recipients)** to specify the recipients of the email. You can provide multiple email addresses in an array.
* **Use intent.putExtra(Intent.EXTRA\_SUBJECT, "Subject of the Email")** to set the subject of the email.
* **Use intent.putExtra(Intent.EXTRA\_TEXT, "Body of the Email")** to set the body or content of the email.
* Before starting the activity, **use intent.resolveActivity(getPackageManager())** to verify that there's an app available to handle the intent. This ensures that the email can be sent without crashing the app.
* **Fallback**: Always handle the case where there's no email app available to handle the intent (intent.resolveActivity(getPackageManager()) == null).
* **User Interaction**: The user will still need to confirm and send the email using their chosen email app.

1. **Display Map based on the Current/given location.**

1**. Click Start- Android Studio**, a Welcome to Android Studio dialog box will appear. Click New Project, the **New Project Dialog box appears**.

2. **Choose Empty Views Activity** then click Next.

3. Specify the Name of your project, Select the **Language as Java**, and Select the **SDK as API 24(“Nougat”,Android 7.0).**Click Finish Button.

4. Add a new **GoogleMaps activity**; (right click java folder->New- >**GoogleMaps Views Activity**)

A new googlemaps activity with corresponding xml file is generated with a ready made code

5. Generate the API Key

For security and authentication purposes to use google maps ,we need to generate an API key for our project . From the manifest file ,we get a link in the comment of the automatically generated code :

i) Copy the URL(**http://console.cloud.google.com**) in the browser address bar and create your account

ii) A project will be created by default

iii) Click on **Navigation menu** (top left) and select **APIs and Services**-> **Credentials** iv) API key is generated ,copy this key in manifest file as explained further

**6. activity\_main.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<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=".MainActivity">  
 <fragment  
 android:id="@+id/map"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true" />  
</RelativeLayout>

**7. MainActivity.java:**

package com.example.googlemap;  
  
import android.os.Bundle;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.MarkerOptions;  
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {  
 private GoogleMap mMap;  
 private double latitude = 12.911210;  
 private double longitude = 77.610083;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
 SupportMapFragment mapFragment = (SupportMapFragment)  
 getSupportFragmentManager()  
 .findFragmentById(R.id.*map*);  
 if (mapFragment != null) {  
 mapFragment.getMapAsync(this);  
 } else {  
 Toast.*makeText*(this, "Map Fragment Not Found", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 @Override  
 public void onMapReady(@NonNull GoogleMap googleMap) {  
 mMap = googleMap;  
 // Add a marker at current or given location and move the camera  
 LatLng location = new LatLng(latitude, longitude);  
 mMap.addMarker(new MarkerOptions().position(location).title("Marker"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(location, 15));  
 }  
}

**8. MapsActivity.java:**

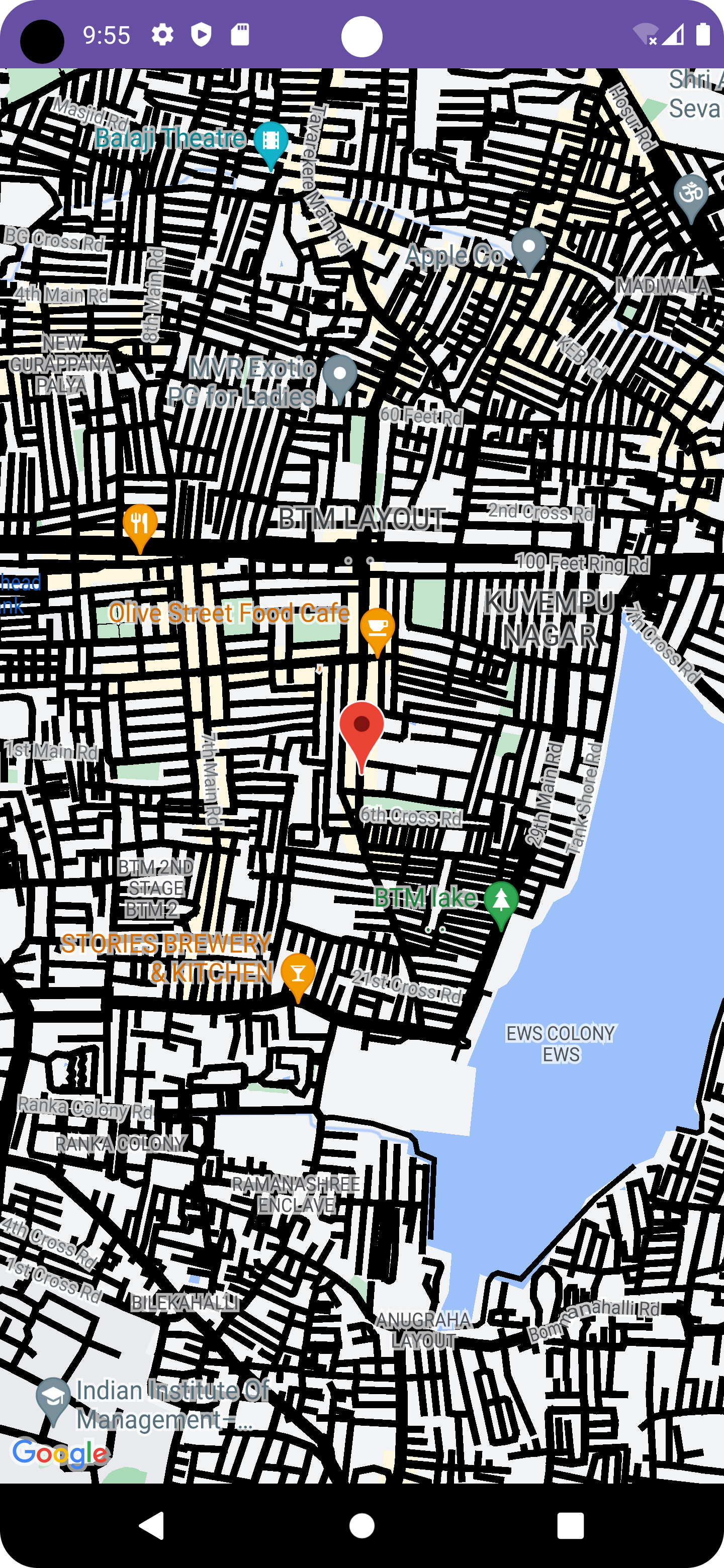
package com.example.googlemap;  
import androidx.fragment.app.FragmentActivity;  
import android.os.Bundle;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.MarkerOptions;  
import com.example.googlemap.databinding.ActivityMapsBinding;  
  
public class MapsActivity extends FragmentActivity implements  
 OnMapReadyCallback {  
 private GoogleMap mMap;  
 private ActivityMapsBinding binding;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityMapsBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
 SupportMapFragment mapFragment = (SupportMapFragment)  
 getSupportFragmentManager().findFragmentById(R.id.*map*);  
 mapFragment.getMapAsync(this);  
 }  
 @Override  
 public void onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
 // Add a marker in India and move the camera latitude and  
 //longitude -12.911210, 77.610083  
 LatLng india = new LatLng(12.911210, 77.610083);  
 mMap.addMarker(new MarkerOptions().position(india).title("I am here"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(india,  
 15));  
 }  
 }

**9. AndroidManifest.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
 <uses-permission  
 android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
 <uses-permission  
 android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
 <uses-permission  
 android:name="android.permission.INTERNET" />  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Googlemap"  
 tools:targetApi="31">

<!--  
 *TODO: Before you run your application, you need a Google Maps API key.*  
  
 To get one, follow the directions here:  
 https://developers.google.com/maps/documentation/android-sdk/get-api-key  
  
 Once you have your API key (it starts with "AIza"), define a new property in your  
 project's local.properties file (e.g. MAPS\_API\_KEY=Aiza...), and replace the  
 "YOUR\_API\_KEY" string in this file with "${MAPS\_API\_KEY}".  
 -->  
 <meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="AIzaSyDgxTtOPfZITHZuxKZ0HOPfdfscj3fuBb4" />  
  
 <activity  
 android:name=".MapsActivity"  
 android:exported="false"  
 android:label="@string/title\_activity\_maps" />  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
</manifest>

**Outputs:**



**Imp methods:**

* In Android development with Google Maps SDK, CameraUpdateFactory is a class provided by the Google Maps Android API utility library. It offers methods to create CameraUpdate objects, which allow you to animate the camera position and zoom level on the map.
* **Latitude**: It specifies the north-south position relative to the Equator. The Equator is at 0 degrees latitude, with values ranging from -90 degrees (South Pole) to +90 degrees (North Pole).
* **Longitude**: It specifies the east-west position relative to the Prime Meridian, which passes through Greenwich, England. Values range from -180 degrees (west of the Prime Meridian) to +180 degrees (east of the Prime Meridian).
* In Android development, particularly when integrating Google Maps into an application, the mMap = googleMap statement is a crucial part of the initialization process. When you implement the OnMapReadyCallback interface in your activity or fragment, you override the **onMapReady**(GoogleMap googleMap) method.
* GoogleMap is a class provided by the Google Maps Android API. It represents the main object that you interact with to manipulate and display a map.
* The googleMap parameter passed into onMapReady is an instance of GoogleMap that is initialized by the Google Maps API when the map is ready to be used.

1. **Create a sample application with login module (check user name and password) On successful login change Textview “Login Successful”. On login fail alert using Toast “login fail”**

1. **Click Start- Android Studio**, a Welcome to Android Studio dialog box will appear. Click New Project, the New Project Dialog box appears.

2. Choose **Empty Views Activity** then click Next.

3. Specify the Name of your project, Select the Language as Java, and Select the **SDK as API 24(“Nougat”,Android 7.0)**.Click Finish Button.

4. Update the following code in **activity\_main.xml** and **MainActivity.java**

5. Click Run app or shift+F10 to execute the application.

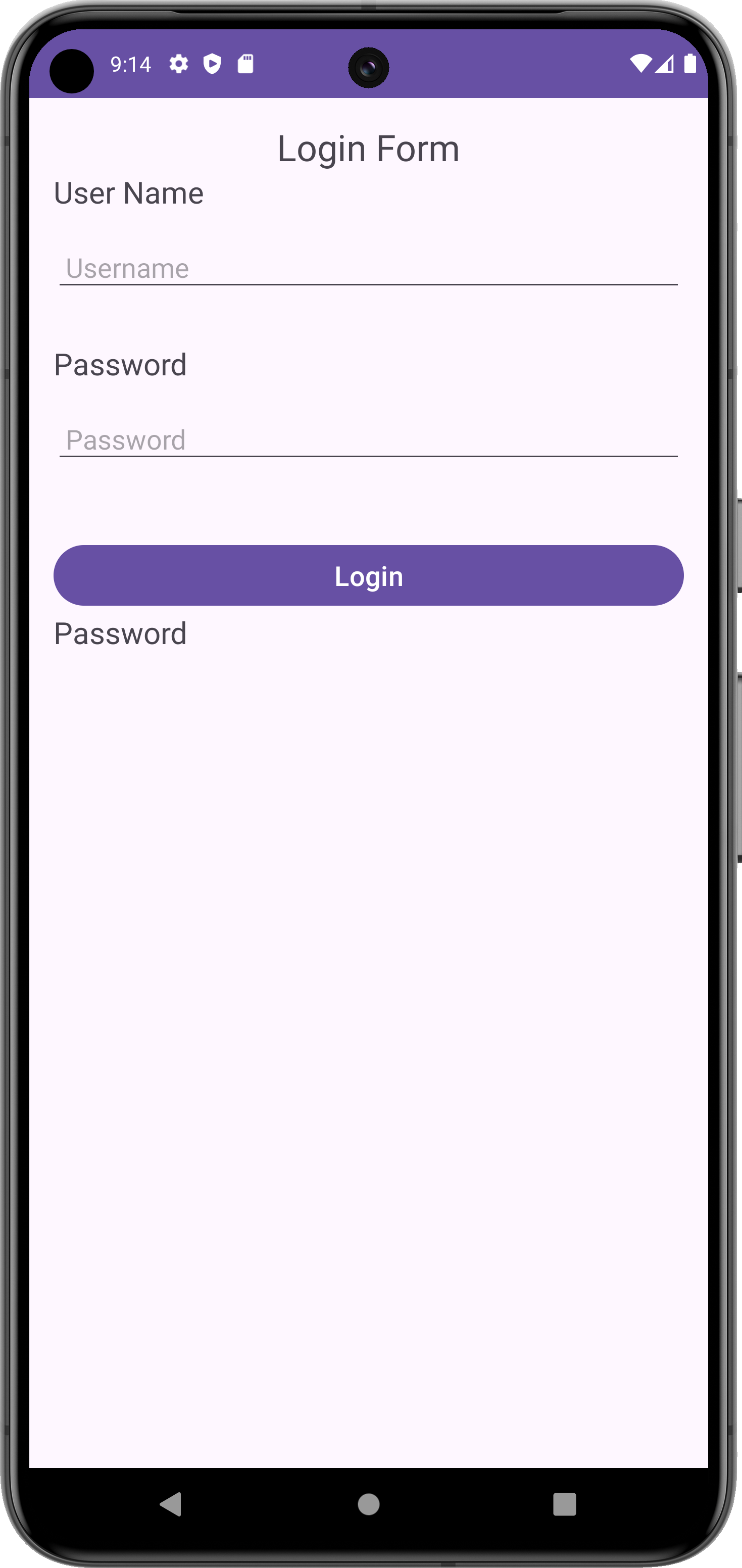
6. **activity\_main.xml:**

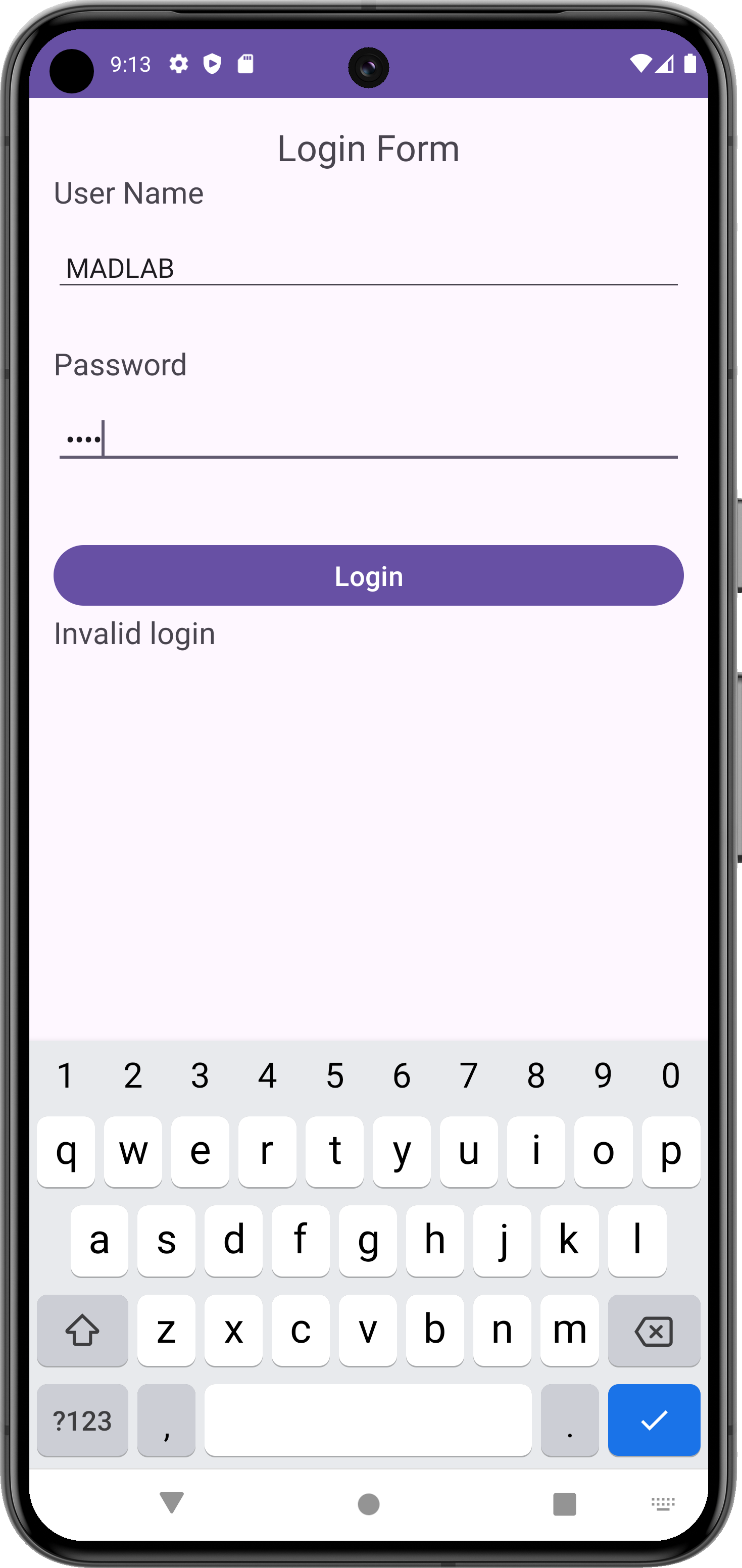
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:orientation="vertical"  
 android:padding="16dp">  
 <TextView android:id="@+id/tvTitle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="24sp"  
 android:text="Login Form"  
 android:layout\_gravity="center"/>  
 <TextView android:id="@+id/tvUserName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:text="User Name" />  
 <EditText  
 android:id="@+id/etUsername"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Username"  
 android:inputType="text"  
 android:padding="8dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginBottom="30dp"/>  
 <TextView android:id="@+id/tvPassword"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:text="Password" />  
 <EditText  
 android:id="@+id/etPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:inputType="textPassword"  
 android:padding="8dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginBottom="30dp"/>  
 <Button  
 android:id="@+id/btnLogin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Login"  
 android:textSize="18sp"  
 android:layout\_marginTop="16dp"/>  
 <TextView android:id="@+id/tvMessage"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="20sp"  
 android:text="Password" />  
</LinearLayout>

**7. MainActivity.java:**

package com.example.lab\_13;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
public class MainActivity extends AppCompatActivity {  
 EditText etUsername,etPassword; Button btnLogin;  
 TextView tvMessage;  
 @Override  
 protected void onCreate(Bundle savedInstanceState)  
 {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 btnLogin = (Button) findViewById(R.id.*btnLogin*);  
 etUsername = (EditText) findViewById(R.id.*etUsername*);  
 etPassword = (EditText) findViewById(R.id.*etPassword*);  
 tvMessage = (TextView) findViewById(R.id.*tvMessage*);  
 btnLogin.setOnClickListener(new View.OnClickListener()  
 { @Override  
 public void onClick(View view)  
 {  
 if(etUsername.getText().toString().isEmpty())  
 {  
 etUsername.setError("Enter User name");  
 } else if (etPassword.getText().toString().isEmpty()) {  
 etPassword.setError("Enter Password");  
 }  
 else if(etUsername.getText().toString().equals("MADLAB") &&  
 etPassword.getText().toString().equals("1234"))  
 {  
 tvMessage.setText("Valid Login");  
 }  
 else  
 {  
 tvMessage.setText("Invalid login");  
 }  
 }  
 });  
 }  
}

**Outputs:**



**Imp methods:**

* In Android development, **tvMessage** typically refers to a TextView component within the layout XML file of an activity or fragment.
* **isEmpty()** is not a method directly available on all objects or data types by default, unlike in some other programming languages. However, depending on the context, isEmpty() can be used with specific data structures or classes to check if they contain any elements or data.

1. **Learn to deploy Android applications**
2. **Open Android Studio**: Launch Android Studio on your computer.
3. **Open Your Project**: Open the Android project for which you want to generate the signed APK.
4. **Build Menu**: Click on the "Build" menu at the top of the screen.
5. **Generate Signed Bundle / APK**: From the "Build" menu, select "Generate Signed Bundle / APK..." option.
6. **Select APK**: In the dialog that appears, choose "APK" and click "Next".
7. **Key Store Creation**: If you haven't already created a keystore, you'll need to do so. Click on "Create new..." and follow the prompts. Remember to remember the details you use here as you'll need them for future updates.
8. **Fill in Key Store Information**: Fill in the required fields in the "New Key Store" dialog: Key store path, Password, Alias, Password for key, Validity (years), First and Last name, Organization, City or locality, and State or province.
9. **Build Types**: Choose the build type (debug or release) and configure it accordingly. For a release build, ensure that "Signature Versions" is set to "V1 (Jar Signature)" or "V1 (Jar Signature) and V2 (Full APK Signature)".
10. **Destination Folder**: Choose where you want to save the signed APK and click "Next".
11. **Build**: Review the build settings and click on "Finish" to start the process of building the signed APK.
12. **Wait**: Android Studio will take some time to compile and sign the APK.
13. **Confirmation**: Once the process is complete, you'll receive a confirmation message along with the location of the signed APK.
14. **Locate APK**: Navigate to the location where the signed APK is saved.
15. **Distribution**: You can now distribute the signed APK to users or upload it to the Google Play Store if you're publishing your app.

**Note:** Remember to keep your keystore file and passwords secure, as they are needed for future updates to your app.

**1. activity\_main.xml:**

xml version="1.0" encoding="utf-8"?>  
 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#E69B9B"  
 tools:context=".MainActivity">

<TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 **android:background="#8ED3F3"  
 android:fontFamily="cursive"** android:text="Hello World!"  
 **android:textSize="34sp"  
 android:textStyle="bold"** app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**2. MainActivity.java:**

package com.example.myapplication;  
  
import android.os.Bundle;  
  
import androidx.activity.EdgeToEdge;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
 }  
}

Output:

