Mobile application development.

Lab Manual

1. Creating Hello World Application

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!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
android:textSize="30dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java package

```
com.example.helloworldapplication;
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);
}
}
```

output

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

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">
<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:supportsRtl="true"
android:theme="@style/Theme.SecondProgram"
tools:targetApi="31">

<activity
android:name=".NextActivity"
android:exported="false" android:screenOrientation="landscape"/>

<activity
android:name=".MainActivity"
android:exported="true"
android:screenOrientation="portrait">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

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/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"        android:layout_marginBottom="8dp"
android:layout_marginTop="112dp"
android:onClick="onClick"
android:text="Launch next activity"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.612"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editText1"
app:layout_constraintVertical_bias="0.613" />

<TextView
android:id="@+id/editText1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="124dp"
android:ems="10"
android:textSize="22dp"
android:text="This activity is portrait orientation"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.502"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

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

MainActivity.java

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

public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
public void onClick(View v) {
Intent intent = new Intent(MainActivity.this,NextActivity.class);
startActivity(intent);
}}
```

activity_next.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=".NextActivity">

<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="180dp"
android:text="this is landscape orientation"
android:textSize="22dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.502"
```

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />

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

NextActivity.java package

com.example.secondprogram;

Import ndroidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class NextActivity **extends** AppCompatActivity {

@Override

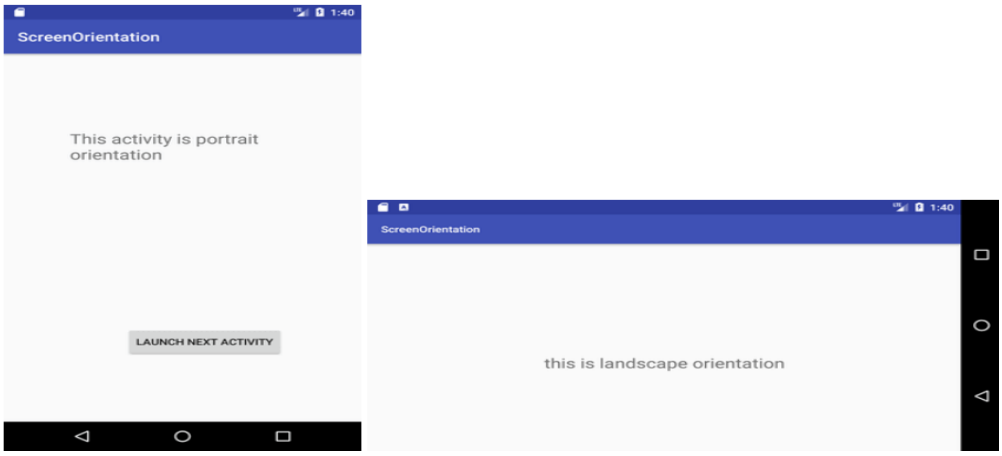
protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.***activity_next***);

}}

Output:



3. Creating an application to develop Login window using UI controls.

<?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"

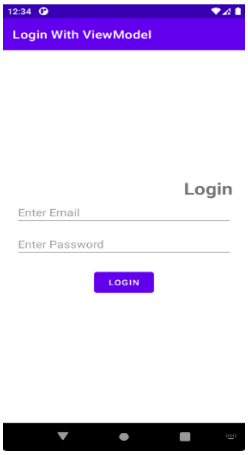
android:textSize="24sp"

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"/>

</LinearLayout>



4. Create an application to implement new activity using explicit intent and implicit intent. /*

Explicit Intent: This involves navigating from one activity to another within the same application.

Implicit Intent: This involves triggering an action that can be handled by another application. */

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="30dp">

<Button

android:id="@+id/btnExplicitContent"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:text="Explicit Content"

android:textSize="30sp"

android:layout_marginTop="30dp"></Button>

</LinearLayout>

MainActivity.java

package com.example.fourthprogram;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

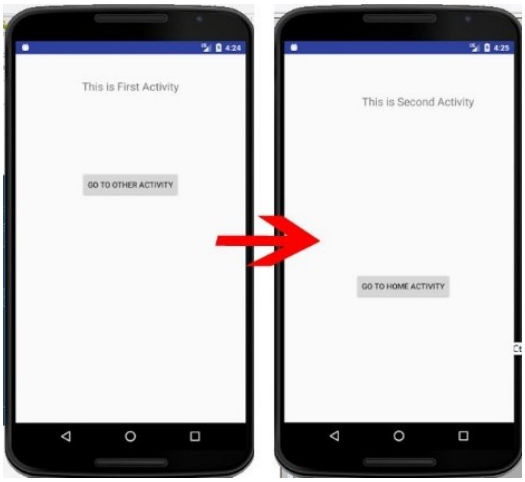
```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
Button btnExplicitContent;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btnExplicitContent=findViewById(R.id.btnExplicitContent);
btnExplicitContent.setOnClickListener(new View.OnClickListener() {            @Override
public void onClick(View view) {
Intent intent = new Intent(MainActivity.this,
SecondActivity.class);
startActivity(intent);
}
});
}
}
```

activity_second.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=".SecondActivity"
android:orientation="vertical"
android:padding="30dp">

<Button
android:id="@+id/btnImplicitContent"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Implicit Content"
android:textSize="30sp"
android:layout_marginTop="30dp"></Button>
</LinearLayout>
```

SecondActivity.java

```
package com.example.fourthprogram;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class SecondActivity extends AppCompatActivity {

Button btnImplicitContent;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_second);
btnImplicitContent=findViewById(R.id.btnImplicitContent);
btnImplicitContent.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Uri webpage = Uri.parse("https://www.google.com");
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
startActivity(intent);
}
});
}}
```



5. Create an application that displays custom designed Opening Screen

Step 1: Create a New Project in Android Studio

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio.

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:id="@+id/idRLContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

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

    <TextView
        android:id="@+id/idTVHeading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="20dp"
        android:gravity="center"
        android:padding="10dp"
        android:text="Background Drawable in Android"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:textStyle="bold" />
 </RelativeLayout>
```

back_drawable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <!--on below line we are adding gradient and
    specifying start and end color with angle-->
    <gradient
        android:angle="270"
        android:endColor="@color/white"
        android:startColor="#0F9D58" />
 </shape>
```

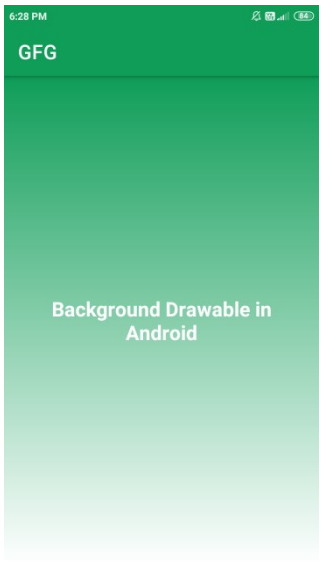
Step 4: Working with the MainActivity file

MainActivity.java

```
package com.gtappdevelopers.kotlingfgproject;
 import android.os.Bundle;
import android.widget.RelativeLayout;
import androidx.appcompat.app.AppCompatActivity;
 public class MainActivity extends AppCompatActivity {
```

```
    // on the below line we are creating a variable.
    private RelativeLayout containerRL;

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

        // on below line we are initializing variables with ids.
        containerRL = findViewById(R.id.idRLContainer);
         // on below line we are setting background for
        // our relative layout on below line.
        containerRL.setBackground(getResources().getDrawable(R.drawable.back_drawable));
    }
}
```

Output:



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

Create a new layout

When adding a new layout for your app, first create a default layout file in your project's default layout/ directory so that it applies to all device configurations. Once you have a default layout, you can create layout variations, as described in a section on this page, for specific device configurations, such as for large screens.

You can create a new layout in one of the following ways:

Use Android Studio's main menu

1. In the **Project** window, click the module you want to add a layout to.

2. In the main menu, select **File > New > XML > Layout XML File**.
3. In the dialog that appears, provide the filename, the root layout tag, and the source set where the layout belongs.
4. Click **Finish** to create the layout.

Use the Project view

1. Choose the **Project** view from within the **Project** window.
2. Right-click the layout directory where you'd like to add the layout.
3. In the context menu that appears, click **New > Layout Resource File**.

Use the Android view

1. Choose the **Android** view from within the **Project** window.
2. Right-click the layout folder.
3. In the context menu that appears, select **New > Layout Resource File**.

Use the Resource Manager

1. In the Resource Manager, select the **Layout** tab.
2. Click the + button, and then click **Layout Resource File**.

Find items in the Palette

To search for a view or view group by name in the **Palette**, click the **Search**  button at the top of the palette. Alternatively, you can type the name of the item whenever the **Palette** window has focus.

In the **Palette**, you can find frequently used items in the **Common** category. To add an item to this category, right-click a view or view group in the **Palette** and then click **Favorite** in the context menu.

Open documentation from the Palette

To open the Android Developers reference documentation for a view or view group, select the UI element in the **Palette** and press Shift+F1.

Add sample data to your view

Because many Android layouts rely on runtime data, it can be difficult to visualize the look and feel of a layout while designing your app. You can add sample preview data to a TextView, an ImageView, or a RecyclerView from within the Layout Editor.

Note: When you add sample data to a **View**, Android Studio makes changes to your project as though you were using your own data. You can then modify these changes as needed.

To display the **Design-time View Attributes** window, right-click one of these view types and choose **Set Sample Data**, as shown in figure 6.

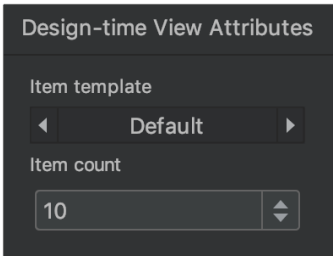


Figure . The **Design-time View Attributes** window.

For a TextView, you can choose between different sample text categories. When using sample text, Android Studio populates the text attribute of the TextView with your chosen sample data. Note that you can choose sample text via the **Design-time View Attributes** window only if the text attribute is empty.

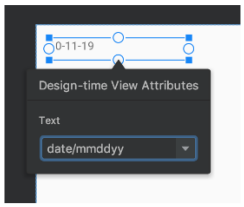


Figure . A TextView with sample data.

For an ImageView, you can choose between different sample images. When you choose a sample image, Android Studio populates the tools:src attribute of the ImageView (or tools:srcCompat if using AndroidX).

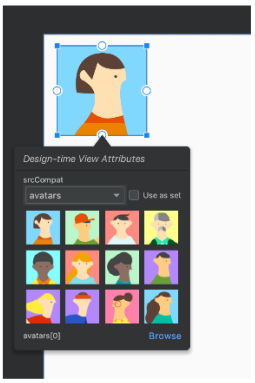


Figure . An ImageView with sample data.

For a RecyclerView, you can choose from a set of templates that contain sample images and texts. When using these templates, Android Studio adds a file to your res/layout directory, recycler_view_item.xml, that contains the layout for the sample data. Android Studio also adds metadata to the RecyclerView to properly display the sample data.

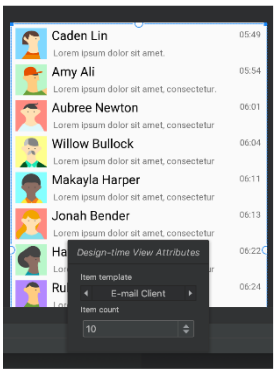


Figure . A RecyclerView with sample data.

Download fonts and apply them to text

When using Android 8.0 (API level 26) or the Jetpack Core library, you can select from hundreds of fonts by following these steps:

1. In the Layout Editor, click the **Design**  icon to view your layout in the design editor.
2. Select a text view.
3. In the **Attributes** panel, expand **textAppearance**, and then expand the **fontFamily** box.
4. Scroll to the bottom of the list and click **More Fonts** to open the **Resources** dialog.
5. In the **Resources** dialog, to select a font, browse the list or type into the search bar at the top. If you select a font under **Downloadable**, then you can either click **Create downloadable font** to load the font at runtime as a downloadable font or click **Add font to project** to package the TTF font file in your APK. The fonts listed under **Android** are provided by the Android system, so they don't need to be downloaded or bundled in your APK.
6. Click **OK** to finish.

Layout Validation

Layout Validation is a visual tool for simultaneously previewing layouts for different devices and display configurations, helping you catch problems in your layouts earlier in the process. To access this feature, click the **Layout Validation** tab in the top-right corner of the IDE window:

Figure . Layout Validation tab.

To switch between the available configuration sets, select one of the following from the **Reference Devices** drop-down at the top of the Layout Validation window:

- Reference Devices
- Custom
- Color Blind
- Font Sizes

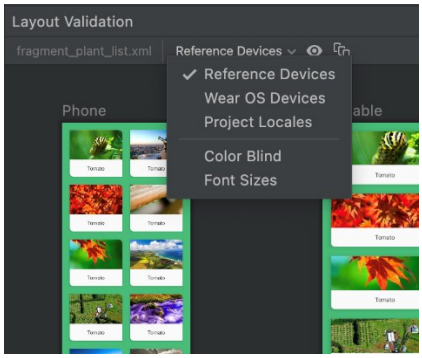


Figure. Reference Devices drop-down.

Custom

To customize a display configuration to preview, choose from a variety of settings including language, device, or screen orientation:

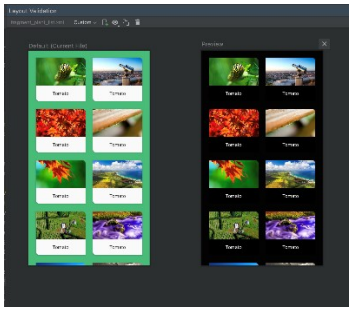


Figure. Configure a custom display in the Layout Validation tool.

Color Blind

To help make your app more accessible for users who are color blind, validate your layout with simulations of common types of color blindness:

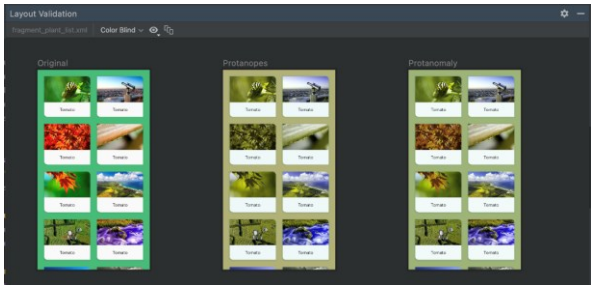


Figure. Color blindness simulation previews in the Layout Validation tool.

Font Sizes

Validate your layouts at various font sizes, and improve your app's accessibility for visually impaired users by testing your layouts with larger fonts:

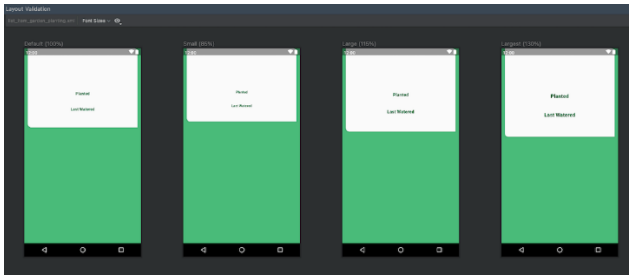


Figure . Variable font size previews in the Layout Validation tool.

7. **Create menu in application.**

res/menu/main_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/action_settings"
android:title="Settings"
android:icon="@drawable/ic_setting" />
</menu>
```

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

MainActivity.java package

```
com.example.seventhprogram;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main_menu, menu);
return true;}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
int id = item.getItemId();
// Handle item selection
if (id == R.id.action_settings) {
// Open settings activity or perform desired action
Intent intent = new Intent(getApplicationContext(),
MainActivity2.class);
startActivity(intent); // Start MainActivity2
return true;}
return super.onOptionsItemSelected(item);
}}
```

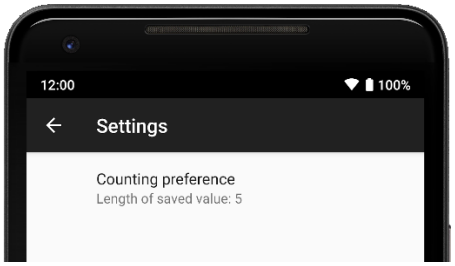
activity_main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity2">

<TextView
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Settings Page"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity2.java

```
package com.example.seventhprogram;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity2 extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main2);
}
}
```



8. Read / Write the Local data.

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="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>

<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>
```

MainActivity.java package

```
com.example.eigthprogram;
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) {
// Writing data to SharedPreferences
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);
});}}
```

activity_main2.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=".MainActivity2"
```

```
android:orientation="vertical">

<Button
android:id="@+id/btnFetch"
android:layout_width="wrap_content"
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>
```

MainActivity2.java package

```
com.example.eigthprogram;
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) {
// Reading data from SharedPreferences
SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE);
String username = sharedPreferences.getString("username", "");
String password = sharedPreferences.getString("password", "");
etUserName.setText(username);
etPassword.setText(password);
}});}}
```

9.Create / Read / Write data with database (SQL Lite)

HOW TO CREATE AN SQLITE DATABASE:

In the AndroidManifest.xml file you add permission to access the storage.

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

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"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/etna"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Please enter name"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/etcell"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Please enter cell no"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/bsubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="submit"
        android:text="SUBMIT" />

    <Button
        android:id="@+id/bshow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="showdata"
        android:text="SHOW DATA" />

    <Button
        android:id="@+id/bedit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="edit"
        android:text="EDIT DATA " />

    <Button
        android:id="@+id/bdele"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="delete"
        android:text="DELETE DATA" />
</LinearLayout>
```

activity_data

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/data"
        android:textSize="18sp"
        android:textStyle="bold" />
</LinearLayout>
```

Data.java

```
package com.example.sqlitedatabasesavedata;

import androidx.appcompat.app.AppCompatActivity;
import android.database.SQLException;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

public class Data extends AppCompatActivity {
  TextView textView;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_data);
    textView = findViewById(R.id.textView);
    try {
      ContactsDB db = new ContactsDB(this);
      db.open();
      textView.setText(db.returndata());
      db.close();
    } catch (SQLException e) {
      Toast.makeText(Data.this, e.getMessage(), Toast.LENGTH_LONG)
        .show();
    }
  }
}
```

Create SQLiteOpenHelper class.

ContactsDB.java

```
package com.example.sqlitedatabasesavedata;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class ContactsDB {
```

```
  public static final String Key_RowId = "_id";
  public static final String Key_Name = "person_name";
  public static final String Key_Cell = "_cell";

  private final String Database_Name = "ContactsDB"; //Database Name
  private final String Database_Table = "ContactsTavle";
  private final int Database_Version = 1;

  private DBHelper ourHelper;
  private final Context ourContext;
  private SQLiteDatabase ourdatabase;

  public ContactsDB(Context context) {
    ourContext = context;
  }

  private class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) {
      super(context, Database_Name, null, Database_Version);

    }
    @Override
    public void onCreate(SQLiteDatabase db) {
      String sqlcode = "CREATE TABLE ContactsTable(_id INTEGER PRIMARY KEY
AUTOTNCREMENT, person_name TEXT NOT NULL, _cell TEXT NOT NULL);";
      db.execSQL(sqlcode);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
      db.execSQL("DROP TABLE IF EXISTS " + Database_Table);
      onCreate(db);
    }
  }
  public ContactsDB open() throws SQLException {
    ourHelper = new DBHelper(ourContext);
    ourdatabase = ourHelper.getWritableDatabase();
    return this;
  }
  public void close() {
    ourHelper.close();
  }
  public long creat(String name, String cell) {
    ContentValues cv = new ContentValues();
    cv.put(Key_Name, name);
    cv.put(Key_Cell, cell);
    return ourdatabase.insert(Database_Table, null, cv);
  }
  public String returndata() {
    String[] column = new String[] {
      Key_RowId,
      Key_Name,
      Key_Cell
    };
    Cursor c = ourdatabase.query(Database_Table, column, null, null, null, null, null);
    String resu = "";
    int irowid = c.getColumnIndex(Key_RowId);
    int iname = c.getColumnIndex(Key_Name);
```

```
    int icell = c.getColumnIndex(Key_Cell);
    for (c.moveToFirst(); c.isAfterLast(); c.moveToNext()) {
      resu = resu + c.getString(irowid) + ":" + c.getString(iname) + " " + c.getString(icell) + "\n";
    }
    c.close();
    return resu;
  }
  public long deleteEnter(String rowId) {
    return ourdatabase.delete(Database_Table, Key_RowId + "=?", new String[] {
      rowId
    });
  }

  public long update(String rowId, String cell, String name) {
    ContentValues cu = new ContentValues();
    cu.put(Key_Name, name);
    cu.put(Key_Cell, cell);
    return ourdatabase.update(Database_Table, cu, Key_RowId + "=?", new String[] {
      rowId
    });
  }
}
```

MainActivity.java

```
package com.example.sqlitedatabasesavedata;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.database.SQLException;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.Toolbar;

public class MainActivity extends AppCompatActivity {
  EditText etname, etcell;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    etname = findViewById(R.id.etna);
    etcell = findViewById(R.id.etcell);
  }
  public void showdata(View v) {
    startActivity(new Intent(this, Data.class));
  }

  public void submit(View v) {
    String name = etname.getText()
      .toString()
      .trim();
    String cell = etcell.getText()
      .toString()
      .trim();
    try {
      ContactsDB db = new ContactsDB(this);
```

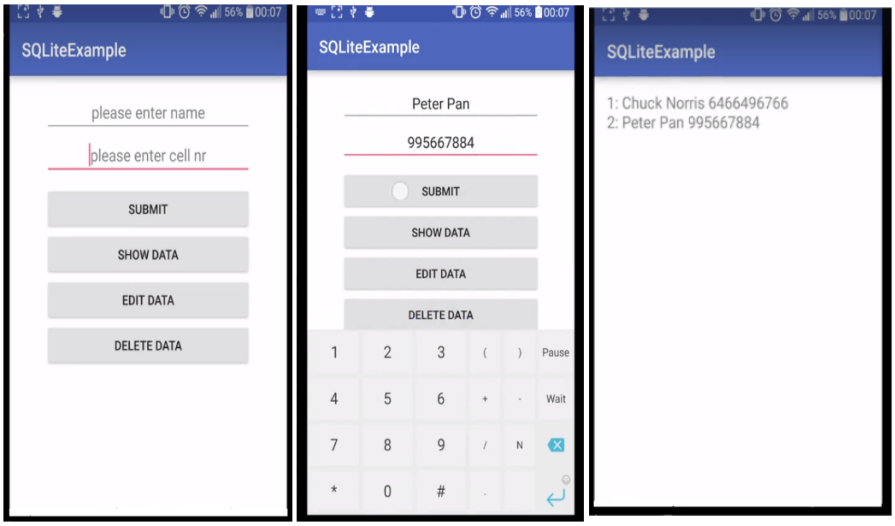
```
      db.open();
      db.creat(name, cell);
      db.close();
      Toast.makeText(MainActivity.this, "Successfully saved ", Toast.LENGTH_LONG)
        .show();
      etname.setText("");
      etcell.setText("");
    } catch (SQLException e) {
      Toast.makeText(MainActivity.this, e.getMessage(), Toast.LENGTH_LONG)
        .show();
    }
  }

  public void edit(View v) {
    try {
      ContactsDB db = new ContactsDB(this);
      db.open();
      db.update("1", "John", "24334421");
      db.close();
      Toast.makeText(MainActivity.this, "Successfully updated", Toast.LENGTH_LONG)
        .show();
    } catch (SQLException e) {
      Toast.makeText(MainActivity.this, e.getMessage(), Toast.LENGTH_LONG)
        .show();
    }
  }

  public void delete(View v) {
    try {
      ContactsDB db = new ContactsDB(this);
      db.open();
      db.deleteEnter("1");
      Toast.makeText(MainActivity.this, "Successfully delete", Toast.LENGTH_LONG)
        .show();
      db.close();
    } catch (SQLException e) {
      Toast.makeText(MainActivity.this, e.getMessage(), Toast.LENGTH_LONG)
        .show();
    }
  }
}
```

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"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/etna"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
```

```
        android:hint="Please enter name"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/etcell"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Please enter cell no"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/bsubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="submit"
        android:text="SUBMIT" />

    <Button
        android:id="@+id/bshow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="showdata"
        android:text="SHOW DATA" />

    <Button
        android:id="@+id/bedit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="edit"
        android:text="EDIT DATA " />

    <Button
        android:id="@+id/bdele"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="delete"
        android:text="DELETE DATA" />
</LinearLayout>
```

Output:



10. Create an application to send SMS and receive SMS.

How to Send SMS in Android?

Now we will implement SMS in our ***application*** and see how it works:
Step 1: First of all as always, we will create a new project and we will name it. After that we will create the layout in **Activity_main.xml** as follows:

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

<androidx.constraintlayout.widget.ConstraintLayout
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">

<RelativeLayout

android:layout_width="match_parent"

android:layout_height="match_parent"

tools:context=".MainActivity">

<TextView

android:id="@+id/textView2"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_centerHorizontal="true"

android:layout_marginLeft="100dp"

android:layout_marginTop="100dp"

android:fontFamily="@font/arbutus"

android:text="DataFlair "

android:textColor="@color/colorPrimaryDark"

android:textSize="50dp" />

<TextView

android:id="@+id/textView1"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_alignParentTop="true"
```

```
android:layout_marginLeft="110dp"
android:layout_marginTop="184dp"
android:fontFamily="@font/arbutus"
android:text="SMS Service"
android:textColor="#EE47ADDD"
android:textSize="30dp" />
<EditText
android:id="@+id/editText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="100dp"
android:layout_marginTop="270dp"
android:hint=" Please Enter Phone Number"
android:textColorHint="#9FAEE9" />
<EditText
android:id="@+id/editText2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/editText"
android:layout_marginStart="100dp"
android:layout_marginLeft="100dp"
android:layout_marginTop="45dp"
android:hint="Please write the message "
android:textColorHint="#CE9C9C" />
<Button
android:id="@+id/btnSendSMS"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/editText2"
android:layout_centerHorizontal="true"
```

```
android:layout_marginTop="48dp"
android:text="Send Sms" />
</RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 2: Now we will write the code for **MainActivity.java** file as follows:

```
package com.DataFlair.smssample;
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
public class MainActivity extends Activity {
private static final int PERMISSION_RQST_SEND = 0;
Button button1;
EditText phoneNo;
EditText myMessage;
String phoneNo;
String message;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
//We'll create objects
button1 = (Button) findViewById(R.id.btnSendSMS);
phoneNo = (EditText) findViewById(R.id.editText);
```

```
myMessage = (EditText) findViewById(R.id.editText2);
button1.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
sendSMSMessage();
}});}
protected void sendSMSMessage() {
phoneNo = phoneNo.getText().toString(); //we'll get the phone number from the user
message = myMessage.getText().toString();//we'll get the Message from the user
//We'll check the permission is granted or not . If not we'll change
if (ContextCompat.checkSelfPermission(this,Manifest.permission.SEND_SMS) !=
PackageManager.PERMISSION_GRANTED) {
if (ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.permission.SEND_SMS)) {
}
else { ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.SEND_SMS},
PERMISSION_RQST_SEND);
}}}
//Now once the permission is there or not would be checked
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
switch (requestCode) {
case PERMISSION_RQST_SEND: {
if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNo, null, message, null, null);
Toast.makeText(getApplicationContext(), "SMS sent.",Toast.LENGTH_LONG).show();
} else {Toast.makeText(getApplicationContext(), "SMS failed, you may try again later.",
Toast.LENGTH_LONG).show();
return;
}}}}}
```

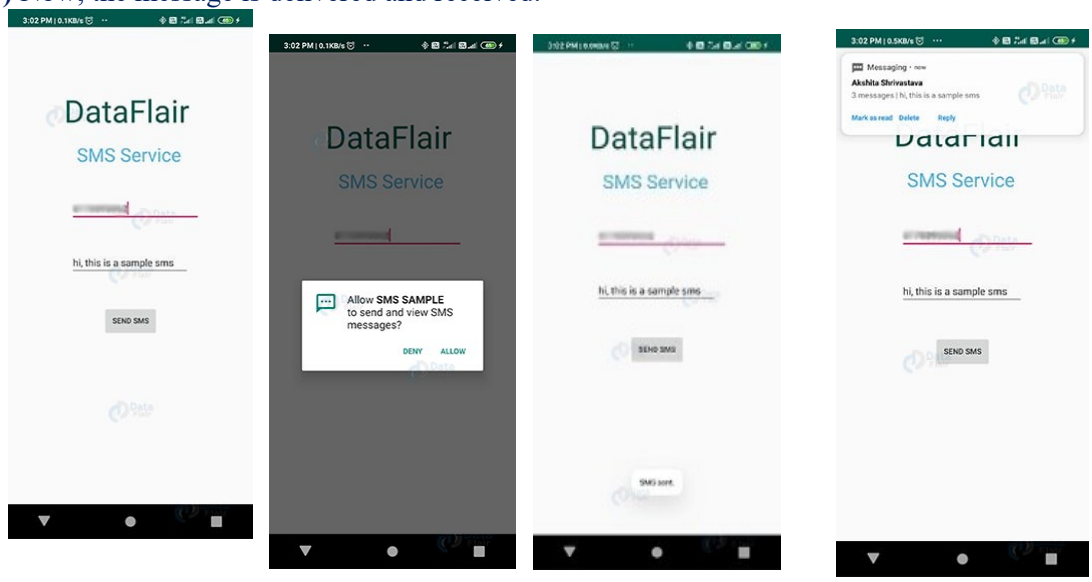
Step 3: Now we will update the **manifest.xml** file as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
package="com.DataFlair.smssample">

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

<application

android:allowBackup="true"

android:icon="@mipmap/ic_launcher"

android:label="@string/app_name"

android:roundIcon="@mipmap/ic_launcher_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity

android:name="com.DataFlair.smssample.MainActivity"

android:label="@string/app_name">

<intent-filter>

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

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

</intent-filter>

</activity>

<meta-data

android:name="preloaded_fonts"

android:resource="@array/preloaded_fonts" />

</application>

</manifest>
```

Step 4: After this, we will now implement our application as follows:
i) Our application would look like this.
ii) Now, we will **enter the number** and the **message**.
iii) After entering the number and message, we will **grant permission** to access our messages.
iv) After that, you will see that the message has been sent.

v) Now, the message is delivered and received.



11.Create an application to send an e-mail

Creating Layout of Send Email App

Here, we will create a layout for our app. So, for tutorial purposes, I am keeping it simple. For sending an email, we usually need three fields, i.e., To, Subject, and Message. So, we will be creating a layout for three fields with the help of TextView and EditText and finally, we need a Send button to send the email.

The XML code for our layout is shown below.

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"
tools:context="in.amitsin6h.sendemail.MainActivity"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="20px"
android:paddingRight="20px"
android:orientation="horizontal"
>

<EditText
    android:id="@+id/etTo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_marginRight="22dp"
    android:layout_marginTop="16dp"
    android:ems="10" />

<EditText
   android:id="@+id/etSub"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/etTo"
   android:layout_below="@+id/etTo"
```

```
  android:layout_marginTop="18dp"
  android:ems="10" >
</EditText>

<EditText
   android:id="@+id/etMsg"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/etSub"
   android:layout_below="@+id/etSub"
   android:layout_marginTop="28dp"
   android:ems="10"
   android:inputType="textMultiLine" />

<TextView
   android:id="@+id/textView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/etTo"
   android:layout_alignBottom="@+id/etTo"
   android:layout_alignParentLeft="true"
   android:text="To:" />

<TextView
   android:id="@+id/textView2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/etSub"
   android:layout_alignBottom="@+id/etSub"
   android:layout_alignParentLeft="true"
   android:text="Subject:" />

<TextView
   android:id="@+id/textView3"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/etMsg"
   android:layout_alignBottom="@+id/etMsg"
   android:layout_alignParentLeft="true"
   android:text="Message:" />

<Button
   android:id="@+id/btSend"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/etMsg"
   android:layout_below="@+id/etMsg"
   android:layout_marginLeft="76dp"
   android:layout_marginTop="20dp"
   android:text="Send" />

</RelativeLayout>
```

Step 3 – Send Email Android App Java Code

This is our main part where we will write the code for our app. We discussed in the introduction that we will be using intent to send an email. Intent uses the email service which will call the email client and send our app data from string to the email client and email client will use that data to send the email. It's simple and easy to do. :)

Just copy the below java code and paste it to MainActivity.java.

MainActivity.java

```
package in.amitsin6h.sendemail;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    EditText etTo, etSub, etMsg;
    Button btSend;
    String to, subject, message;

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

        etTo = (EditText) findViewById(R.id.etTo);
        etSub = (EditText) findViewById(R.id.etSub);
        etMsg = (EditText) findViewById(R.id.etMsg);

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

        btSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                to = etTo.getText().toString();
                subject = etSub.getText().toString();
                message = etMsg.getText().toString();


                Intent email = new Intent(Intent.ACTION_SEND);
                email.putExtra(Intent.EXTRA_EMAIL, new String[]{ to});
                email.putExtra(Intent.EXTRA_SUBJECT, subject);
                email.putExtra(Intent.EXTRA_TEXT, message);

                //need this to prompts email client only
                email.setType("message/rfc822");

                startActivity(Intent.createChooser(email, "Choose Email client :"));
            }
        });
    }
```

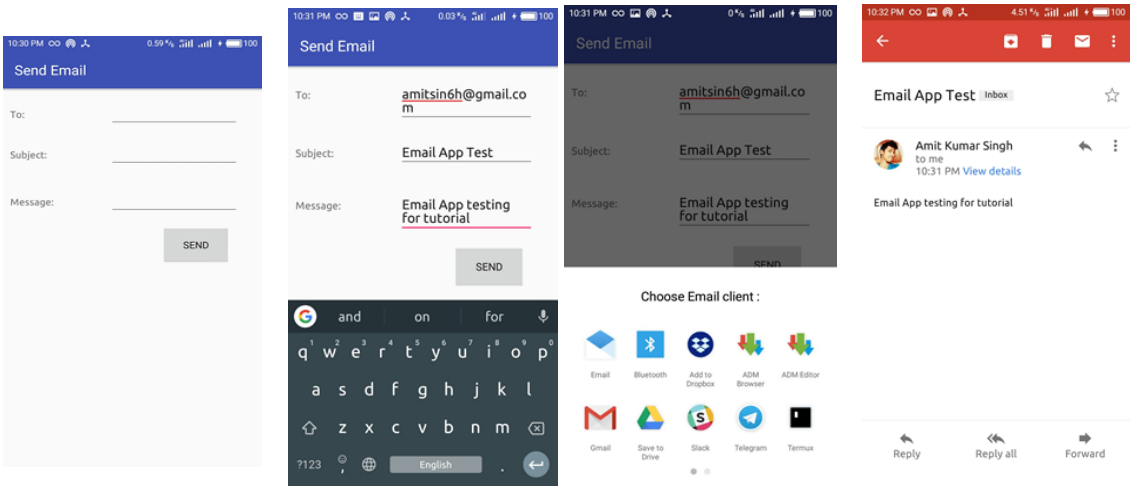
```
}
```

If anyone faces a problem in understanding the java code, they can comment below and I will help you guys to understand.

Step 4 - Compile and Run

Now, we are ready to compile and run our Send Email Android app. The following screen will appear once our app gets installed.

Now, let us write and test the email and press the Send button to check whether it works or not. So, once we click the Send button, it will ask to choose the email client and then choose your email client and after sending the email check your inbox.



12. 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".

Create sample application with login module.(Check username and password) On successful login, Chnage TextView "Login Sucessful". And on failing login, alert user using Toast "Login fail"

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"
   android:layout_height="match_parent"
   >
   <TextView
      android:id="@+id/tvName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="21dp"
      android:layout_marginTop="49dp"
      android:text="User Name"
      android:textSize="18sp" />

   <EditText
      android:id="@+id/etUsername"
      android:layout_width="wrap_content"
```

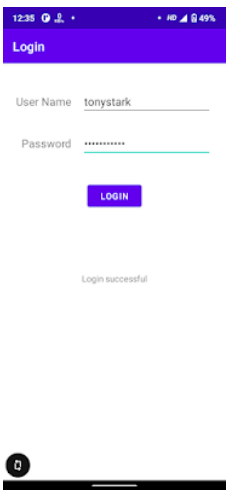
```
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/tvName"
        android:layout_alignBottom="@+id/tvName"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="23dp"
        android:ems="10"
        android:inputType="textPersonName" />

    <TextView
        android:id="@+id/tvPass"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/tvName"
        android:layout_below="@+id/etUsername"
        android:layout_marginTop="32dp"
        android:text="Password"
        android:textSize="18sp" />

    <EditText
        android:id="@+id/etPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/tvPass"
        android:layout_alignBottom="@+id/tvPass"
        android:layout_alignStart="@+id/etUsername"
        android:ems="10"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/etPassword"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="38dp"
        android:text="LOGIN"
        />
    <TextView
        android:id="@+id/tvLoginStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100sp"
        />
</RelativeLayout>
```

MainActivity.java

```
package com.example.helloworld;

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;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText etUsername, etPassword;
    Button btnStatus;
    TextView tvLoginStatus;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etUsername = (EditText) findViewById(R.id.etUsername);
        etPassword = (EditText) findViewById(R.id.etPassword);
        btnStatus = (Button) findViewById(R.id.button);
        tvLoginStatus = (TextView) findViewById(R.id.tvLoginStatus);

        btnStatus.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                check();
            }
        });
    }

    public void check(){
        if(etUsername.getText().toString().equals("tonystark") &&
etPassword.getText().toString().equals("loveyou3000")){
            tvLoginStatus.setText("Login successful");
        }else{
            Toast.makeText(this, "Login fail", Toast.LENGTH_LONG).show();
        }
    }
}
```

Output



13.Display Map based on the Current/given location.

Android Google Map Displaying Current Location

In the previous tutorial of Android Google Map, we simply displayed the default coordinates (location) set by the *MapsActivity.java* class file.

Callback methods in Google Map

1. **OnMapRreadyCallback:** This callback interface invokes when it instance is set on MapFragment object. The onMapReady(GoogleMap) method of OnMapReadyCallback interface is called when the map is ready to used. In the onMapReady(GoogleMap) method we can add markers, listeners and other attributes.
2. **LocationListener:** This interface is used to receive notification when the device location has changed. The abstract method of LocationListener onLocationChanged(Location) is called when the location has changed.
3. **GoogleApiClient.ConnectionCallbacks:** This interface provide callbacks methods onConnected(Bundle) and onConnectionSuspended(int) which are called when the device is to connected and disconnected.
4. **GoogleApiClient.OnConnectionFailedListener:** This interface provide callbacks method onConnectionFailed(ConnectionResult) which is called when there was an error in connecting the device to the service.

The **setMyLocationEnabled()** method of GoogleMap is used to enable location layer, which allows device to interact with current location.

activity_maps.xml

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.com.mapexample.MapsActivity" />
```

build.gradel

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.google.android.gms:play-services-maps:11.8.0'
    compile 'com.google.android.gms:play-services-location:11.8.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
```

```
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'

}
```

MapsActivity.java

```
package example.com.mapexample;
import android.os.Build;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.common.api.GoogleApiClient;
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.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.location.LocationServices;
 import android.location.Location;
import android.Manifest;
import android.content.pm.PackageManager;
import android.support.v4.content.ContextCompat;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
        LocationListener,GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener{
    private GoogleMap mMap;
   Location mLastLocation;
   Marker mCurrLocationMarker;
   GoogleApiClient mGoogleApiClient;
   LocationRequest mLocationRequest;
    @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_maps);
       // Obtain the SupportMapFragment and get notified when the map is ready to be used.
```

```
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                buildGoogleApiClient();
                mMap.setMyLocationEnabled(true);
            }
        }
        else {
            buildGoogleApiClient();
            mMap.setMyLocationEnabled(true);
        }
    }
    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API).build();
        mGoogleApiClient.connect();
    }
    @Override
    public void onConnected(Bundle bundle) {
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);
        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, t
his);
```

```
        }
    }
    @Override
    public void onConnectionSuspended(int i) {
    }
    @Override
    public void onLocationChanged(Location location) {

        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }
        //Place current location marker
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Position");
        markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
        mCurrLocationMarker = mMap.addMarker(markerOptions);

        //move map camera
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }
    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {
    }
}
```

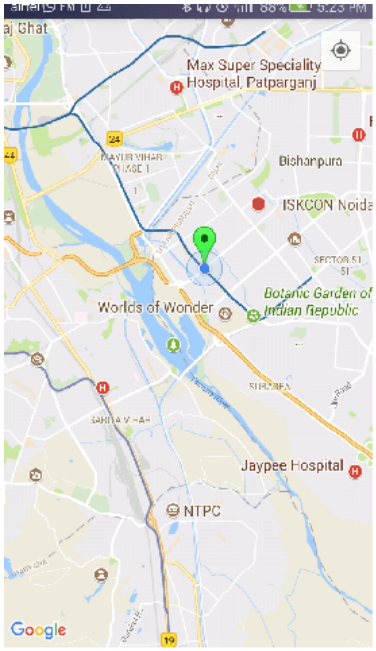
Required Permission in AndroidManifest.xml

```
<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" />
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.com.mapexample">
     <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:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <meta-data
           android:name="com.google.android.geo.API_KEY"
           android:value="@string/google_maps_key" />
         <activity
           android:name=".MapsActivity"
           android:label="@string/title_activity_maps">
           <intent-filter>
              <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
        </activity>
    </application>
</manifest>
```

Output



14.Learn to deploy Android applications

1. Perform manual testing

Before you even think about deploying your application, you need to make sure that the version is stable enough. To do this, you can run it on a smartphone and try it out to see if all features work properly.

2. Run all your automated tests

Testing your application manually isn't really an easy task, and it's not even very efficient, since it leaves room for errors. To overcome this, make sure you have automatic tests for your application. You don't have to cover all your features with unit tests, but at the very least, make sure you've tested the main features, as well as those most likely to crash. Then, before deployment, run all your tests.

3. Enable shrinking and obfuscation

To make your app smaller and more efficient, you should enable shrinking in your release build. This will remove unnecessary code and resources, making your app lighter. Additionally, enabling shrinking also includes obfuscation, which shortens the names of your app's classes and members, and optimization, which uses smart strategies to further reduce the app's size. In essence, it's like tidying up your app's code, giving it shorter names, and making it as compact as possible. This operation is achieved thanks to R8.

```
android {
   buildTypes {
      getByName("release") {
         isMinifyEnabled = true
         isShrinkResources = true
         proguardFiles(
            getDefaultProguardFile("proguard-android-optimize.txt"),
            "proguard-rules.pro"
         )
      }
   }
   ...
}
```

- isMinifyEnbled = true enables code shrinking, obfuscation, and optimization for the project's release build type.
- isShrinkResources = true enables resource shrinking, which is performed by the android Gradle plugin.
- The proguard file includes the default ProGuard rules files, which can help us to customize how our code will be shrinked.

4. Run a release build on your phone

All these optimizations related to shrinking and obfuscation may create an unexpected behavior, for example If your code relies on reflection. Since obfuscated names are different from the original ones, it may break the functionality of these parts of your code.

In order to see stack trace, you can temporarily set isDebuggable=true for your release build. Don't forget to remove it after.

```
android {
   buildTypes {
      getByName("release") {
         isDebuggable = true
         isMinifyEnabled = true
         isShrinkResources = true
         proguardFiles(
            getDefaultProguardFile("proguard-android-optimize.txt"),
            "proguard-rules.pro"
```

```
        )
      }
    }
    ...
}
```

Code processed by R8 is changed in various ways this can result in stack traces that don't directly match your source code, making it harder to pinpoint the exact location of an issue. Additionally, if debugging information is not preserved during obfuscation, line numbers may change, further complicating the debugging process.

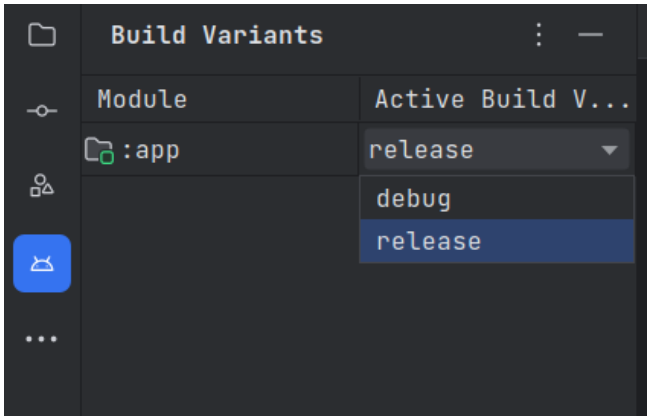
To recover the original stack trace, R8 provides the retrace command-line tool, which is bundled with the command-line tools package.

To support retracing of your application's stack traces, you should ensure the build retains sufficient information to retrace with by adding the following rules to your module's proguard-rules.pro file:

```
-keepattributes LineNumberTable,SourceFile
-renamesourcefileattribute SourceFile
```

Run a release build variant from android studio

Android Studio simplifies the process of selecting a build variant, with each variant corresponding to a specific buildType. To run the release build configuration, you'll need to switch to the "release" build variant.



You'll also need to include a signing configuration using your developer key.

```
...
android {
    ...
    defaultConfig {...}
    signingConfigs {
        // create a signingConfig for release
        create("release") {
            storeFile = file("myreleasekey.keystore")
            storePassword = "password"
            keyAlias = "MyReleaseKey"
            keyPassword = "password"
        }
    }
    buildTypes {
        getByName("release") {
            isMinifyEnabled = true
            isShrinkResources = true
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
            // setting the signingConfig here
            signingConfig = signingConfigs.getByName("release")
        }
    }
}
```

After that, you just have to click the Run ▶ button from Android Studio

*You must be careful not to version sensitive information linked to your developer key. You can use the **local.properties** file and put this information in it, then read it with gradle. In the same way, you can use a plugin such as secrets-gradle-plugin.*

Common issues

The error I've encountered most often is related to serialization. The solution is to use annotations such as `@SerialName` or `@SerializedName` on the properties of the class in question. Or prevent the class from being modified by R8 by using the `@Keep` annotation on top of it.

Some third-party libraries may also cause issues. To solve them, consult library documentation for guidance, and incorporate recommended Proguard or R8 rules provided by library developers. Thorough testing of the app, especially the parts that depend on third-party libraries, is crucial to identify and resolve any obfuscation-induced issues. Additionally, seeking support from the library's developers or community can offer valuable solutions for maintaining compatibility while enhancing app security through obfuscation.

5. Create an Internal release

By conducting closed alpha or beta testing with a select group of users, you can uncover and address bugs, compatibility issues, and usability concerns before releasing the app to a wider audience. This controlled testing environment not only helps you refine the app's performance and security but also provides valuable feedback that can lead to user-centric improvements. Furthermore, it allows for a more strategic and controlled app rollout, reducing the risk of negative reviews and ensuring a smoother and more successful launch when you make the app available to the public.

6. Finally, the public release 🚀

This is a moment of both excitement and opportunity, but it also comes with important responsibilities. You'll need to prepare a compelling app listing, complete with engaging visuals, a clear description, and any necessary marketing materials to catch the attention of potential users. It's vital to consider user feedback and continuously monitor reviews and ratings after the public release, as these insights can lead to ongoing improvements.