Practical 1: Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals: Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple "Hello World" program.

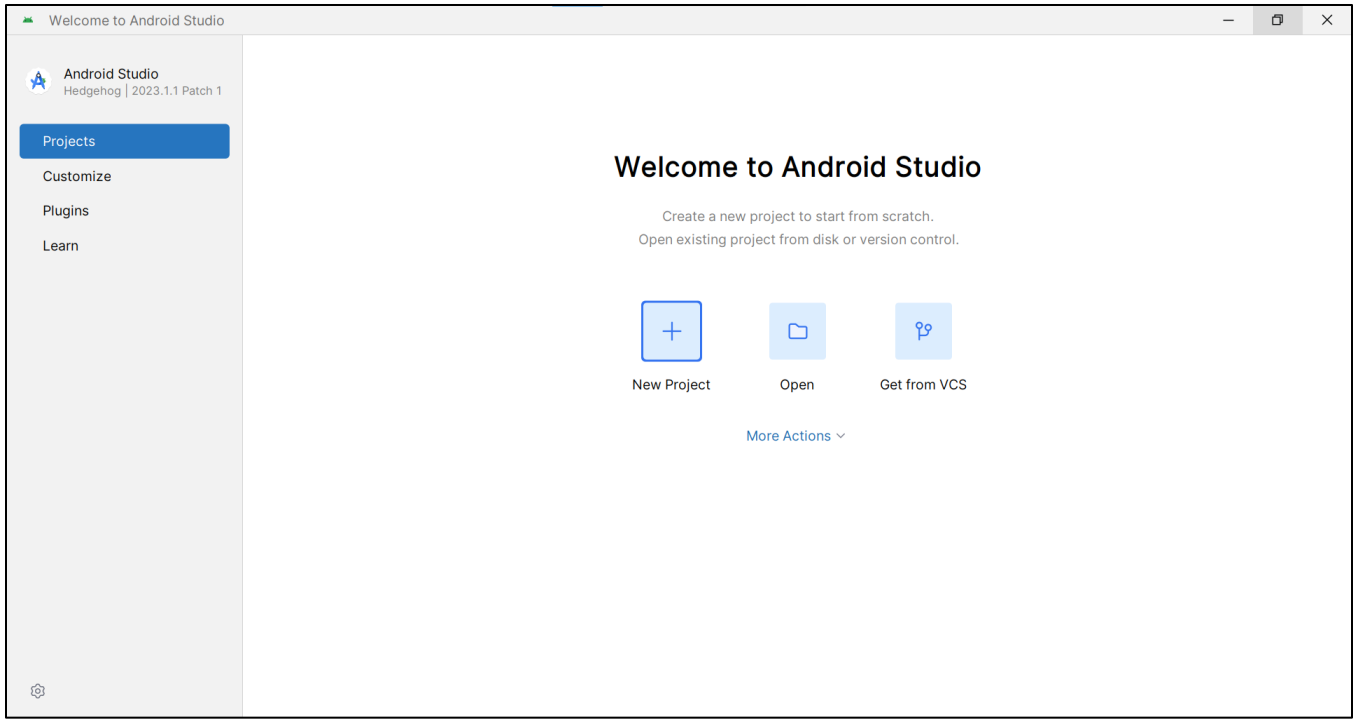
Create a project using the template:

In this codelab, you create an Android app with the **Empty Activity** project template provided by Android Studio.

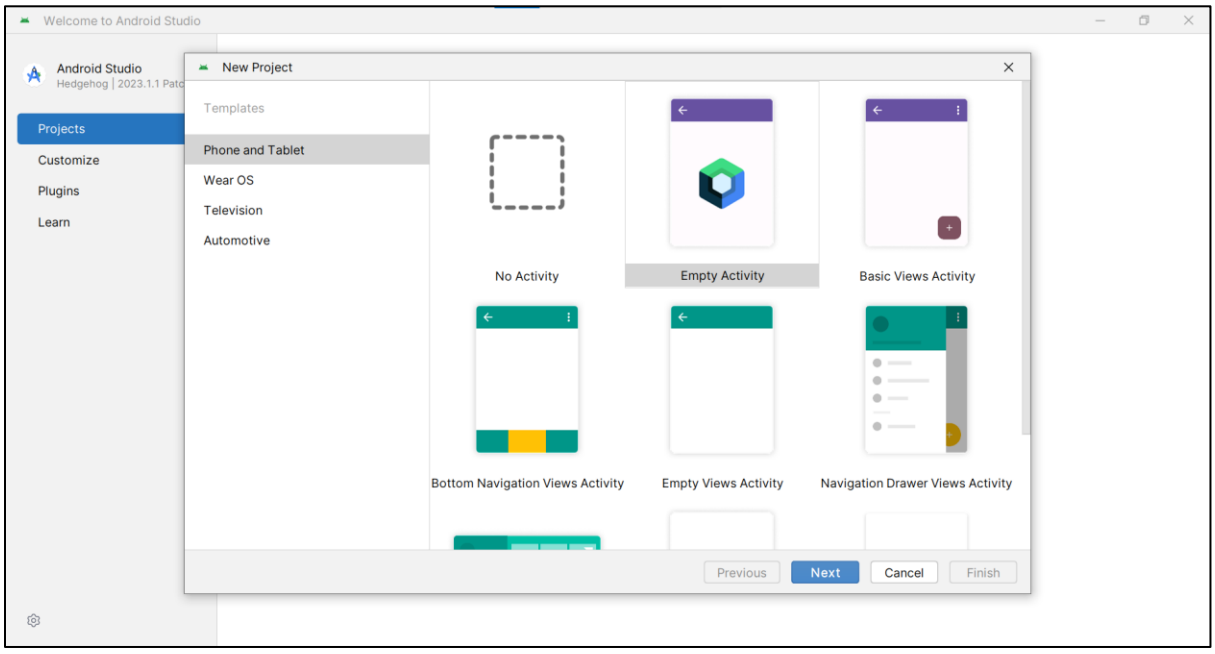
To create a project in Android Studio:

1. Double click the Android Studio icon to launch Android Studio.

2. In the **Welcome to Android Studio** dialog, click **New Project**.



The **New Project** window opens with a list of templates provided by Android Studio.



In Android Studio, a project template is an Android project that provides the blueprint for a certain type of app. Templates create the structure of the project and the files needed for Android Studio to build your project. The template that you choose provides starter code to get you going faster.

3. Make sure the **Phone and Tablet** tab is selected.

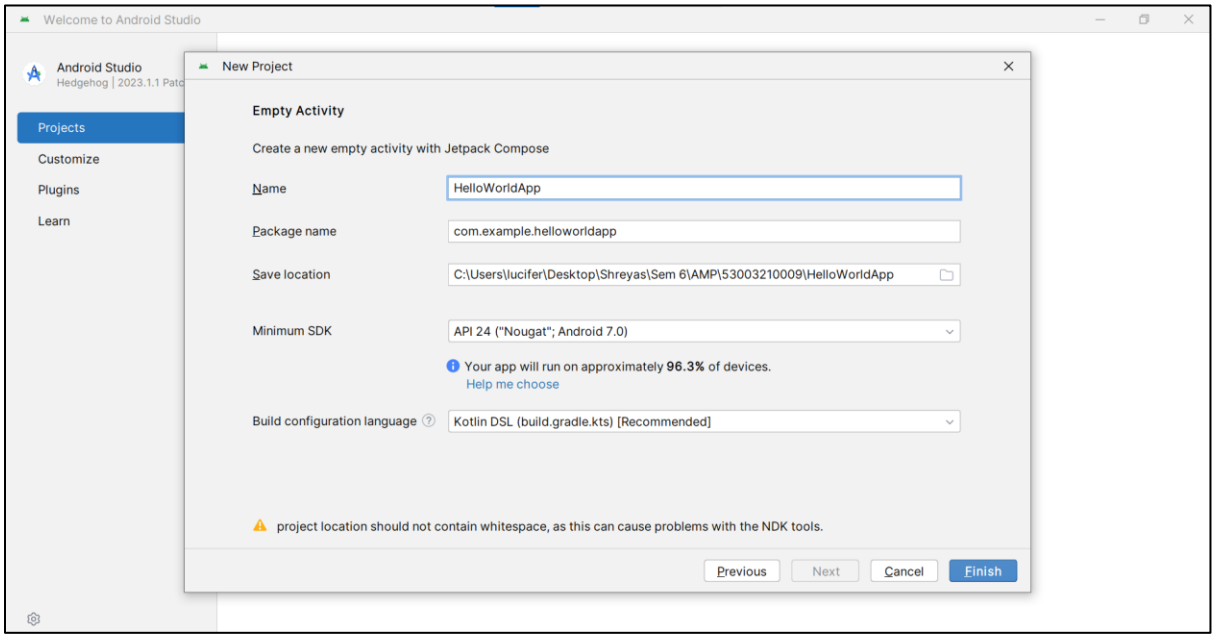
4. Click the **Empty Activity** template to select it as the template for your project. The **Empty Activity** template is the template to create a simple project that you can use to build a Compose app. It has a single screen and displays the text "Hello Android!".
5. Click **Next**. The **New Project** dialog opens. This has some fields to configure your project.
6. Configure your project as follows:

The **Name** field is used to enter the name of your project, for this codelab type "Greeting Card".

Leave the **Package name** field as is. This is how your files will be organized in the file structure. In this case, the package name will be com.example.greetingcard.

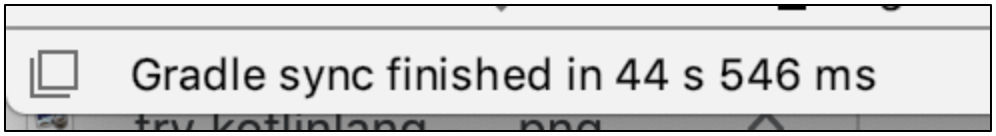
Leave the **Save location** field as is. It contains the location where all the files related to your project are saved. Take a note of where that is on your computer so that you can find your files.

Select **API 24: Android 7.0 (Nougat)** from the menu in the **Minimum SDK** field. Minimum SDK indicates the minimum version of Android that your app can run on.



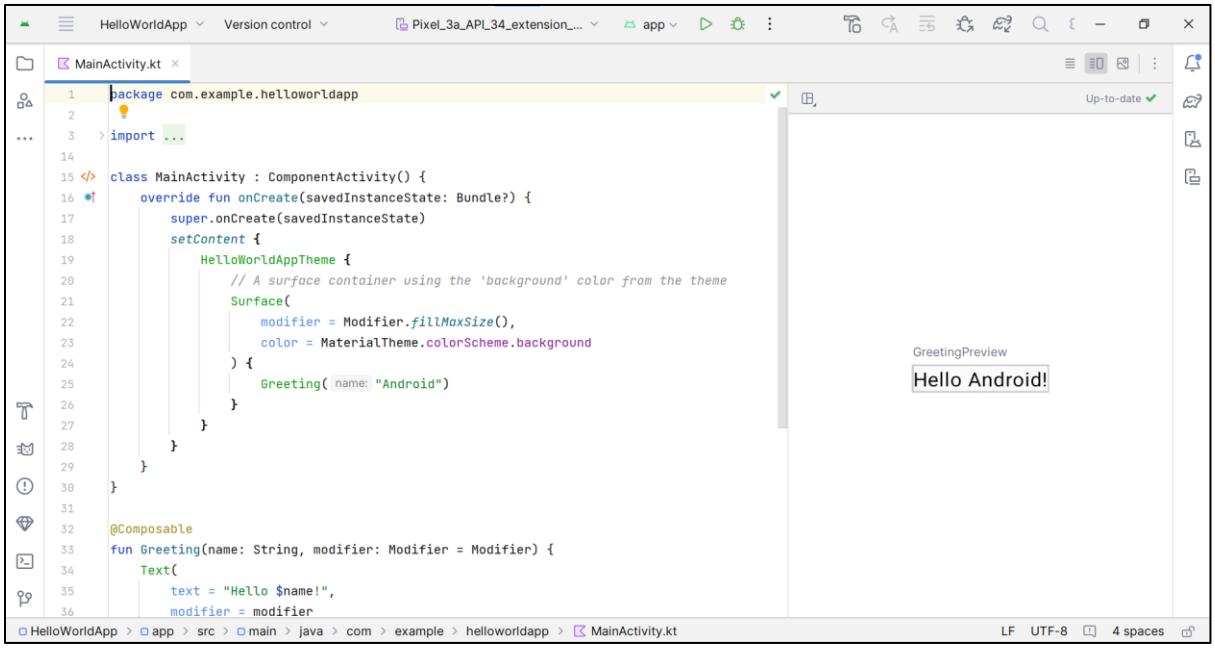
7. Click **Finish**. This may take a while - this is a great time to get a cup of tea! While Android Studio is setting up, a progress bar and message indicates whether Android Studio is still setting up your project. It may look like this:

A message that looks similar to this informs you when the project set up is created.



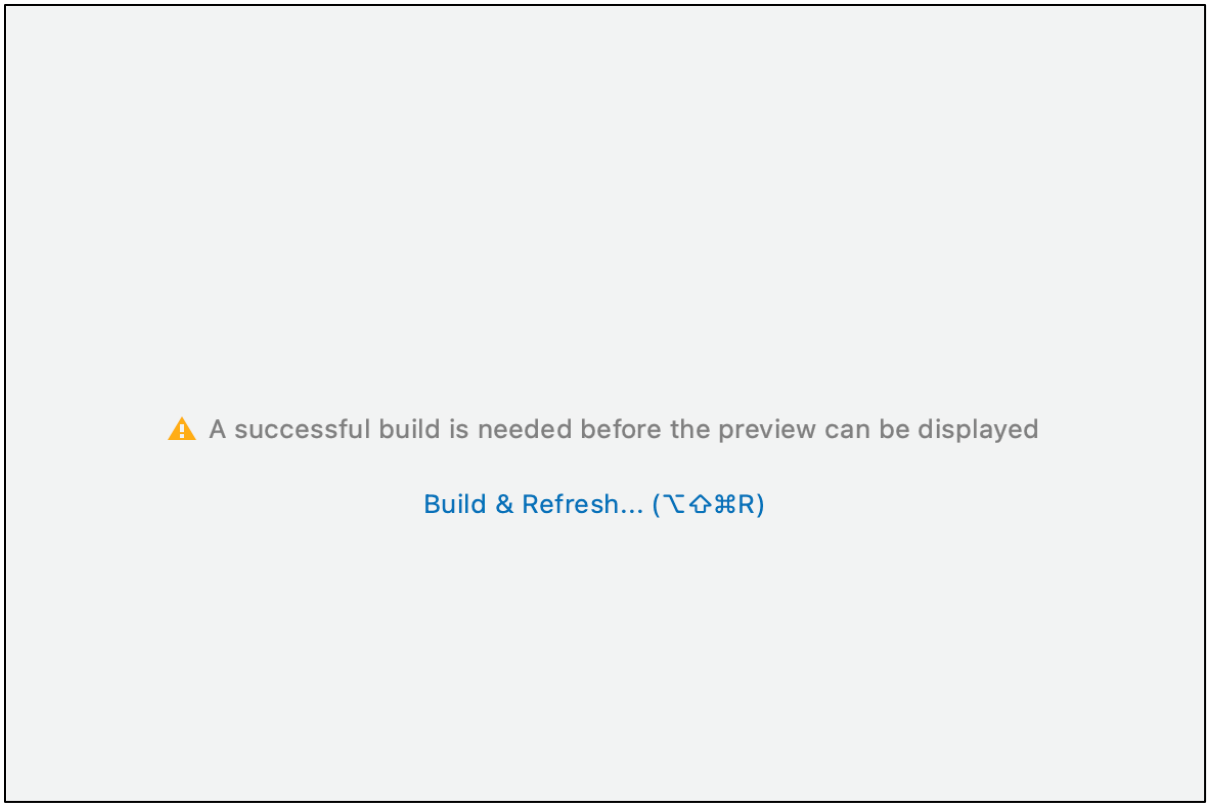
9. Click **Split** on the top right of Android Studio, this allows you to view both code and design. You can also click **Code** to view code only or click **Design** to view design only.

After pressing **Split** you should see three areas:

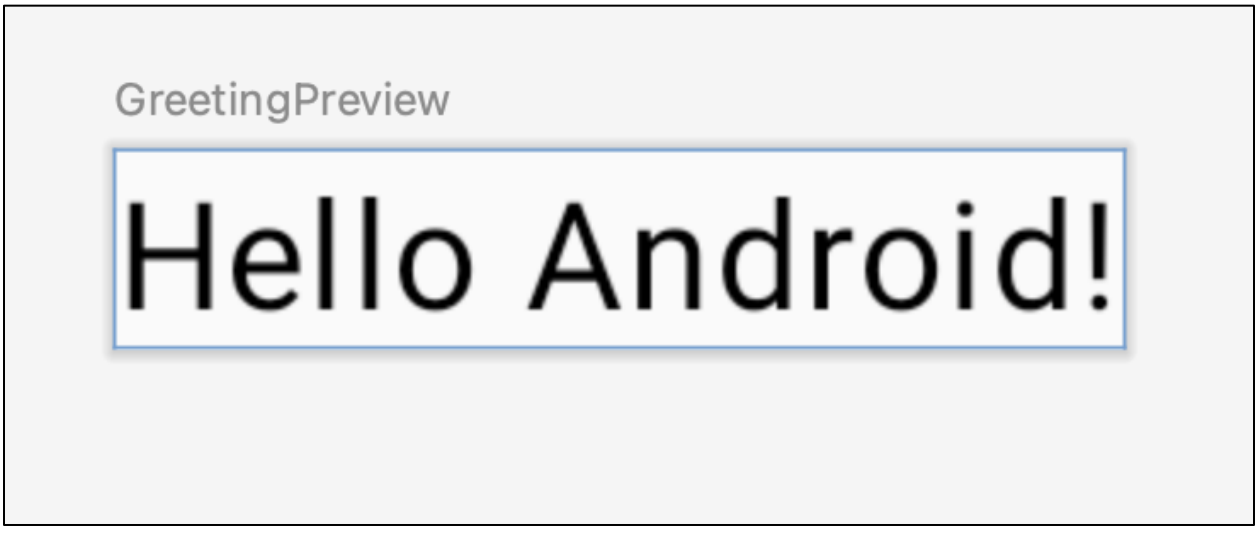


- The **Project** view (1) shows the files and folders of your project
- The **Code** view (2) is where you edit code
- The **Design** view (3) is where you preview what your app looks like

In the **Design** view, you will see a blank pane with this text:



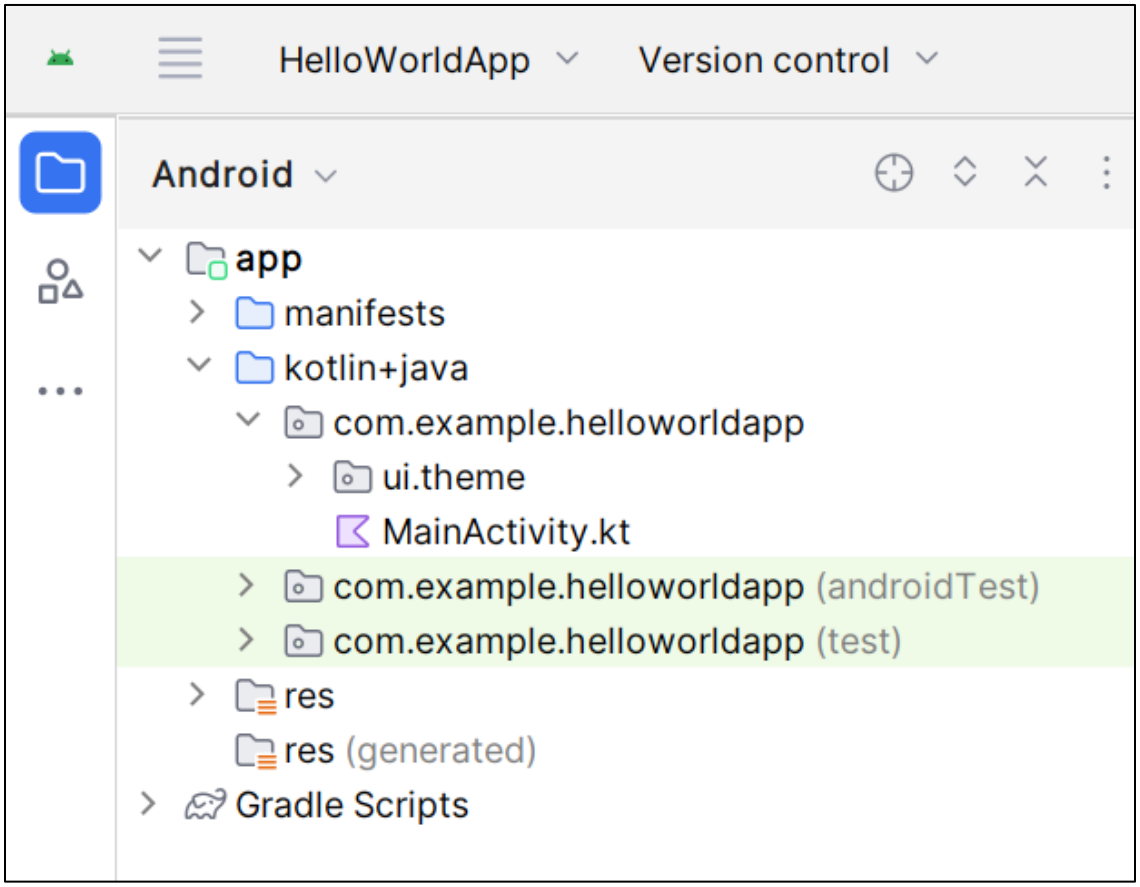
10. Click **Build & Refresh**. It may take a while to build but when it is done the preview shows a text box that says "**Hello Android!**". Empty Compose activity contains all the code necessary to create this app.



3. Find project files

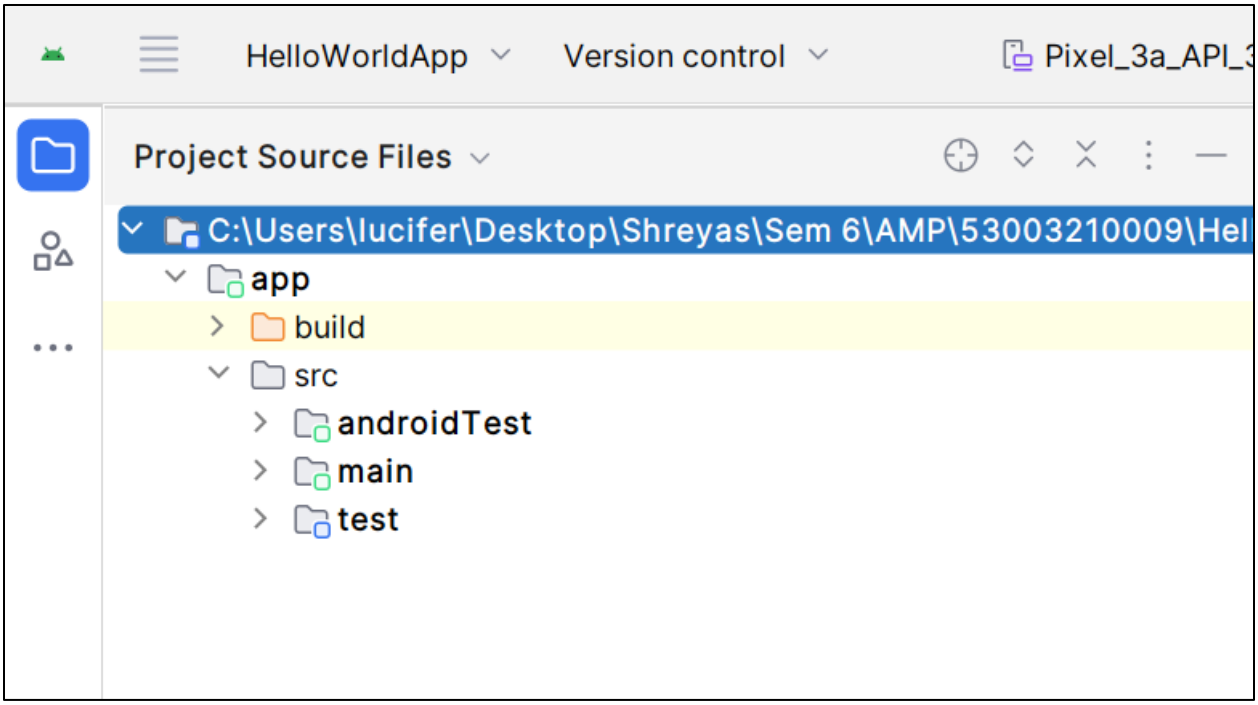
In this section you will continue to explore Android Studio by becoming familiar with the file structure.

1. In Android Studio, take a look at the **Project** tab. The **Project** tab shows the files and folders of your project. When you were setting up your project the package name was **com.example.greetingcard**. You can see that package right here in the **Project** tab. A package is basically a folder where code is located. Android Studio organizes the project in a directory structure made up of set of packages.
2. If necessary, select **Android** from the drop-down menu in the **Project** tab.



This is the standard view and organization of files that you use. It's useful when you write code for your project because you can easily access the files you will be working on in your app. However, if you look at the files in a file browser, such as Finder or Windows Explorer, the file hierarchy is organized very differently.

3. Select **Project Source Files** from the drop-down menu. You can now browse the files in the same way as in any file browser.



4. Select **Android** again to switch back to the previous view. You use the **Android** view for this course. If your file structure ever looks strange, check to make sure you're still in **Android** view.

4. Update the text

Now that you have gotten to know Android Studio, it's time to start making your greeting card!

Look at the **Code** view of the MainActivity.kt file. Notice there are some automatically generated functions in this code, specifically the onCreate() and the setContent() functions.

```
class MainActivity :ComponentActivity()
{
override fun onCreate(savedInstanceState:Bundle?)
{
super.onCreate(savedInstanceState)
  setContent {
     GreetingCardTheme {
       // A surface container using the 'background' color from the theme
       Surface(
              modifier = Modifier.fillMaxSize(),
              color = MaterialTheme.colors.background
                ){
              Greeting("Android")

               }}}}}
```

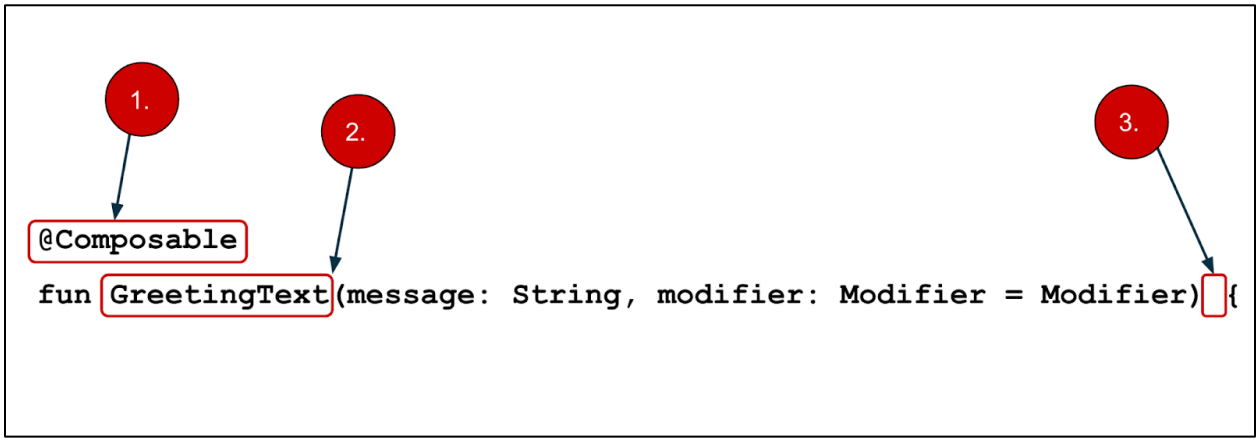
The onCreate() function is the entry point to this Android app and calls other functions to build the user interface. In Kotlin programs, the main() function is the entry point/starting point of execution. In Android apps, the onCreate() function fills that role.

The setContent() function within the onCreate() function is used to define your layout through composable functions. All functions marked with the @Composable annotation can be called from the setContent() function or from other Composable functions. The annotation tells the Kotlin compiler that this function is used by Jetpack Compose to generate the UI.

Next, look at the Greeting() function. The Greeting() function is a Composable function, notice the @Composable annotation above it. This Composable function takes some input and generates what's shown on the screen.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(text = "Hello $name!")
}
```

You've learned about functions before, but there are a few differences with composable functions.



- You add the @Composable annotation before the function.
- @Composable function names are capitalized.
- @Composable functions can't return anything.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(text = "Hello $name!")
}
```

Right now the Greeting() function takes in a name and displays Hello to that person.

1. Update the Greeting() function to introduce yourself instead of saying "Hello":

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(text = "Hello $name!")}
```

2. Android should automatically update the preview.
3. Preview.

Great! You changed the text, but it introduces you as Android, which is probably not your name. Next, you will personalize it to introduce you with your name!

The GreetingPreview() function is a cool feature that lets you see what your composable looks like without having to build your entire app. To enable a preview of a composable, annotate it with @Composable and @Preview. The @Preview annotation tells Android Studio that this composable should be shown in the design view of this file.

As you can see, the @Preview annotation takes in a parameter called showBackground. If showBackground is set to **true**, it will add a background to your composable preview.

Since Android Studio by default uses a light theme for the editor, it can be hard to see the difference between showBackground = true and showBackground = false. However, this is an example of what the difference looks like. Notice the white background on the image set to true.

showBackground = true

showBackground = false

3. Update the GreetingPreview() function with your name. Then rebuild and check out your personalized greeting card!

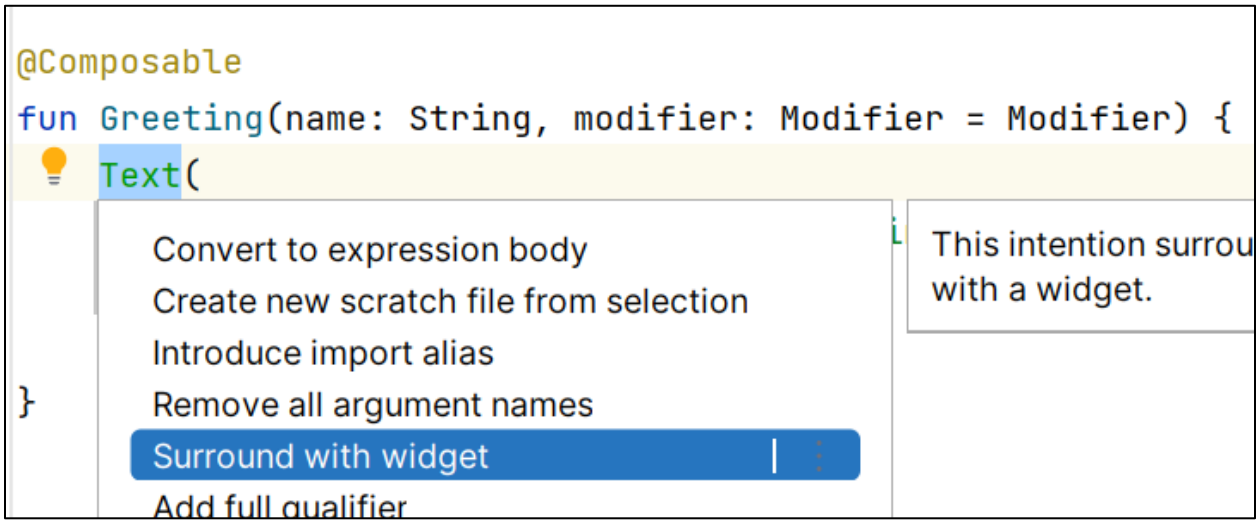
```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    GreetingCardTheme {
        Greeting("Shreyas")
   }
}
```

5. Change the background color

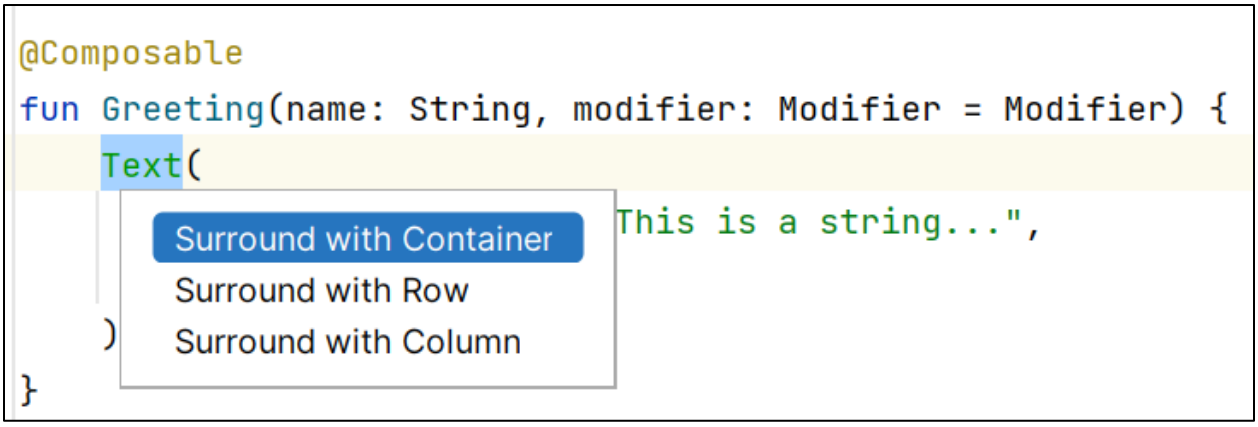
Now you have the introduction text, but it's a little boring! In this section, you learn to change the background color.

To set a different background color for your introduction, you'll need to surround your text with a Surface. A Surface is a container that represents a section of UI where you can alter the appearance, such as the background color or border.

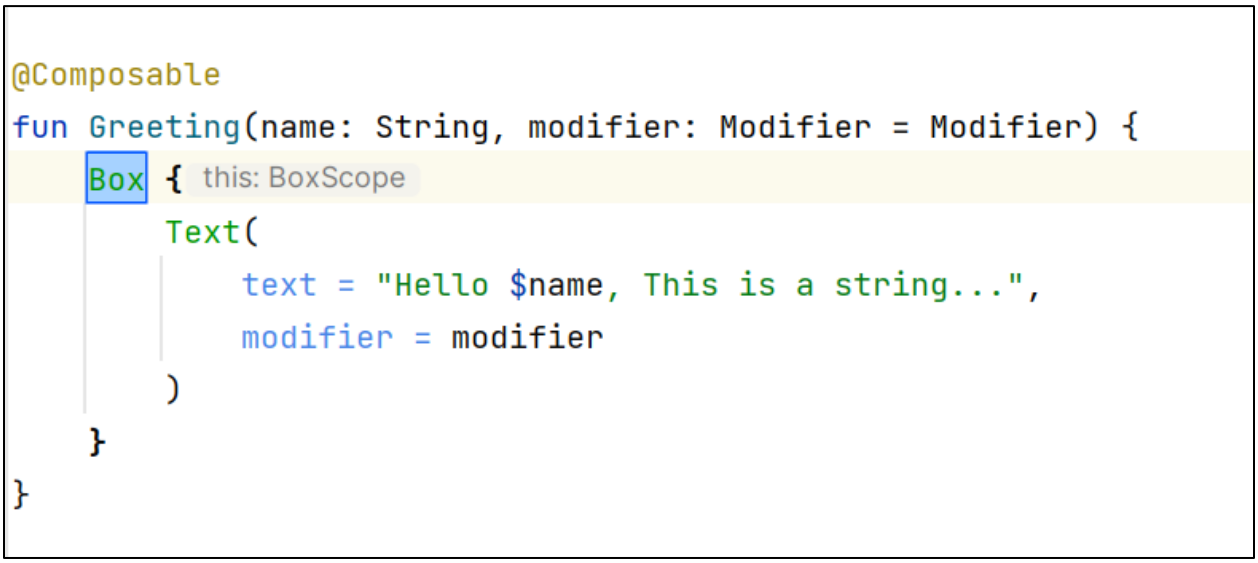
1. To surround the text with a Surface, highlight the line of text, press (Alt+Enter for Windows or Option+Enter on Mac), and then select **Surround with widget**.



2. Choose **Surround with Container**.



The default container it will give you is Box, but you can change this to another container type. You will learn about Box layout later in the course.



3. Delete Box and type Surface() instead.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
   Surface() {
       Text(
           text = "Hello $name",
           modifier = modifier
       )
   }
}
```

4. To the Surface container add a color parameter, set it to Color.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
   Surface(color = Color) {
       Text(
```

```
            text = "Hello $name",
            modifier = modifier
        )
    }
}
```

5. When you type Color you may notice that it is red, which means Android Studio is not able to resolve this. To solve this scroll to the top of the file where it says import and press the three buttons.

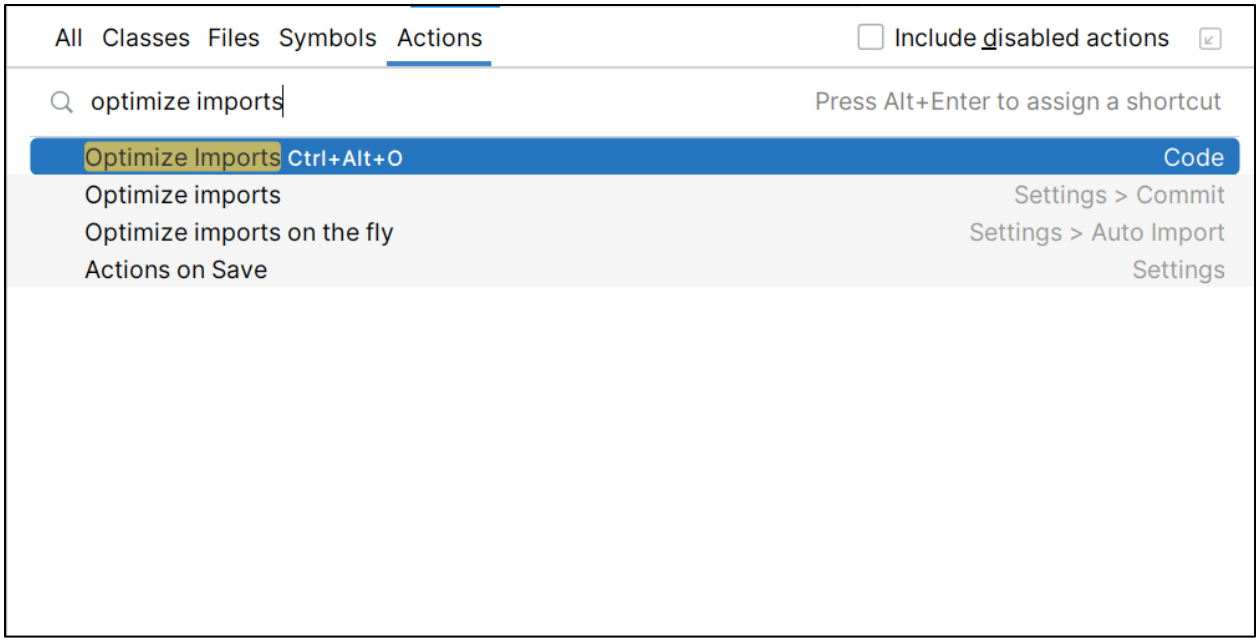
6. Add this statement to the bottom of the list of imports.

```
import androidx.compose.ui.graphics.Color
```

The full list of imports will look similar to this.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.greetingcard.ui.theme.GreetingCardTheme
import androidx.compose.ui.graphics.Color
```

7. In your code, the best practice is to keep your imports listed alphabetically and remove unused imports. To do this press **Help** on the top toolbar, type in **optimize imports**, and click on **Optimize Imports**.



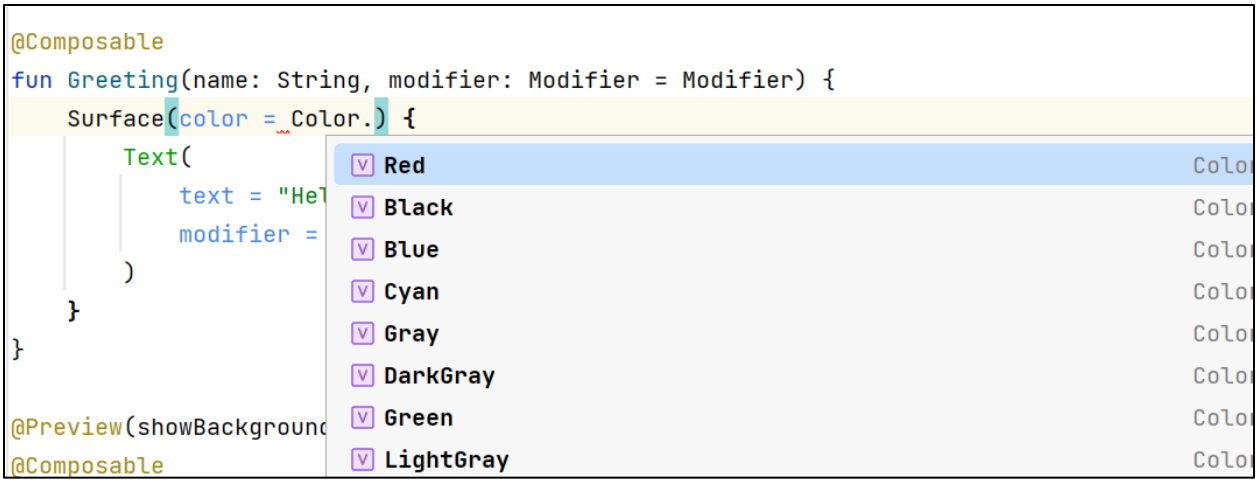
You could open the **Optimize Imports** directly from the menu: **Code** > **Optimize Imports.** Using Help's search option will help you locate a menu item if you don't remember where it is.

The full list of imports will now look like this:

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.tooling.preview.Preview
import com.example.greetingcard.ui.theme.GreetingCardTheme
```

8. Notice that the Color that you typed in the Surface parentheses has switched from being red to being underlined in red. To fix that, add a period after it. You will see a pop-up showing different color options.

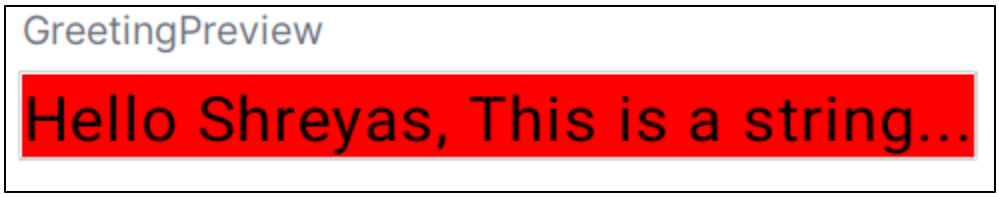
This is one of the cool features in Android Studio, it is intelligent and will help you out when it can. In this case it knows you are wanting to specify a color so it will suggest different colors.



9. Choose a color for your surface. This codelab uses **Red**, but you can choose your favorite!

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
   Surface(color = Color.Cyan) {
       Text(
           text = "Hello $name, This is a string... ",
           modifier = modifier
       )
   }
}
```

10. Notice the updated preview.



6. Add padding

Now your text has a background color, next you will add some space (padding) around the text.

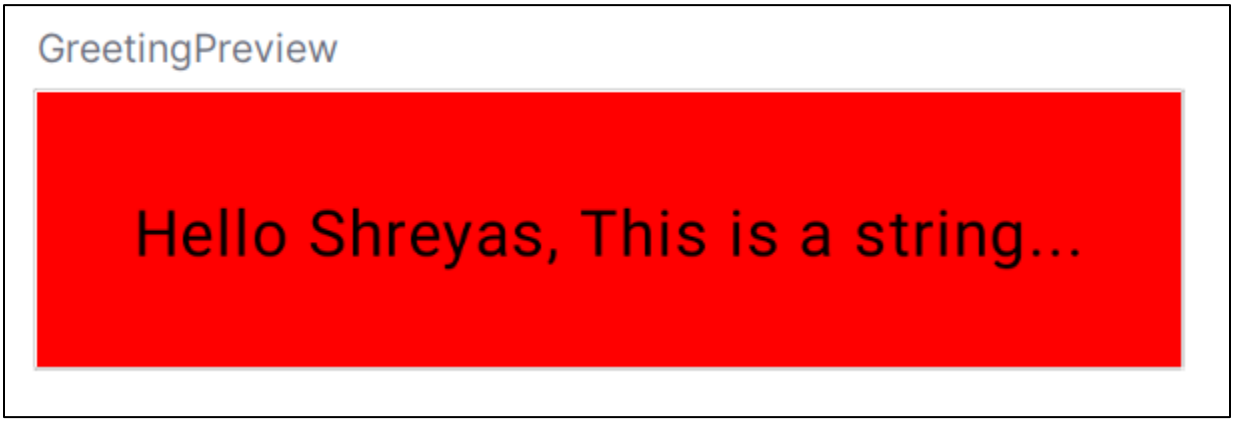
A Modifier is used to augment or decorate a composable. One modifier you can use is the padding modifier, which adds space around the element (in this case, adding space around the text). This is accomplished by using the Modifier.padding() function.

Every composable should have an optional parameter of the type Modifier. This should be the first optional parameter.

1. Add a padding to the modifier with a size of 24.dp.

```
@Composable
```

```
fun Greeting(name: String, modifier: Modifier = Modifier) {
   Surface(color = Color.Cyan) {
       Text(
           text = " Hello $name, This is a string... ",
           modifier = modifier.padding(24.dp)
       )
   }
}
```



2. Add these imports to the import statement section.

Make sure to use **Optimize Imports** to alphabetize the new imports.

```
import androidx.compose.ui.unit.dp
import androidx.compose.foundation.layout.padding
```

Review the solution code

Code snippet for review

```
package com.example.helloworldapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.helloworldapp.ui.theme.HelloWorldAppTheme
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContent {
            HelloWorldAppTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Greeting("Shreyas")
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface(color = Color.Red) {
        Text(
            text = "Hello $name, This is a string...",
            modifier = modifier.padding(24.dp)
        )
    }
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HelloWorldAppTheme {
        Greeting("Shreyas")
    }
}
```

Practical 2 : Programming Resources Android Resources: (Color, Theme, String, Drawable, Dimension, Image)

In addition to writing the application's code, you also manage other resources including colors, bitmaps, layout specifications, animation instructions, and static content that your code utilizes. These materials are always kept apart and organized into different subdirectories within the project's res/ directory.

Sr.No.	Directory & Resource Type
1	**anim/** XML files that define property animations. They are saved in res/anim/ folder and accessed from the **R.anim** class.
2	**color/** XML files that define a state list of colors. They are saved in res/color/ and accessed from the **R.color** class.
3	**drawable/** Image files like .png, .jpg, .gif or XML files that are compiled into bitmaps, state lists, shapes, animation drawable. They are saved in res/drawable/ and accessed from the **R.drawable** class.
4	**layout/** XML files that define a user interface layout. They are saved in res/layout/ and accessed from the **R.layout** class.
5	**menu/** XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. They are saved in res/menu/ and accessed from the **R.menu** class.
6	**raw/** Arbitrary files to save in their raw form. You need to call *Resources.openRawResource()* with the resource ID, which is *R.raw.filename* to open such raw files.
7	**values/** XML files that contain simple values, such as strings, integers, and colors. For example, here are some filename conventions for resources you can create in this directory − • arrays.xml for resource arrays, and accessed from the **R.array** class. • integers.xml for resource integers, and accessed from the **R.integer** class. • bools.xml for resource boolean, and accessed from the **R.bool** class. • colors.xml for color values, and accessed from the **R.color** class. • dimens.xml for dimension values, and accessed from the **R.dimen** class. • strings.xml for string values, and accessed from the **R.string** class. • styles.xml for styles, and accessed from the **R.style** class.
8	**xml/** Arbitrary XML files that can be read at runtime by calling *Resources.getXML()*. You can save various configuration files here which will be used at run time.

Accessing Resources in Code

When your Android application is compiled, a R class gets generated, which contains resource IDs for all the resources available in your res/ directory. You can use R class to access that resource using sub-directory and resource name or directly resource ID.

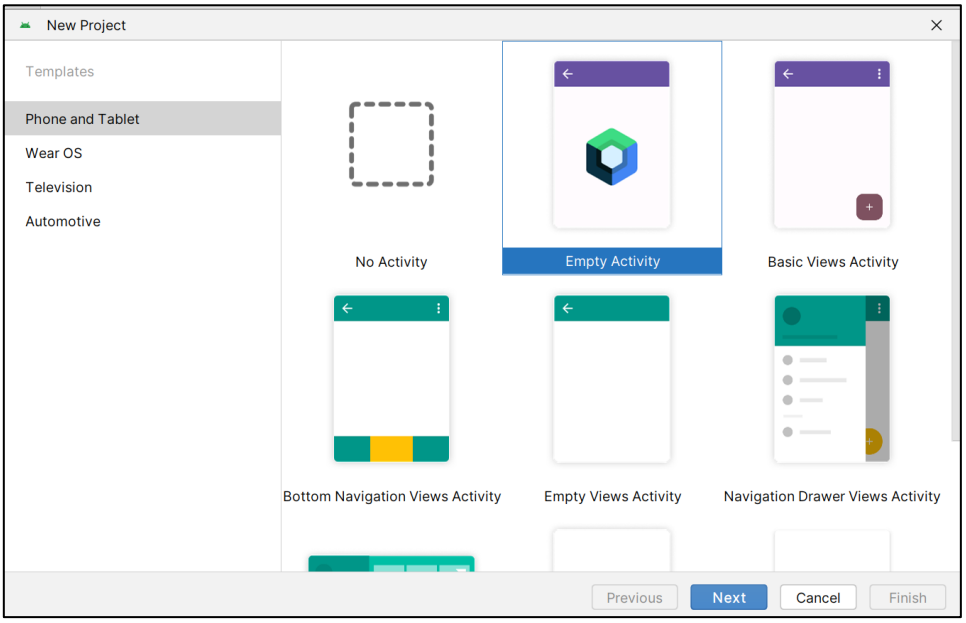
Example

To access res/drawable/myimage.png and set an ImageView you will use following code −

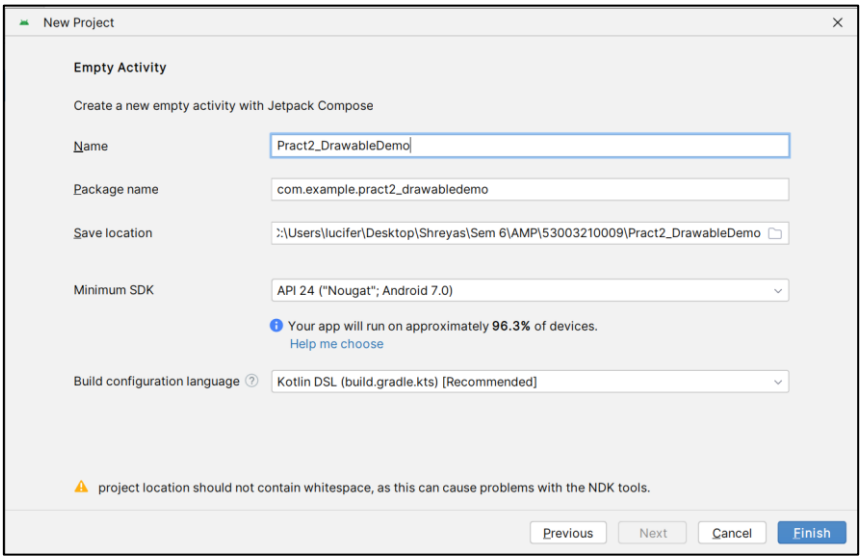
```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);

imageView.setImageResource(R.drawable.myimage);
```

Here first line of the code make use of R.id.myimageview to get ImageView defined with id myimageview in a Layout file. Second line of code makes use of R.drawable.myimage to get an image with name myimage available in drawable sub-directory under /res.

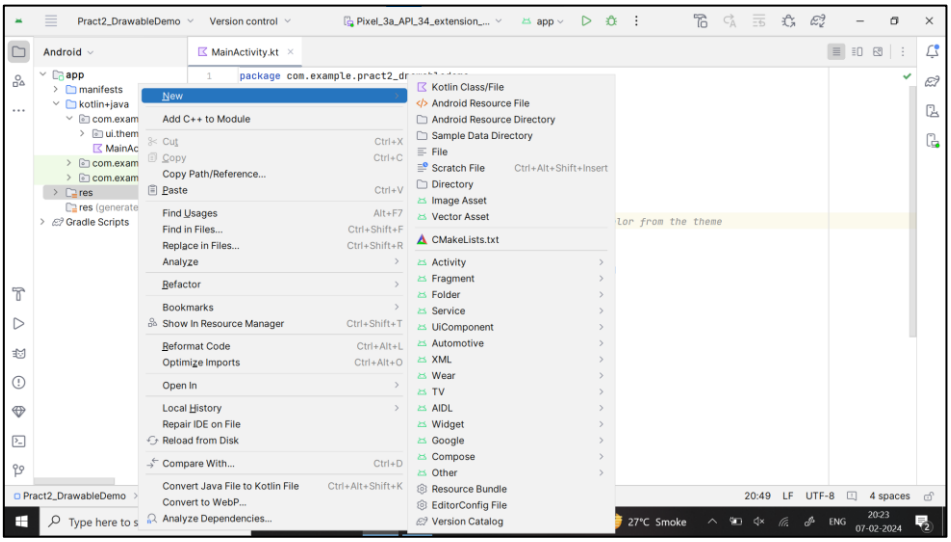
Step 1: Create a new project with empty activity.

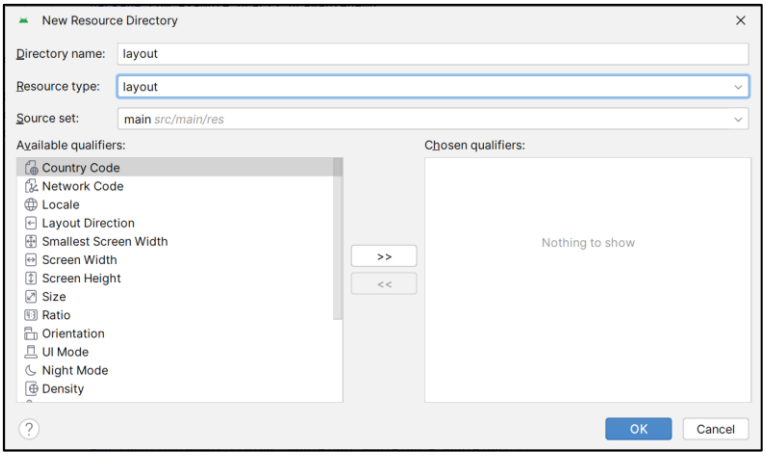


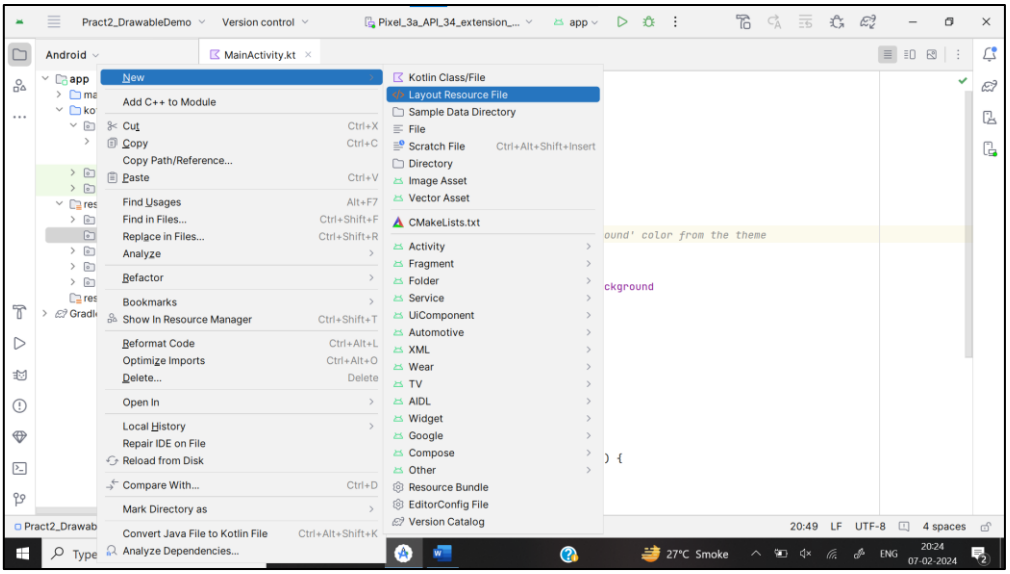
Step 2: Name the project as per your wish.

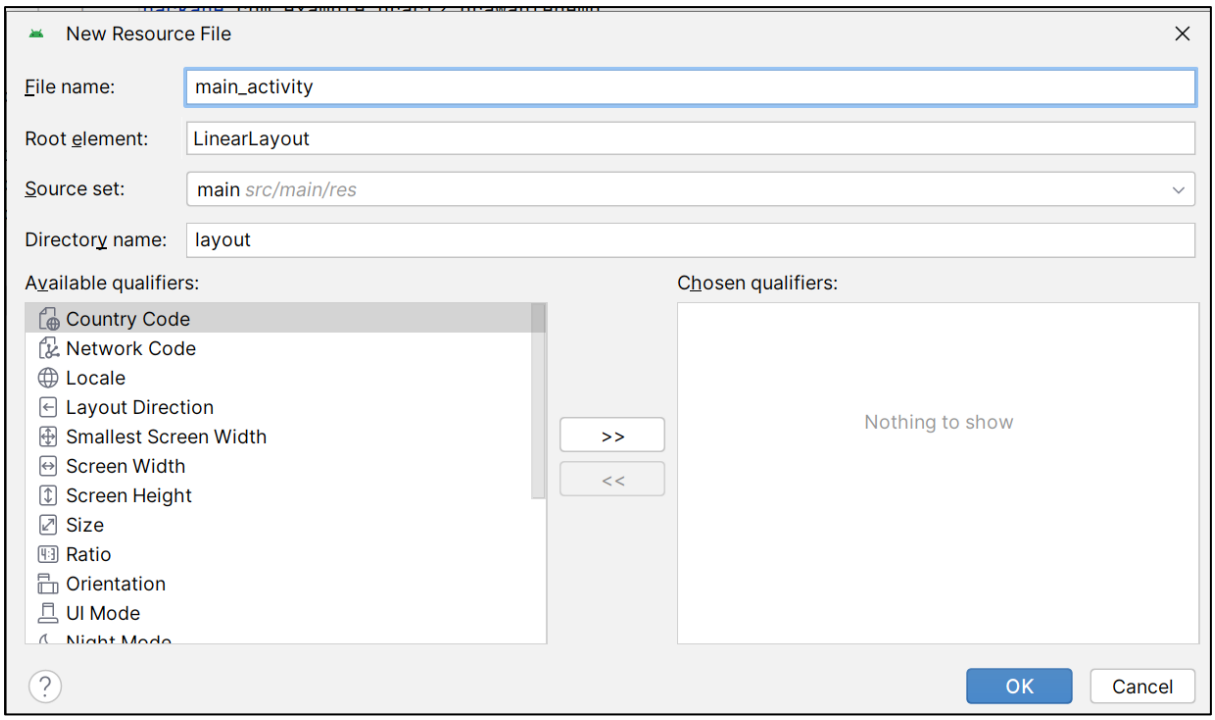


Step 3: Since we have created a project with empty activity it won't be having any xml file(this file is used to design the interface for the application) to create that we need to create layout directory first and then add the layout file.

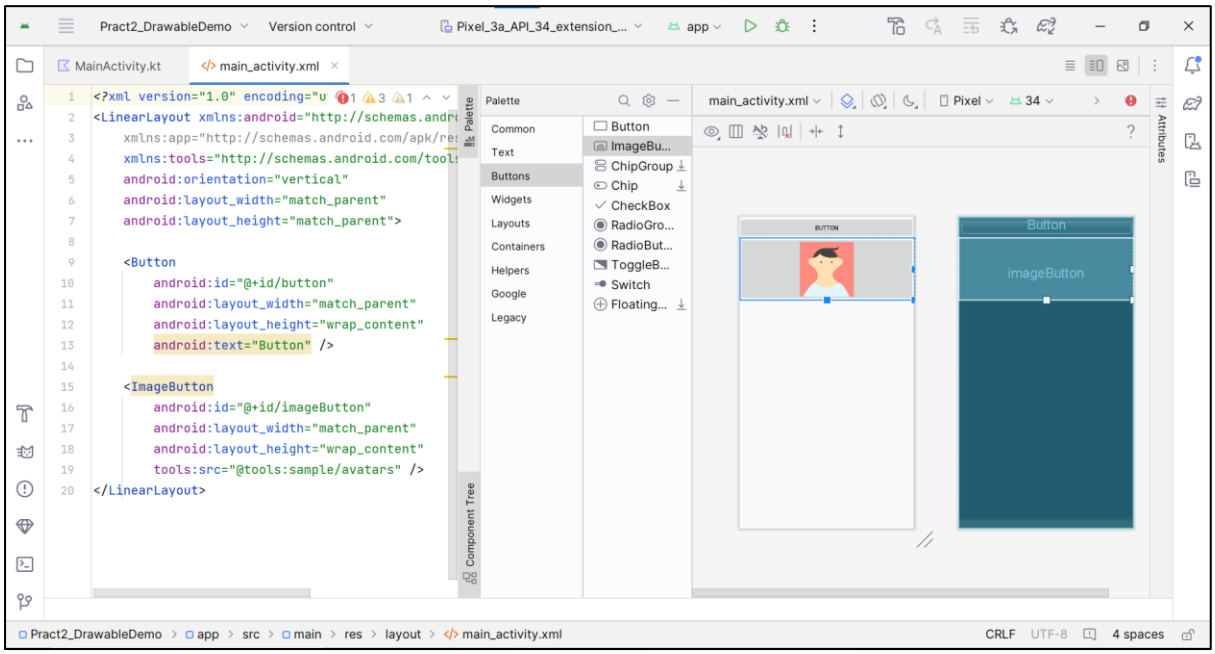








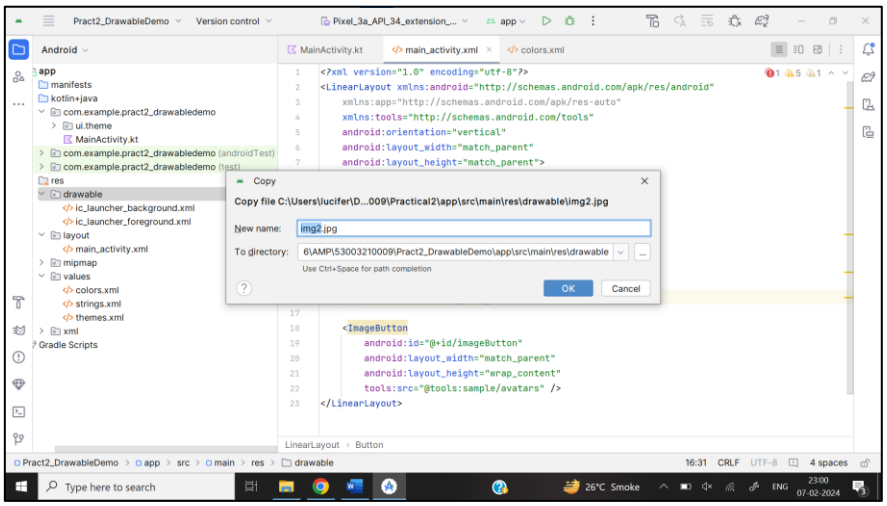
Step 4: Once the file has been created which is named as main_activity here, since it is a blank file , we are adding two components here which are button and image button.



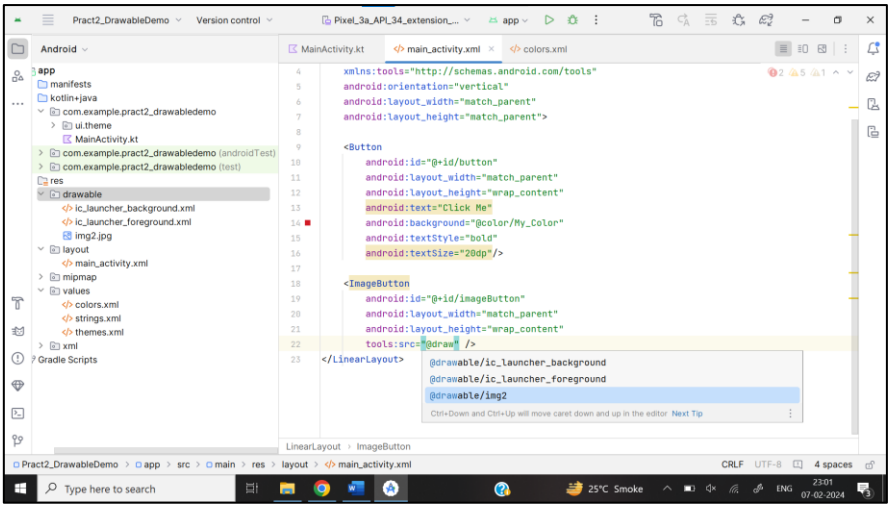
Step 5: When button is added, in inherits default setting of the component, it can be changed with following code.

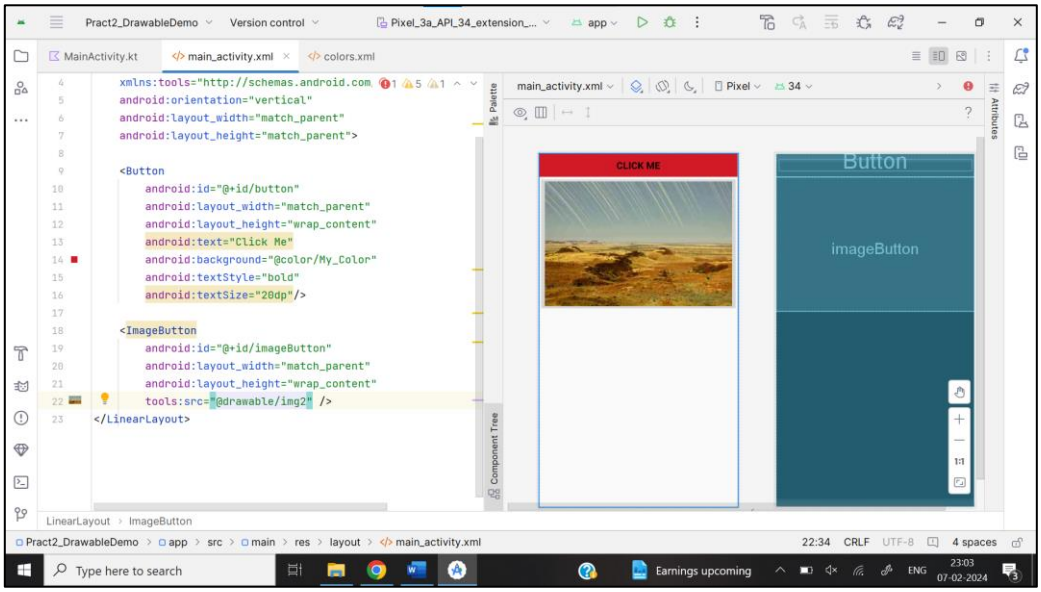
```
<Button
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Click Me!!!"
    android:background="@color/teal_200"
    android:textStyle="bold"
    android:textSize="20dp"/>
```

Step 6: To add image to the image button download an image free any free stock images website and then add it to drawable folder below res.



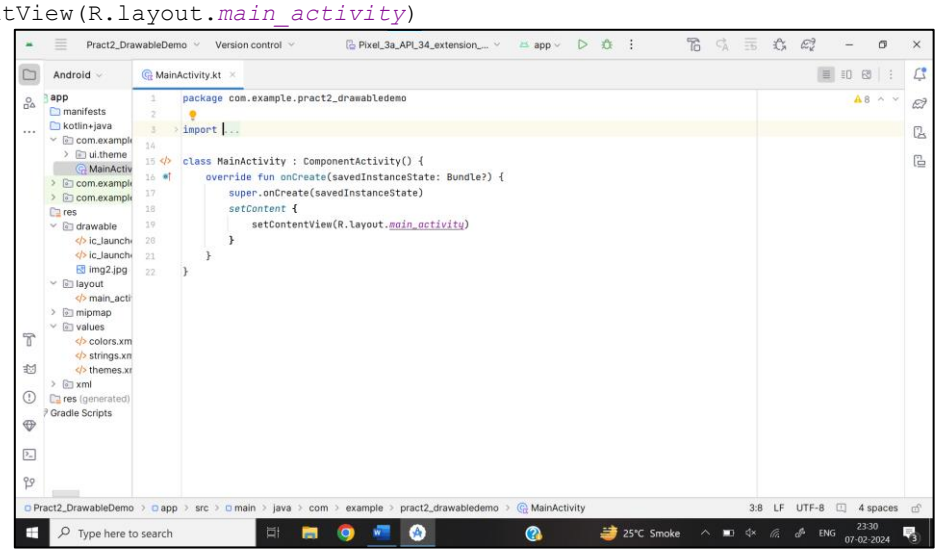
The lighting.jpg file has been added to drawable folder, so to reflect it on image button, change the src from attributes section or from the code, the file will automatically start reflecting in the options.



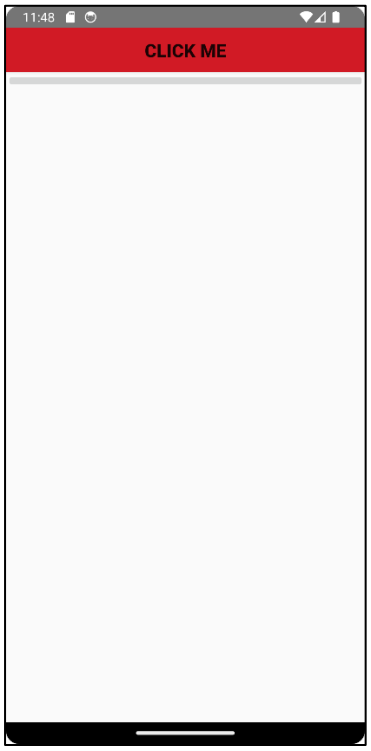


Step 7: Now this is the new file that has been added to the project, the main executable file that is main_activity.kt file is not aware about it, so we need to mention the same in the kt file, to do so following code needs to be added.

```
setContentView(R.layout.main_activity)
```

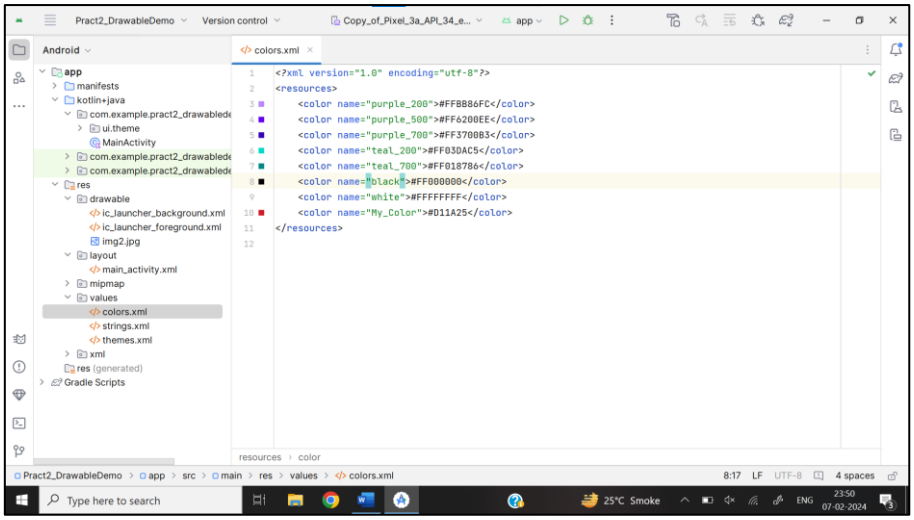


Now you can build the app and changes would be reflected on the AVD.



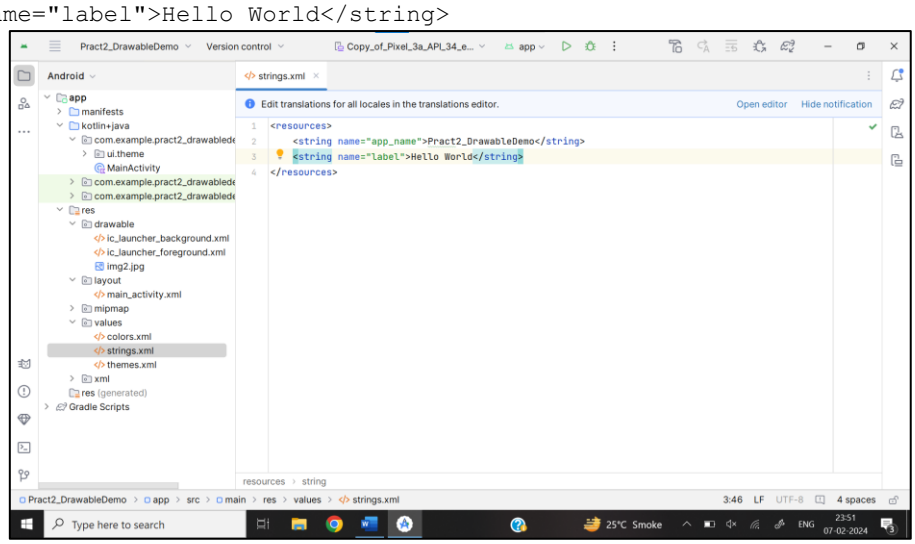
If there are certain colors in the application which has to be used multiple times so instead of remembering the hex code and repeating it every time, it can be simply be added to colors.xml file which is present under the folder res->values, as mentioned below

```
<color name="red">#FF0000</color>
```

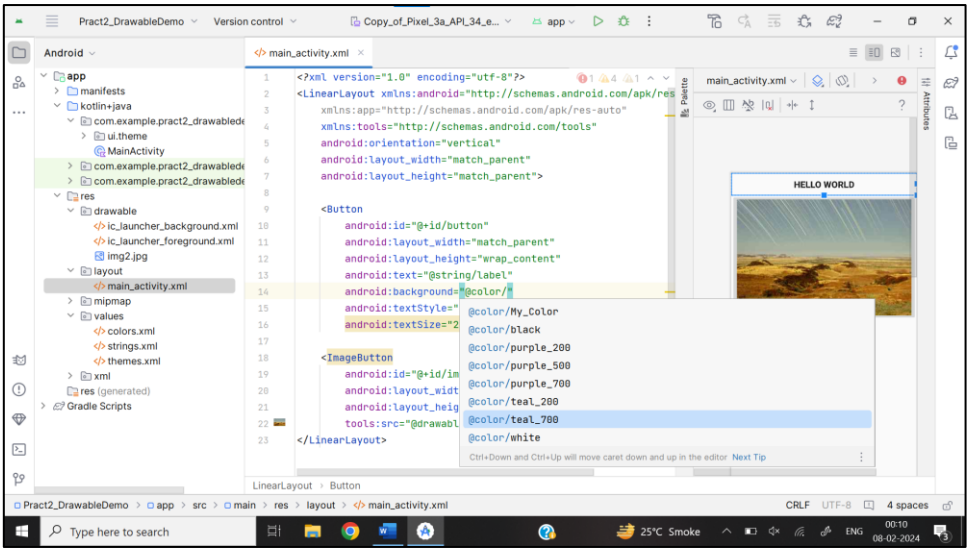


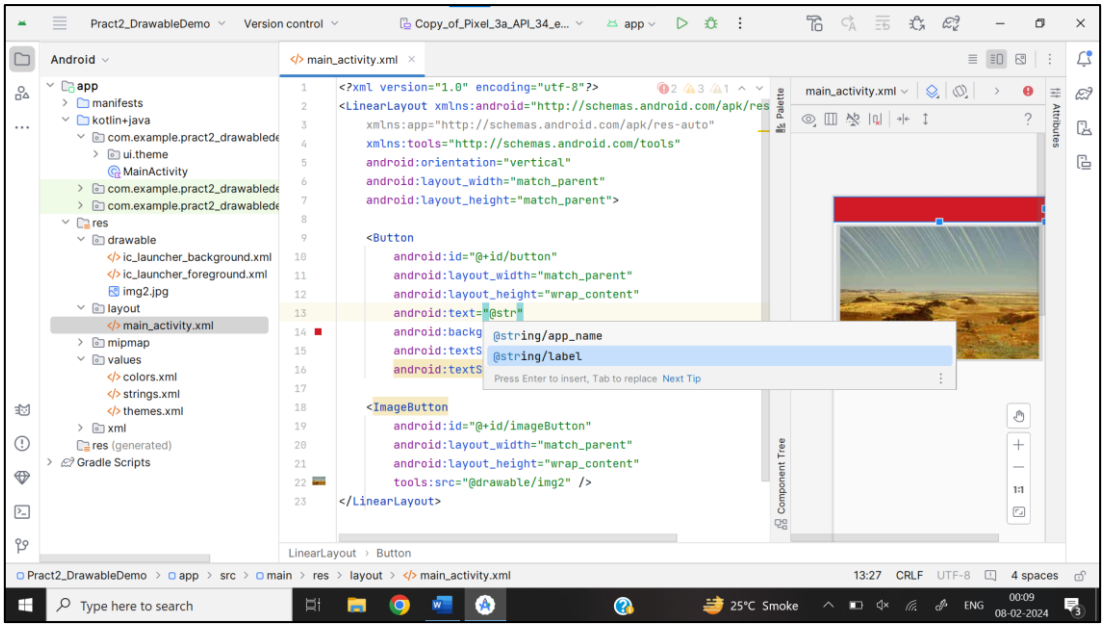
Likewise if there are certain string value in the application which has to be used multiple times so instead of remembering and repeating it every time, it can be simply be added to strings.xml file which is present under the folder res->values, as mentioned below

```
<string name="label">Hello World</string>
```



The color and string that was newly added is now visble in the options and it can be used.




Pract2_DrawableDemo
Version control
Copy_of_Pixel_3a_API_34_e...
app
Android
main_activity.xml
app
manifests
kotlin+java
com.example.pract2_drawablede
ui.theme
MainActivity
com.example.pract2_drawablede
com.example.pract2_drawablede
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml
img2.jpg
layout
main_activity.xml
mipmap
values
colors.xml
strings.xml
themes.xml
xml
res (generated)
Gradle Scripts
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<Button
android:id="@+id/button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@str"
android:backg
android:textS
android:textS
<ImageButton
android:id="@+id/imageButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
tools:src="@drawable/img2" />
</LinearLayout>
@string/app_name
@string/label
Press Enter to insert, Tab to replace Next Tip
Palette
Attributes
Component Tree
LinearLayout
Button
Pract2_DrawableDemo
app
src
main
res
layout
main_activity.xml
13:27
CRLF
UTF-8
4 spaces
Type here to search
25°C Smoke
ENG
00:09
08-02-2024

Practical 3: Programming Activities and fragments
Activity Life Cycle, Activity methods, Multiple Activities, Life Cycle of fragments and multiple fragments.

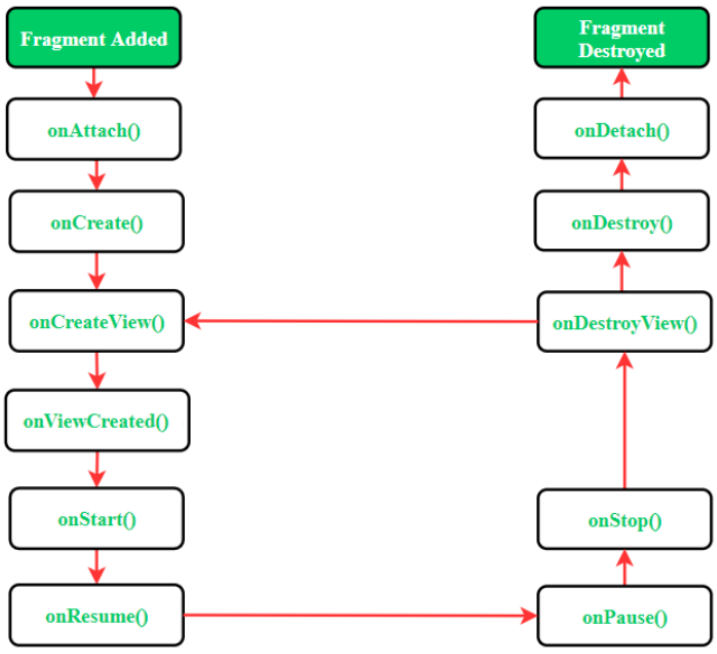
In Android, the fragment is the part of Activity which represents a portion of User Interface(UI) on the screen. It is the modular section of the android activity that is very helpful in creating UI designs that are flexible in nature and auto-adjustable based on the device screen size. The UI flexibility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

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

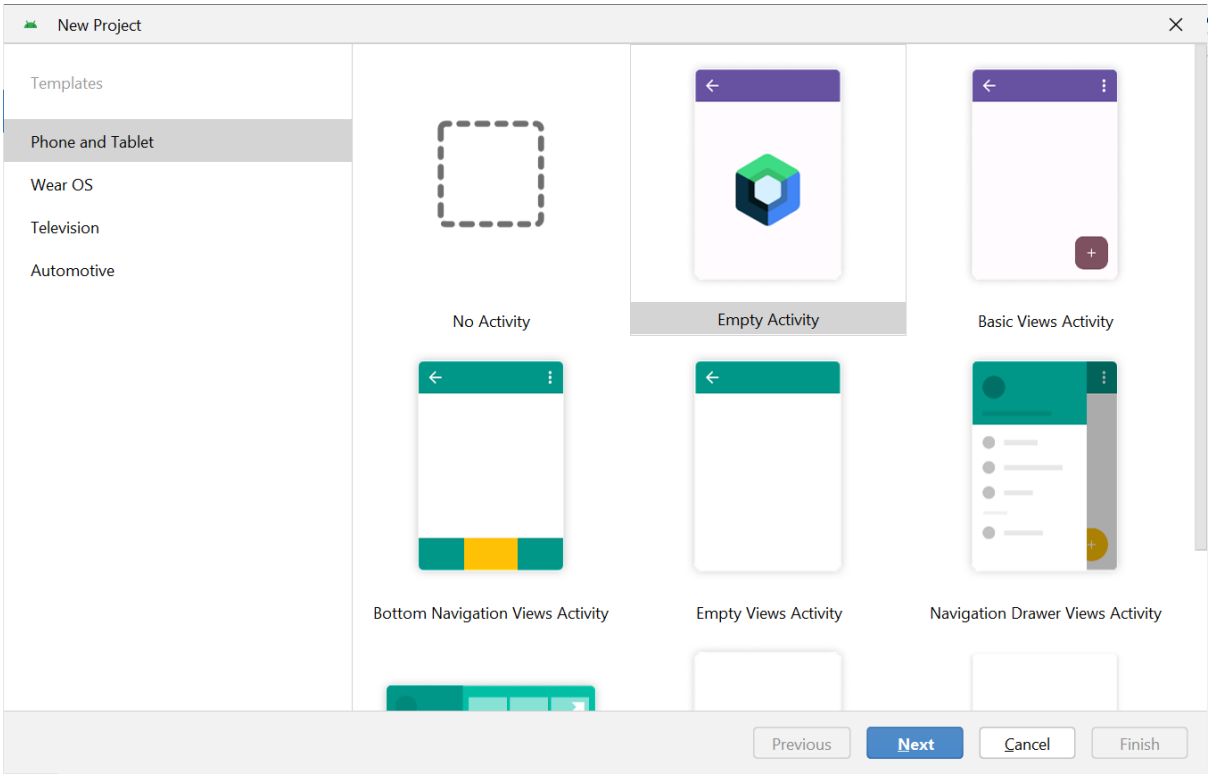
Types of Android Fragments

1. Single Fragment: Display only one single view on the device screen. This type of fragment is mostly used for mobile phones.
2. List Fragment: This Fragment is used to display a list-view from which the user can select the desired sub-activity. The menu drawer of apps like Gmail is the best example of this kind of fragment.
3. Fragment Transaction: This kind of fragments supports the transition from one fragment to another at run time. Users can switch between multiple fragments like switching tabs.

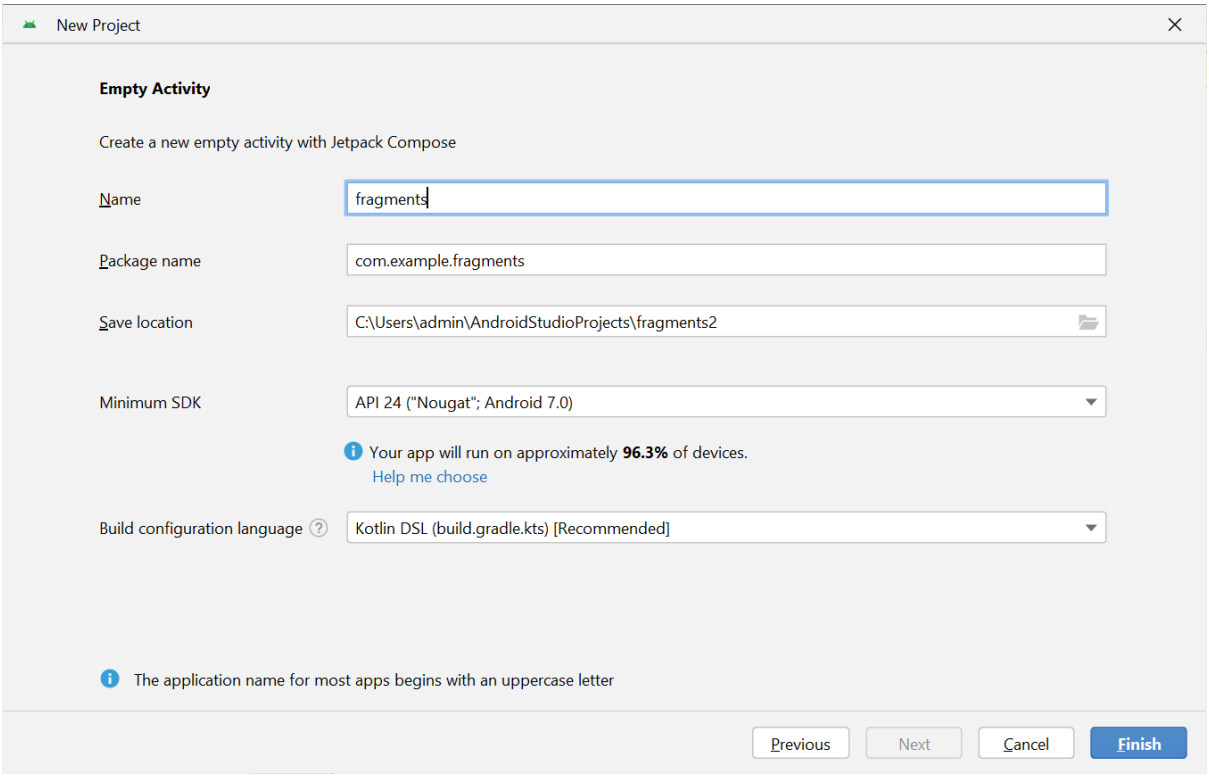
Fragment Lifecycle



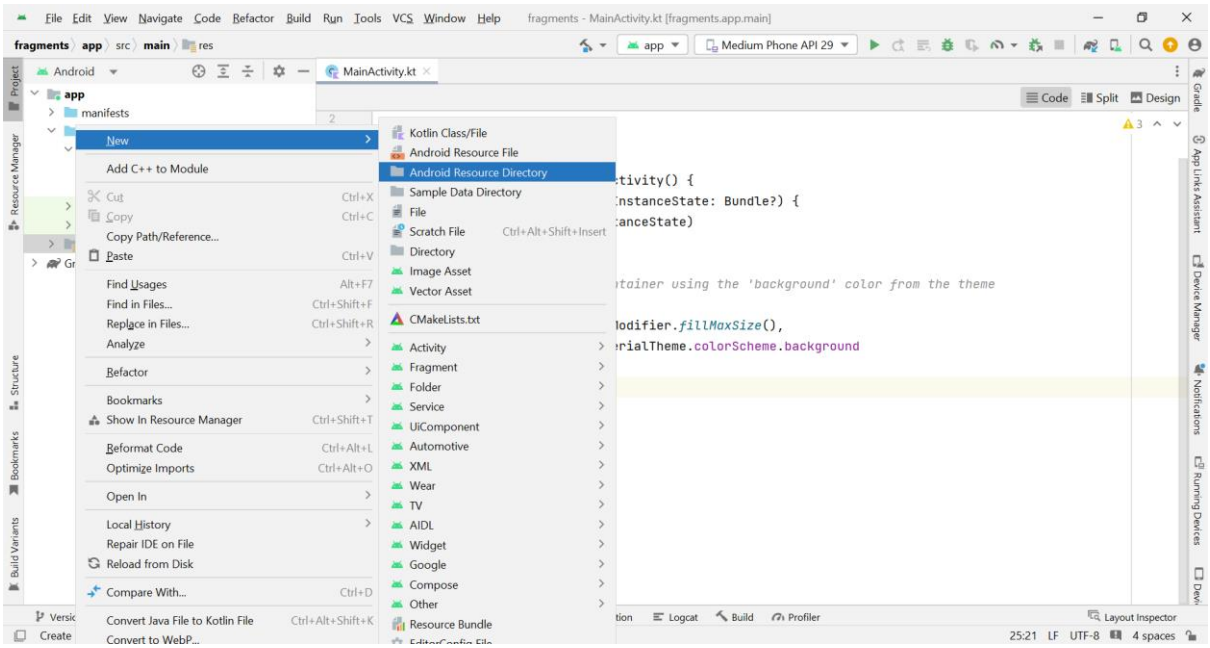
Step 1: Create a new project with empty activity.

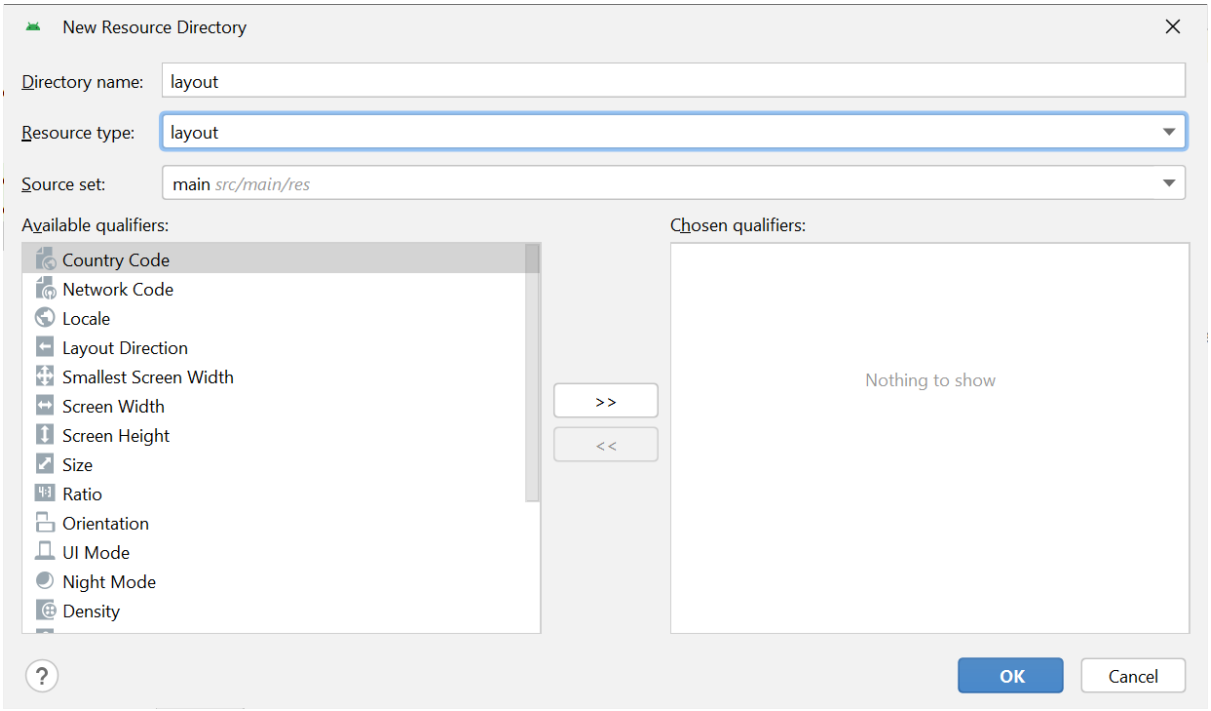


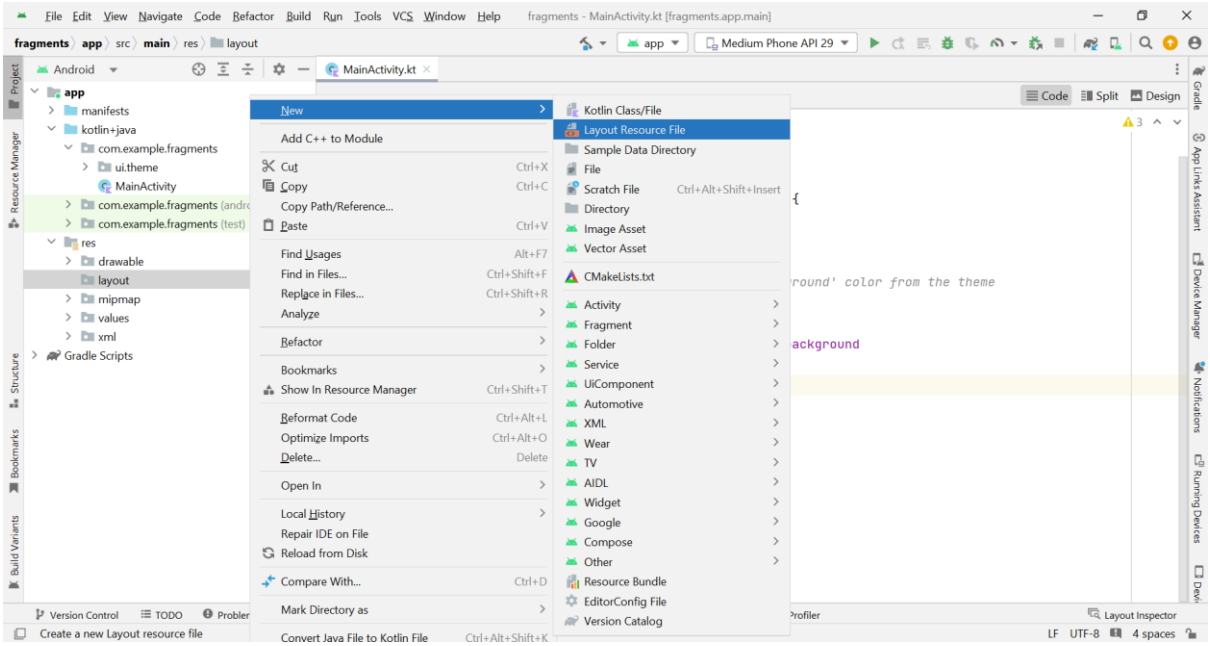
Step 2: Name the project as per your wish.

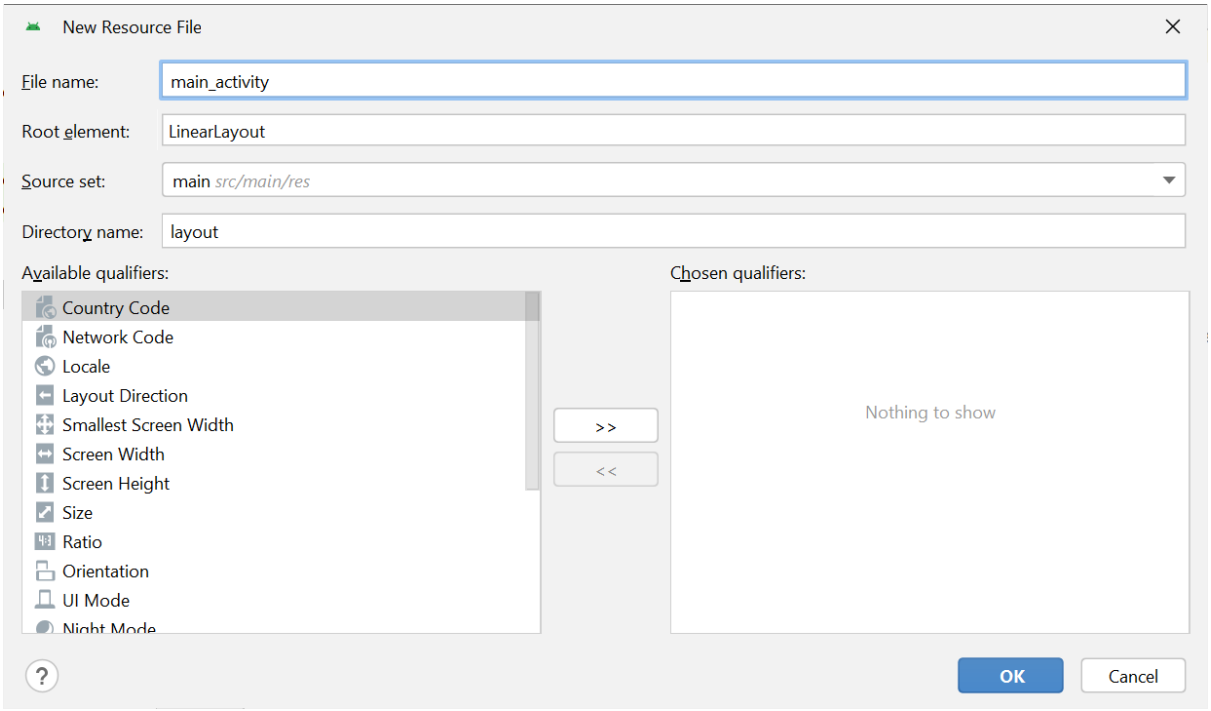


Step 3: Since we have created a project with empty activity it won't be having any xml file(this file is used to design the interface for the application) to create that we need to create layout directory first and then add the layout file.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
fragments - MainActivity.kt [fragments.app.main]
fragments app src main res
app
Medium Phone API 29
Android
MainActivity.kt
Code Split Design
app
manifests
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Convert to WebP...
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
Image Asset
Vector Asset
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
TV
AIDL
Widget
Google
Compose
Other
Resource Bundle
tivity() {
nstanceState: Bundle?) {
anceState)
tainer using the 'background' color from the theme
lodifier.fillMaxSize(),
rialTheme.colorScheme.background
Logcat Build Profiler
Layout Inspector
Create
25:21 LF UTF-8 4 spaces


New Resource Directory
Directory name: layout
Resource type: layout
Source set: main src/main/res
Available qualifiers:
Country Code
Network Code
Locale
Layout Direction
Smallest Screen Width
Screen Width
Screen Height
Size
Ratio
Orientation
UI Mode
Night Mode
Density
>>
<<
Chosen qualifiers:
Nothing to show
OK
Cancel

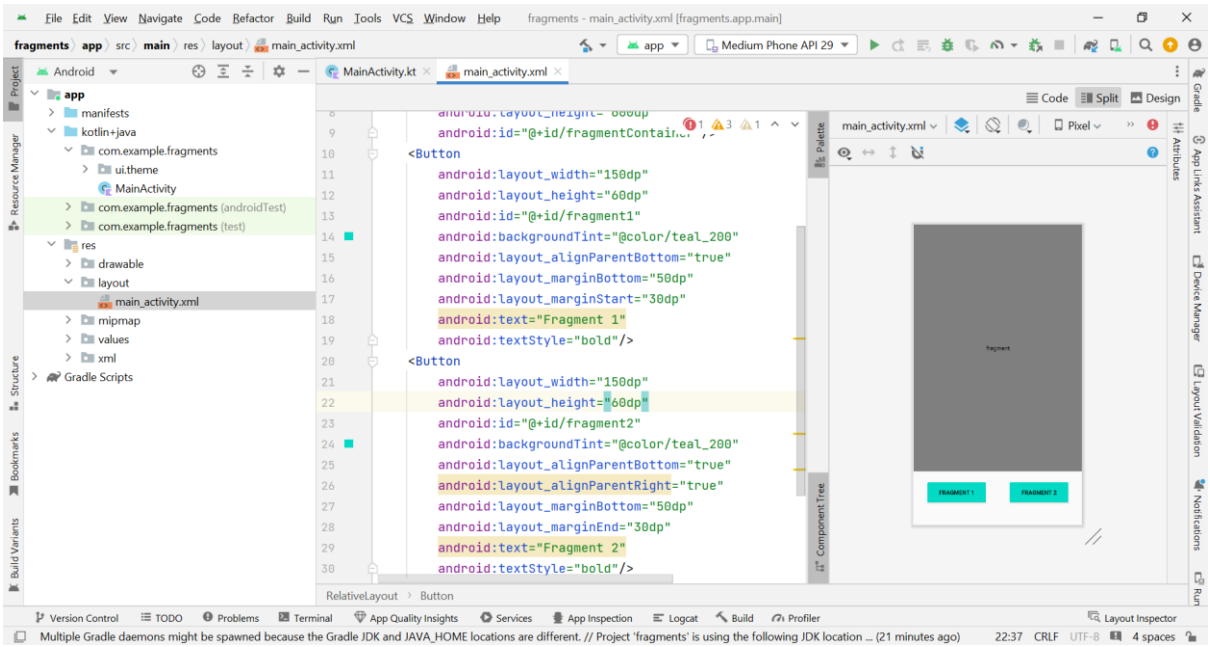




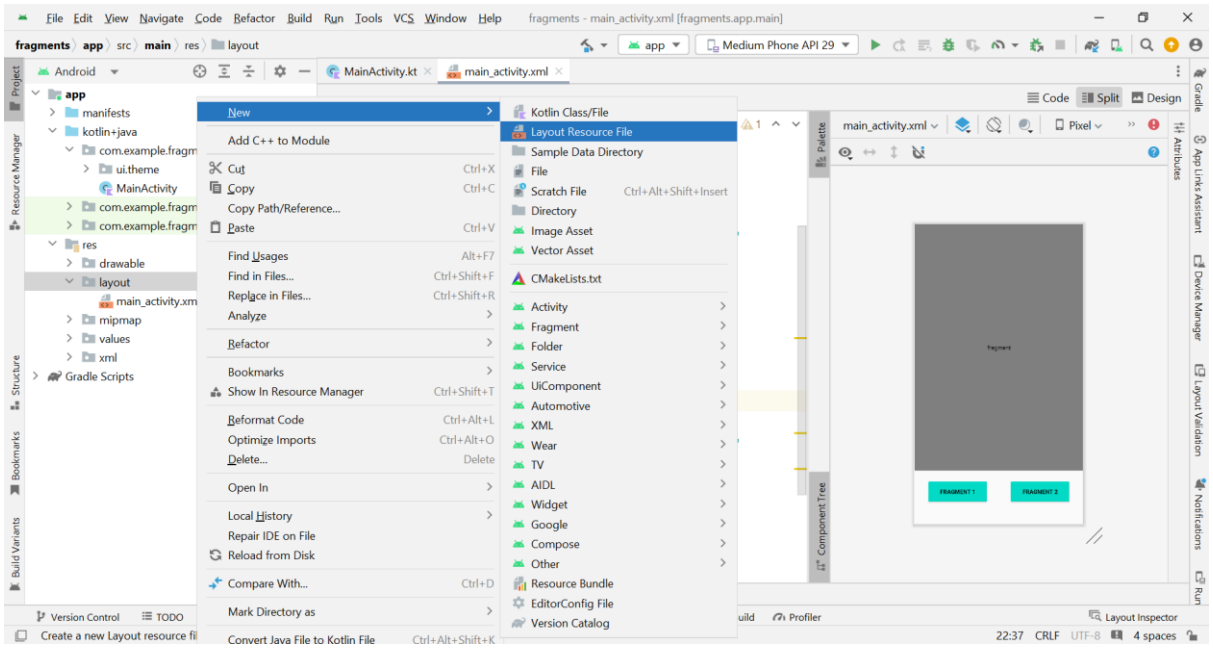
Step 4: Once the file has been created which is named as main_activity here, since it is a blank file , we are adding three components here which are two buttons and a fragment. Change the layout to RelativeLayout and add the following code inside it.

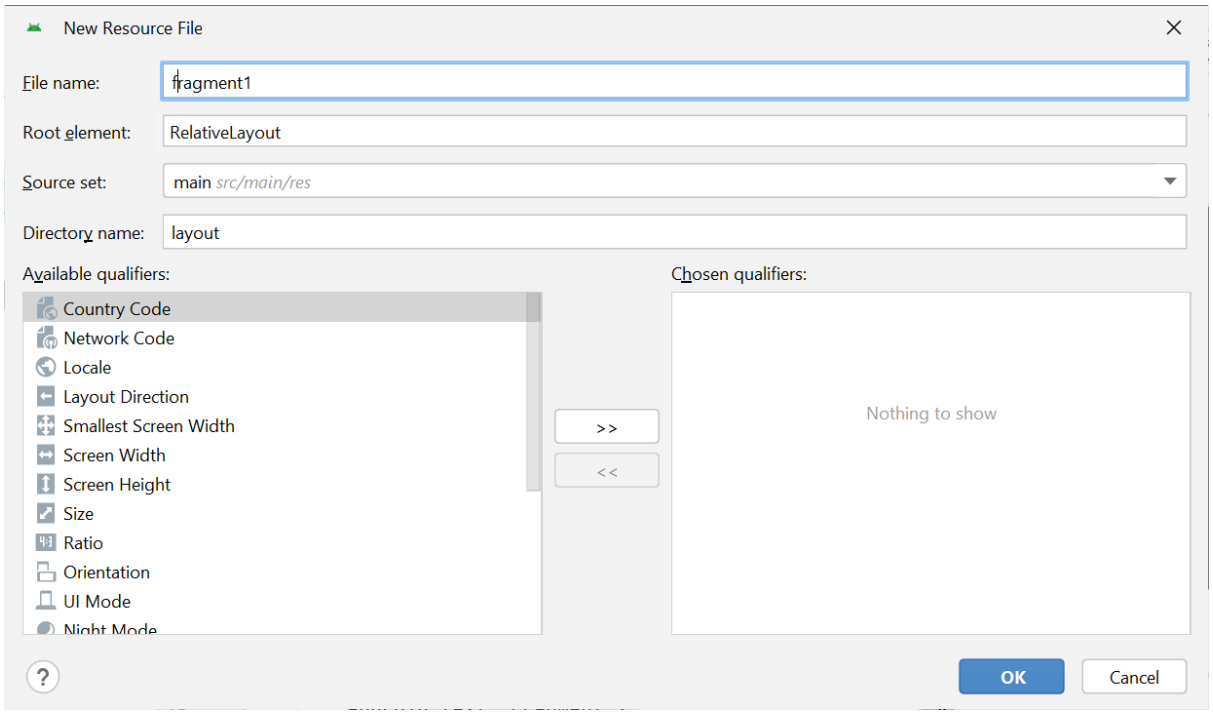
```
<fragment
    android:layout_width="match_parent"
    android:layout_height="600dp"
    android:id="@+id/fragmentContainer"/>
<Button
    android:layout_width="150dp"
    android:layout_height="60dp"
    android:id="@+id/fragment1"
    android:backgroundTint="@color/teal_200"
    android:layout_alignParentBottom="true"
```

```
    android:layout_marginBottom="50dp"
    android:layout_marginStart="30dp"
    android:text="Fragment 1"
    android:textStyle="bold"/>
<Button
    android:layout_width="150dp"
    android:layout_height="60dp"
    android:id="@+id/fragment2"
    android:backgroundTint="@color/teal_200"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="50dp"
    android:layout_marginEnd="30dp"
    android:text="Fragment 2"
    android:textStyle="bold"/>
```



Step 5: Now the first fragment has to be created, to do so right click on layout folder and select New -> Layout Resource Fie and name it has Fragment1 and change its layout to RelativeLayout.





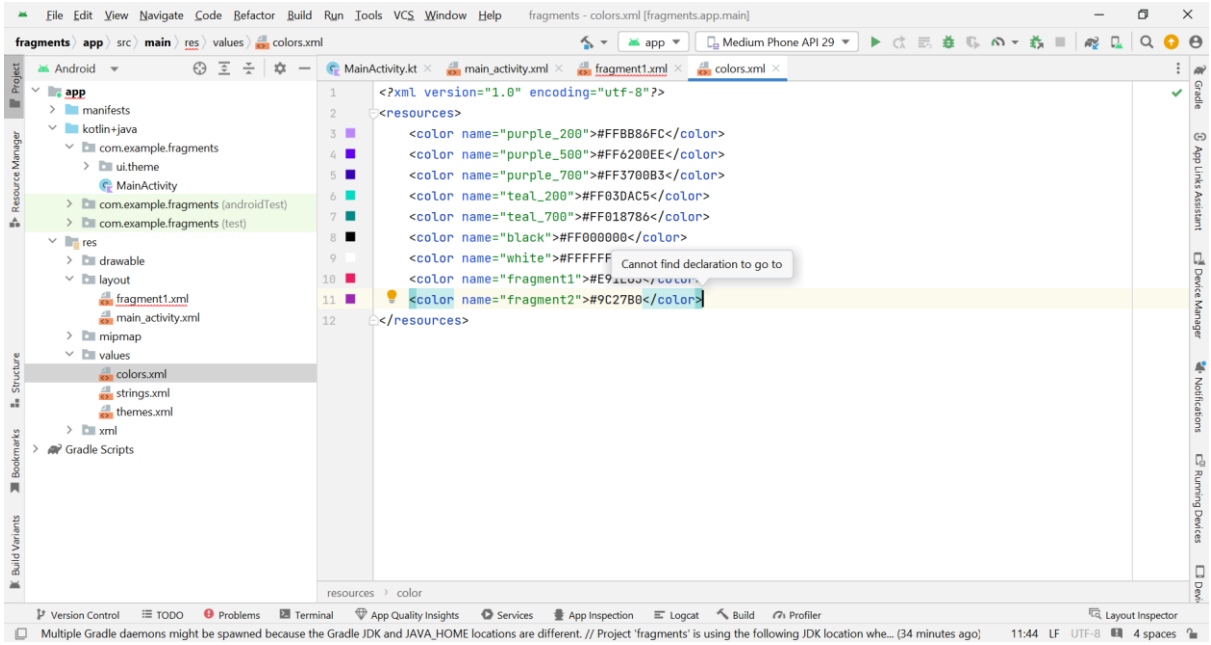
Add the following code below to the layout.

```
<?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"
    android:background="@color/fragment1">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:text="This is Fragment 1"
    android:textSize="40dp"
```

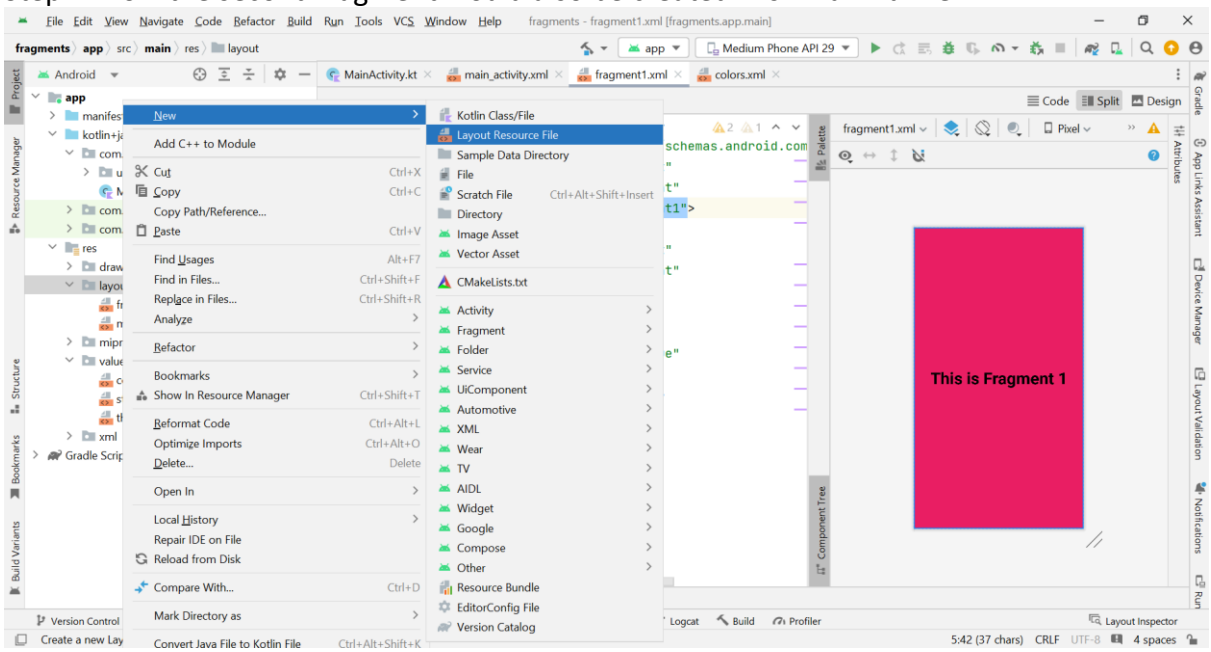
```
    android:layout_centerInParent="true"
    android:layout_marginTop="50dp"
    android:textColor="@color/black"
</RelativeLayout>
```

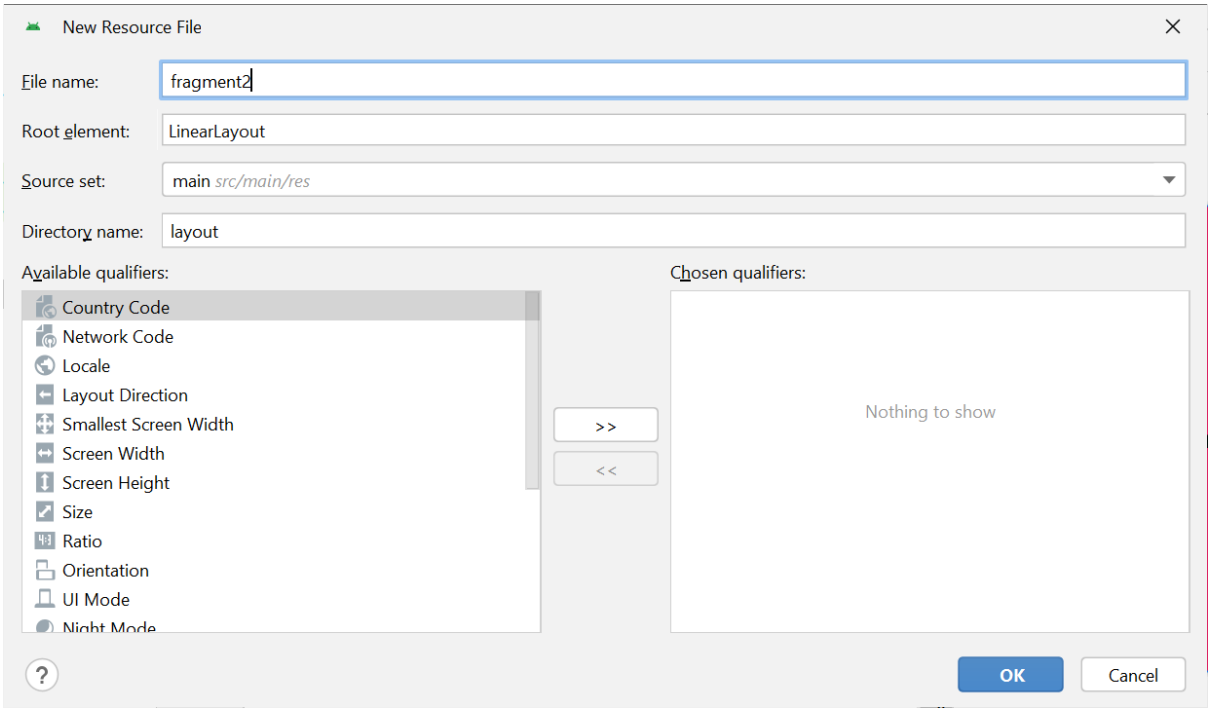
Step 6: To add background color to the fragments, add the color to color.xml file

```
<color name="fragment1">#E91E63</color>
<color name="fragment2">#9C27B0</color>
```



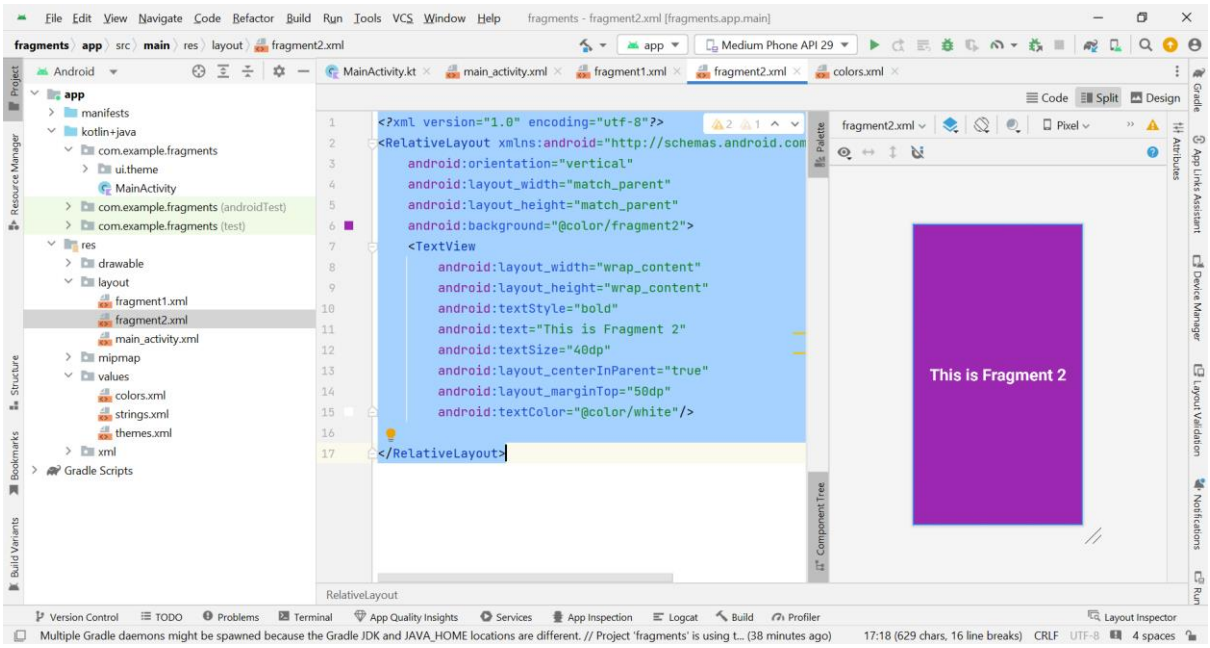
Step 7: Now the second fragment would also be created in similar manner.





Add the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/fragment2">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:text="This is Fragment 2"
        android:textSize="40dp"
        android:layout_centerInParent="true"
        android:layout_marginTop="50dp"
        android:textColor="@color/white"/>

</RelativeLayout>
```



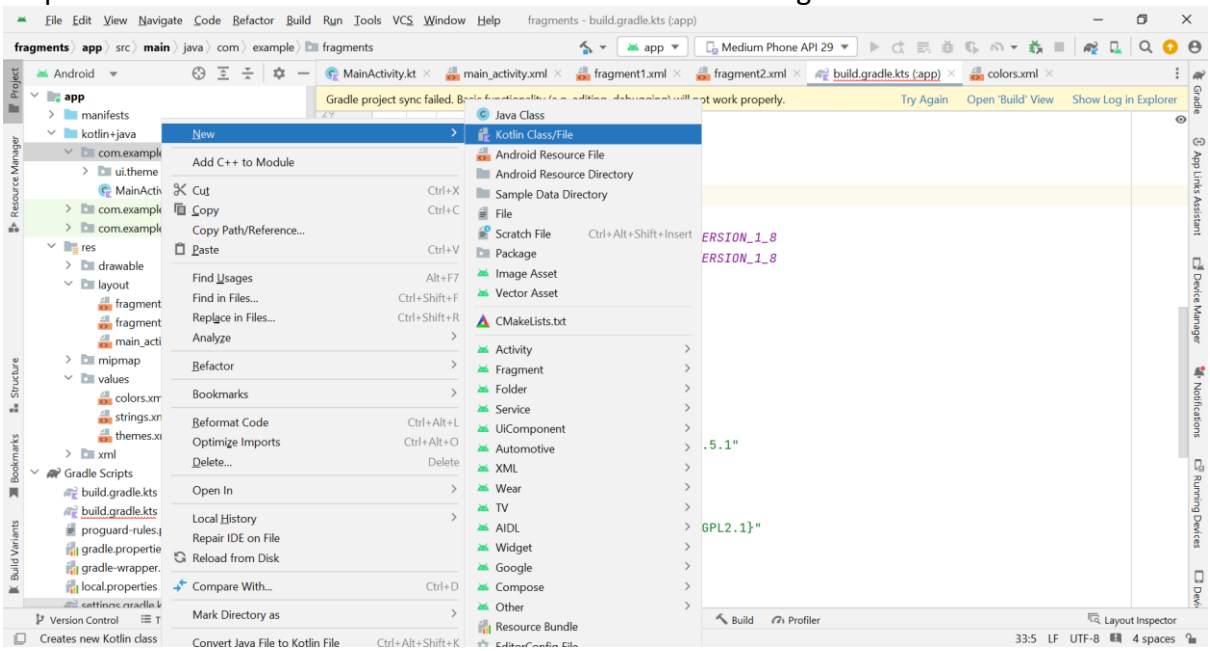
Step 8: Now goto build.gradle and enable view binding feature by add the code mentioned below in buildFeatures function

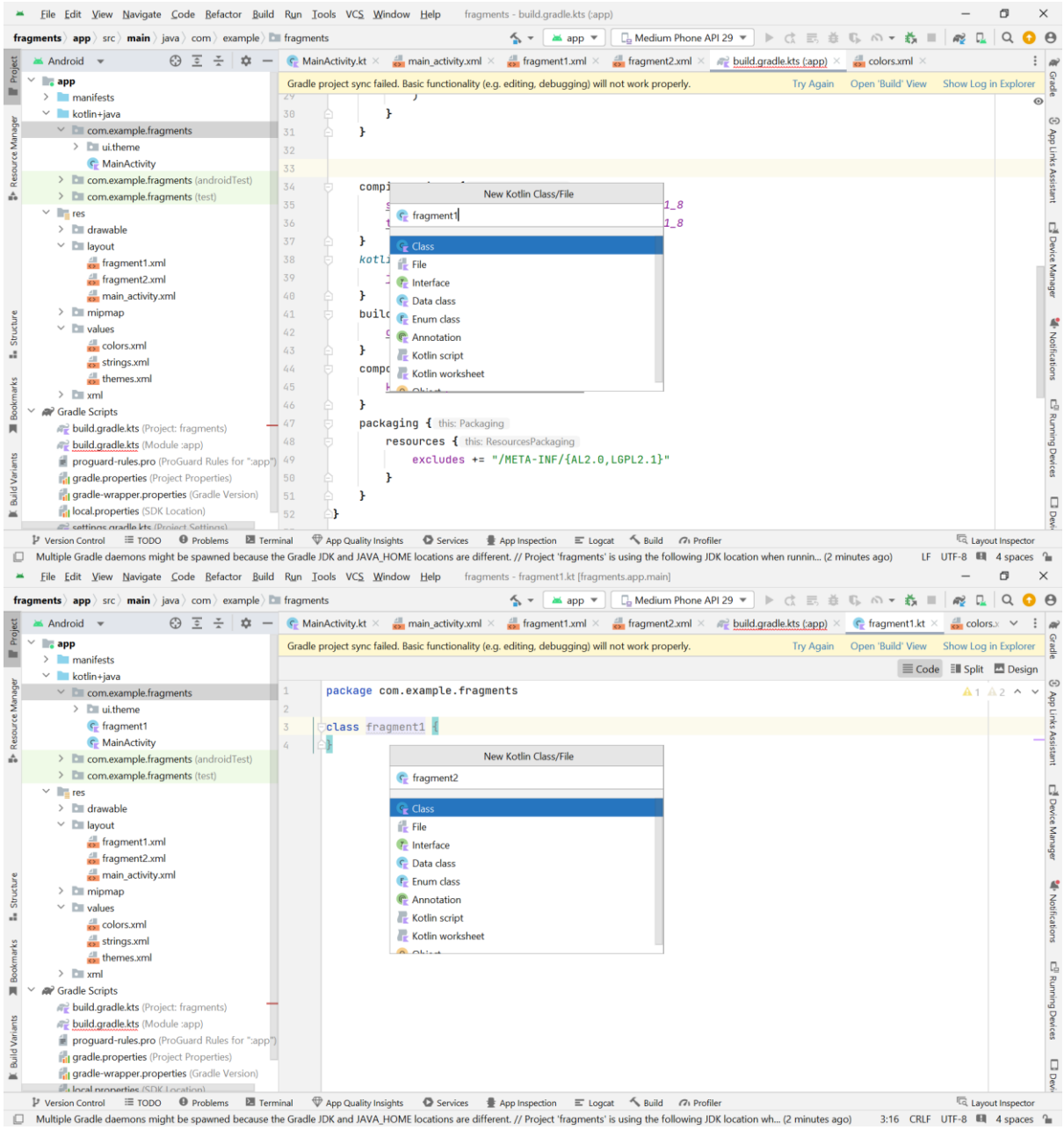
```
viewBinding = true
dataBinding = true
```

and then in dependencies add the following code:

```
implementation ("androidx.fragment:fragment-ktx:1.5.5")
```

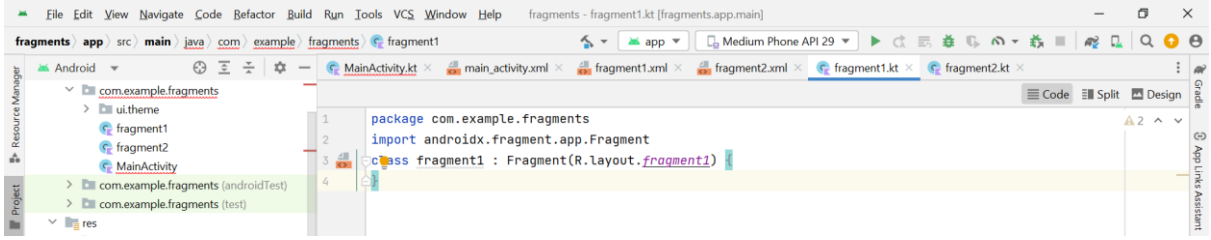
Step 9: Now Kotlin class file has to be created for both the fragment files.





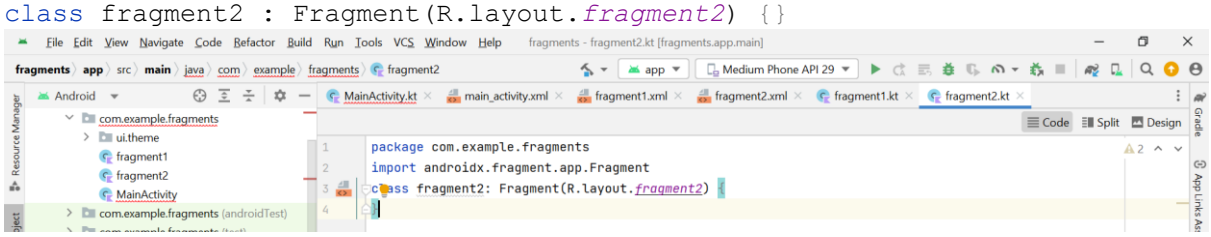
Step 10: Now add the following code to fragment1.kt file

```
package com.example.fragments
import androidx.fragment.app.Fragment
class fragment1 : Fragment(R.layout.fragment1) {}
```



Step 11: Now add the following code to fragment2.kt file

```
package com.example.fragments
import androidx.fragment.app.Fragment
class fragment2 : Fragment(R.layout.fragment2) { }
```



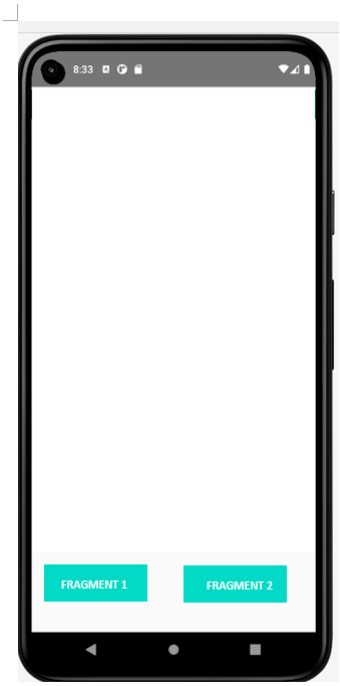
Step 12: Now in main_activity.kt file update the following code:

```
package com.example.fragments

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.fragments.ui.theme.FragmentsTheme
import android.view.View
import androidx.fragment.app.Fragment
import com.example.fragments.databinding.MainActivityBinding

class MainActivity : ComponentActivity() {
    lateinit var binding: MainActivityBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        binding.fragment1.setOnClickListener {
            replaceFragment(fragment1())
        }
        binding.fragment2.setOnClickListener {
            replaceFragment(fragment2())
        }}


}
        private fun replaceFragment(fragment: Fragment){
          val fragmentManager = supportFragmentManager
            val fragmentTransaction = fragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainer, fragment)
            fragmentTransaction.commit()
        }
```

Put your outputs here


This is Fragment 1


This is Fragment 2


FRAGMENT 1
FRAGMENT 2


This is Fragment 1
FRAGMENT 1
FRAGMENT 2


This is Fragment 2
FRAGMENT 1
FRAGMENT 2

Practical 4 : Programs related to different Layouts Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View

What is Android View?

A View serves as the user interface responsible for creating interactive UI elements like TextView, ImageView, EditText, RadioButton, etc. It handles event interactions and drawing tasks, commonly referred to as Widgets.

An Android Layout is employed to specify the user interface that houses the UI controls or widgets visible on the screen of an Android application or activity. Typically, each application is a blend of View and ViewGroup. Given that an Android application comprises numerous activities, and each activity represents a distinct page within the application, it contains multiple user interface components. These components are instances of the View and ViewGroup. Consequently, the elements in a layout are constructed using a hierarchy of View and ViewGroup objects.

Different Types of Layouts in Android Kotlin

1. **Linear Layout**

LinearLayout, a subclass of ViewGroup, is employed to arrange child View elements sequentially in a specific direction, either horizontally or vertically, determined by the orientation property.

2. Relative Layout

RelativeLayout, belonging to the ViewGroup subclass, serves to define the positioning of child View elements in relation to each other, such as (A to the right of B) or in relation to the parent (fixed to the top of the parent).

3. Constraint Layout

ConstraintLayout, a subclass of ViewGroup, is utilized to define layout constraints for each child View in relation to other views within the layout. While similar to RelativeLayout, ConstraintLayout offers enhanced capabilities.

4. Table Layout

As a subclass of ViewGroup, TableLayout is employed to present child View elements in organized rows and columns.

5. Frame Layout

A subclass of ViewGroup, FrameLayout is utilized to dictate the arrangement of View elements it encompasses, stacking them on top of each other to exhibit only a single View within the FrameLayout.

6. List View

A ListView serves as a user interface element employed to showcase a vertically scrollable list of items. This adaptable widget provides a means to present data in a flexible format. Within the list, each item represents an individual view, allowing for customization of their appearance and behavior according to the specific needs of your application.

7. Grid View

The GridView serves as a user interface component specifically crafted to exhibit data in a scrollable grid format with a distinctive two-dimensional structure. Functioning as a highly efficient widget, it excels in presenting information in a tabular layout, providing the capability to organize items seamlessly across both rows and columns.

8. Absolute Layout

An Absolute Layout permits the precise definition of the location, specifically the X and Y coordinates, of its children relative to the origin at the top-left corner of the layout. However, the use of absolute layout is discouraged due to its inflexibility and challenges in maintaining consistency across screens with different sizes.

Android linear layout

Android linear layout is employed for organizing elements in a linear fashion, signifying one element per line. Various types of forms on Android can be crafted using this layout, with elements arranged in a top-to-bottom manner.

There are two possible orientations:

a. Vertical Orientation – Widgets like TextViews are arranged vertically.

b. Horizontal Orientation – TextViews are arranged horizontally.

Android RelativeLayout

Android RelativeLayout is designed to specify the position of elements in relation to other elements within the layout.

In a RelativeLayout, it is possible to align elements with the parent container.

main_activity.xml:

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

    <TextView
        android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Layout Demo"
        android:textSize="30dp"
        android:textAlignment="center"
        android:background="@color/teal_700"
        android:textColor="@color/white"
        android:textStyle="bold"
        android:layout_marginTop="10px"/>
    <TextView
        android:id="@+id/choicelabel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click on button to view respective layouts"
        android:textSize="20dp"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textStyle="italic"
        android:layout_marginTop="30dp"/>

    <Button
        android:id="@+id/linearlayouth"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Linear Layout (Horizontal)"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="50dp"
        />
    <Button
        android:id="@+id/linearlayoutv"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Linear Layout (Vertical)"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
    <Button
        android:id="@+id/relativelayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Relative Layout"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
        <Button
        android:id="@+id/tablelayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Table Layout"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
```

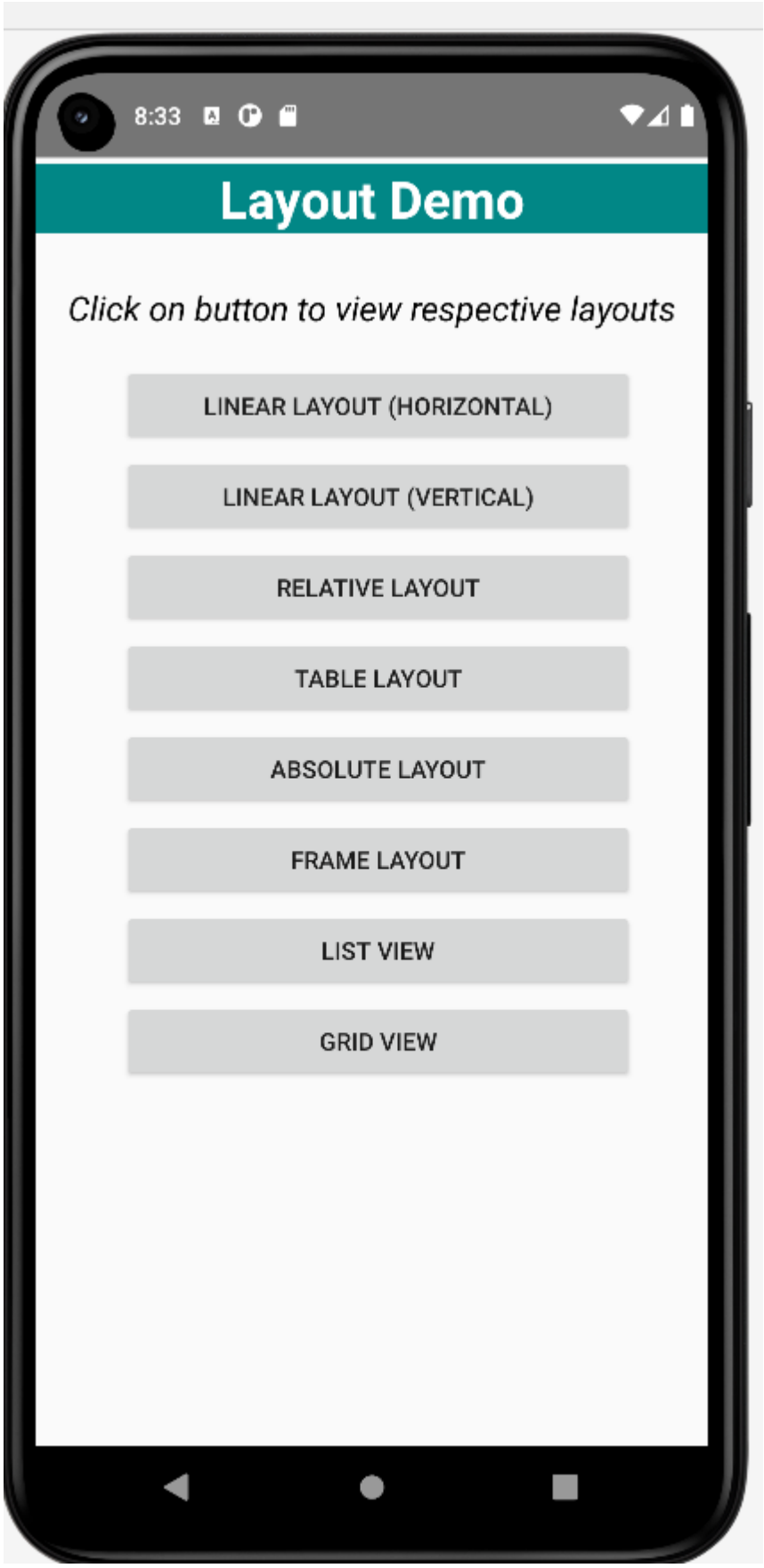
```
        />
    <Button
        android:id="@+id/abslayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Absolute Layout"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
    <Button
        android:id="@+id/framelayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Frame Layout"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
    <Button
        android:id="@+id/listviewlayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="List View"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
    <Button
        android:id="@+id/gridviewlayout"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Grid View"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="50dp"
        />
</LinearLayout>
```

mainActivity.kt:

```
package com.example.layout_demo

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout_demo.ui.theme.Layout_demoTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        val hlinbtn = findViewById<Button>(R.id.linearlayouth)
        hlinbtn.setOnClickListener {
```

```
            val intent = Intent(this, linear_layouth::class.java)
            startActivity(intent)
        }
        val vlinbtn = findViewById<Button>(R.id.linearlayoutv)
        vlinbtn.setOnClickListener {
            val intent = Intent(this, linear_layoutv::class.java)
            startActivity(intent)
        }
        val relbtn = findViewById<Button>(R.id.relativelayout)
        relbtn.setOnClickListener {
            val intent = Intent(this, relative_layout::class.java)
            startActivity(intent)
        }

        val tblbtn = findViewById<Button>(R.id.tablelayout)
        tblbtn.setOnClickListener {
            val intent = Intent(this, table_layout::class.java)
            startActivity(intent)
        }
        val absbtn = findViewById<Button>(R.id.abslayout)
        absbtn.setOnClickListener {
            val intent = Intent(this, absolute_layout::class.java)
            startActivity(intent)
        }
        val frmbtn = findViewById<Button>(R.id.framelayout)
        frmbtn.setOnClickListener {
            val intent = Intent(this, frame_layout::class.java)
            startActivity(intent)
        }
        val listviewbtn = findViewById<Button>(R.id.listviewlayout)
        listviewbtn.setOnClickListener {
            val intent = Intent(this, list_view::class.java)
            startActivity(intent)
        }
        val gridviewbtn = findViewById<Button>(R.id.gridviewlayout)
        gridviewbtn.setOnClickListener {
            val intent = Intent(this, grid_view::class.java)
            startActivity(intent)
        }
    }}
```

Output:


8:33
Layout Demo
Click on button to view respective layouts
LINEAR LAYOUT (HORIZONTAL)
LINEAR LAYOUT (VERTICAL)
RELATIVE LAYOUT
TABLE LAYOUT
ABSOLUTE LAYOUT
FRAME LAYOUT
LIST VIEW
GRID VIEW

Linear layout(vertical)

Main_activity.xml

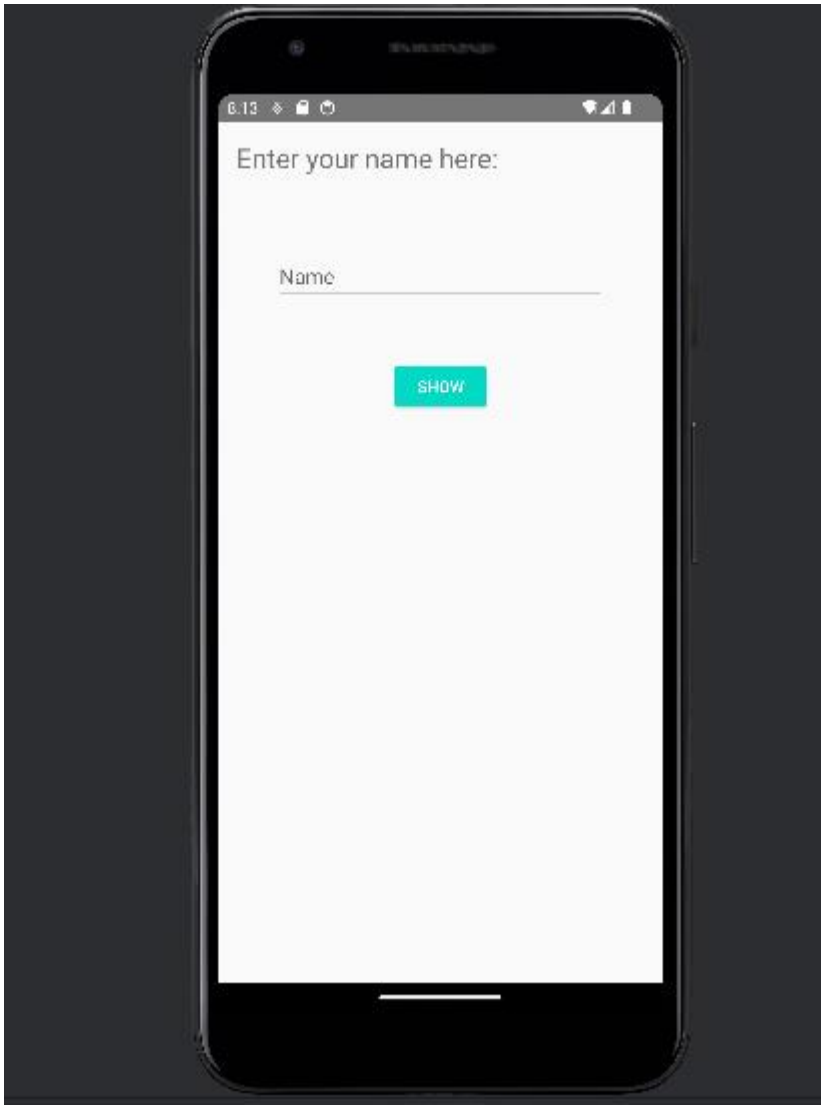
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:text="Enter your name here:"
        android:textSize="24dp"
        android:id="@+id/txtVw"/>

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:hint="Name"
        android:inputType="text"/>

    <Button
        android:id="@+id/showInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="show"
        android:backgroundTint="@color/teal_200"
        android:textColor="@android:color/white"/>

</LinearLayout>
```

Mainactivity.kt

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        val showButton
```

```
                = findViewById<Button>(R.id.showInput)
        val editText
                = findViewById<EditText>(R.id.editText)
        val textView
                = findViewById<TextView>(R.id.txtVw)
    }
}
```

Output:



Linear layout(horizontal)

Main_activity.xml

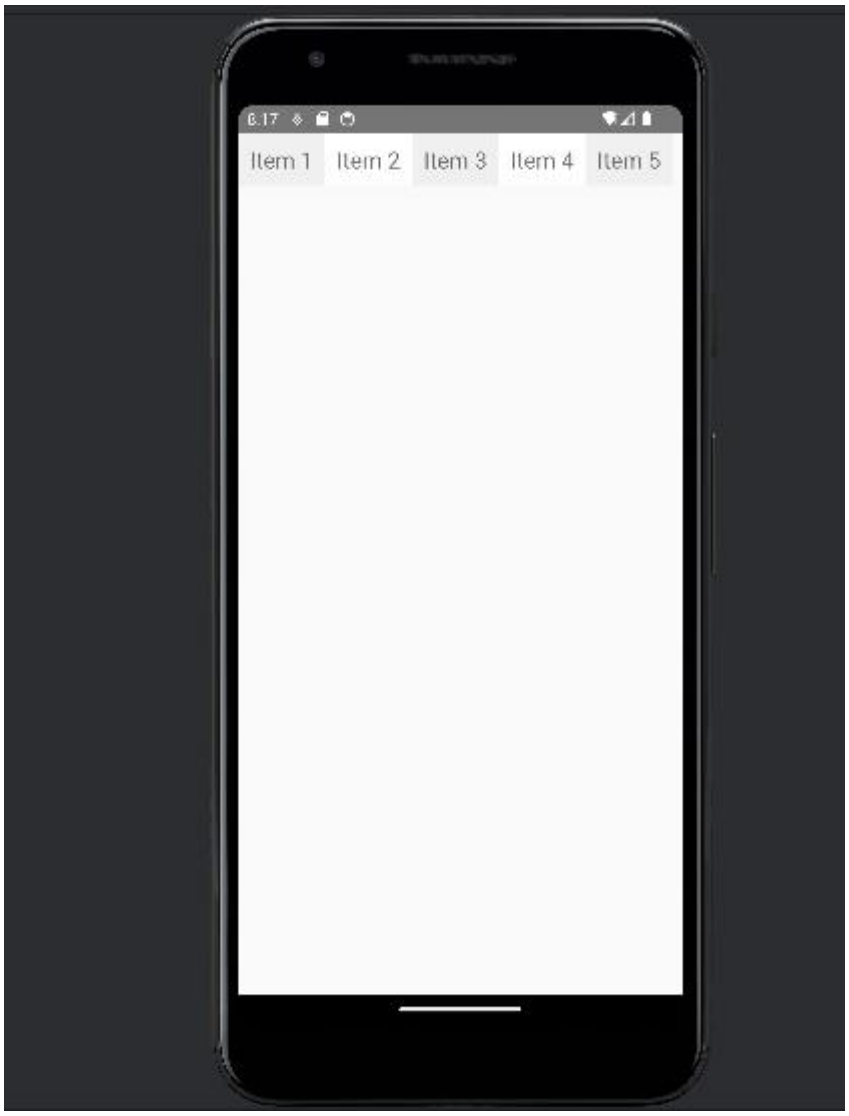
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10sp"
        android:textSize="20sp"
        android:background="#F0F0F0"
        android:text="Item 1" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10sp"
        android:textSize="20sp"
        android:background="#FFFFFF"
        android:text="Item 2" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10sp"
        android:textSize="20sp"
        android:background="#F0F0F0"
        android:text="Item 3" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10sp"
        android:textSize="20sp"
        android:background="#FFFFFF"
        android:text="Item 4" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10sp"
        android:textSize="20sp"
        android:background="#F0F0F0"
        android:text="Item 5" />


</LinearLayout>
```

Mainactivity.kt

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

    }
}
```

Output:



Relative Layout:

Mainactivity.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"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="First name:"
        android:layout_marginTop="20dp"
        android:textSize="20dp"/>

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/textView1"
        android:layout_marginTop="8dp"/>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="10dp"
        android:text="Last name:"
        android:textSize="20dp"/>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_toRightOf="@id/textView2"
        android:layout_marginTop="45dp"/>

    <Button
        android:id="@+id/btn4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/textView2"
        android:layout_marginTop="20dp"
        android:text="Submit" />
</RelativeLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
```

```
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

    }
}
```

Output:

Table Layout:

Mainactivity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp">

    <TextView
        android:id="@+id/txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ICC Ranking of Players:"
        android:textSize = "20dp"
        android:textStyle="bold">

    </TextView>


    <TableRow android:background="#51B435" android:padding="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Rank" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Player" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Team" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Points" />
    </TableRow>
    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="1" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Virat Kohli" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
```

```
            android:text="IND" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="895" />
    </TableRow>
    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="2" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Rohit Sharma" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="IND" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="863" />

    </TableRow>
    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="3" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Faf du Plessis" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="PAK" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="834" />

    </TableRow>

    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="4" />
        <TextView
```

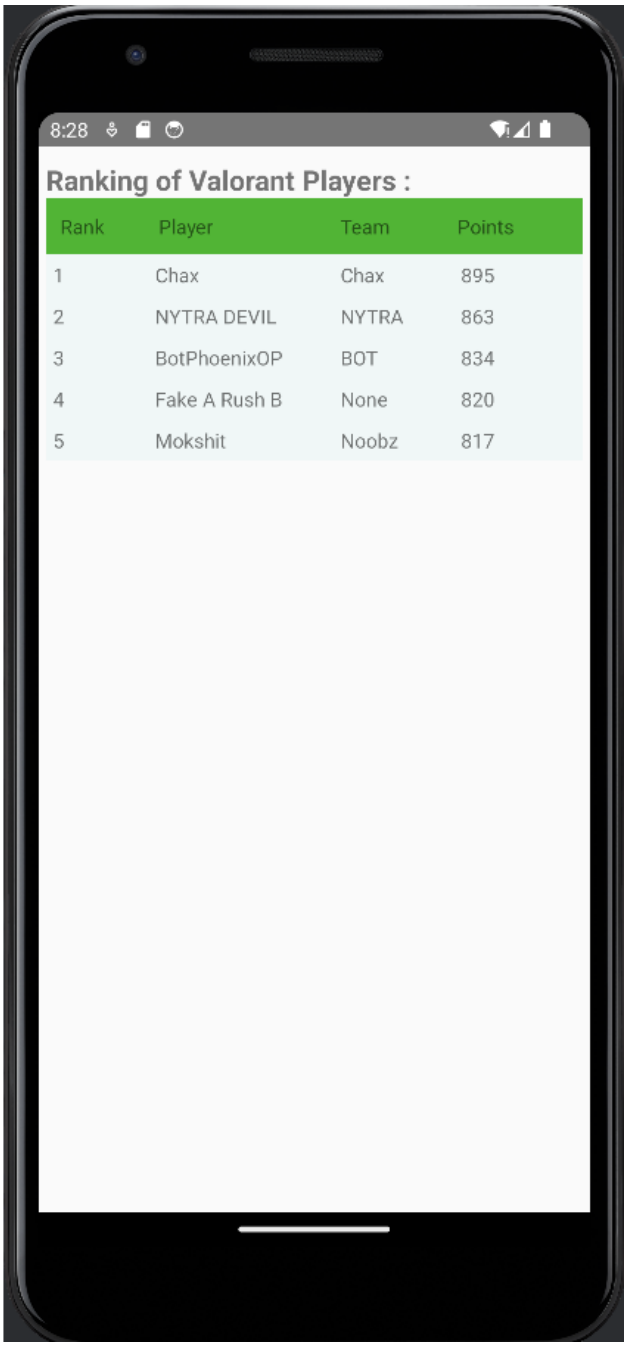
```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Steven Smith" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="AUS" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="820" />

    </TableRow>

    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="5" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Ross Taylor" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="NZ" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="817" />

    </TableRow>

</TableLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

    }
}
```

Output:



Absolute Layout:

Mainactivity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- using absolute lay(not recommended)-->
<AbsoluteLayout
   android:id="@+id/myAbsoluteLayout"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_height="match_parent"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   tools:context=".MainActivity">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/heading"
      android:text="@string/this_is_user_profile"
      android:gravity="center"
      android:textSize="25sp"
      android:layout_x="0dp"
      android:layout_y="0dp" />

   <ImageView
      android:id="@+id/zoomIn"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
    android:layout_x="26dp"
    android:layout_y="259dp"
    app:srcCompat="@android:drawable/btn_plus" />

<ImageView
    android:id="@+id/zoomOut"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="299dp"
    android:layout_y="266dp"
    app:srcCompat="@android:drawable/btn_minus" />

<ImageView
    android:id="@+id/imageView3"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_x="305dp"
    android:layout_y="628dp"
    app:srcCompat="@android:drawable/ic_menu_send" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="16dp"
    android:layout_y="666dp"
```

```
    android:padding="5dp"
    android:text="@string/save_to_favourite" />

  <ImageView
    android:id="@+id/celebImage"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_x="98dp"
    android:layout_y="263dp"
    app:srcCompat="@android:drawable/ic_lock_idle_low_battery" />

</AbsoluteLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

    }
}
```

Output:

Frame Layout:

Mainactivity.xml:

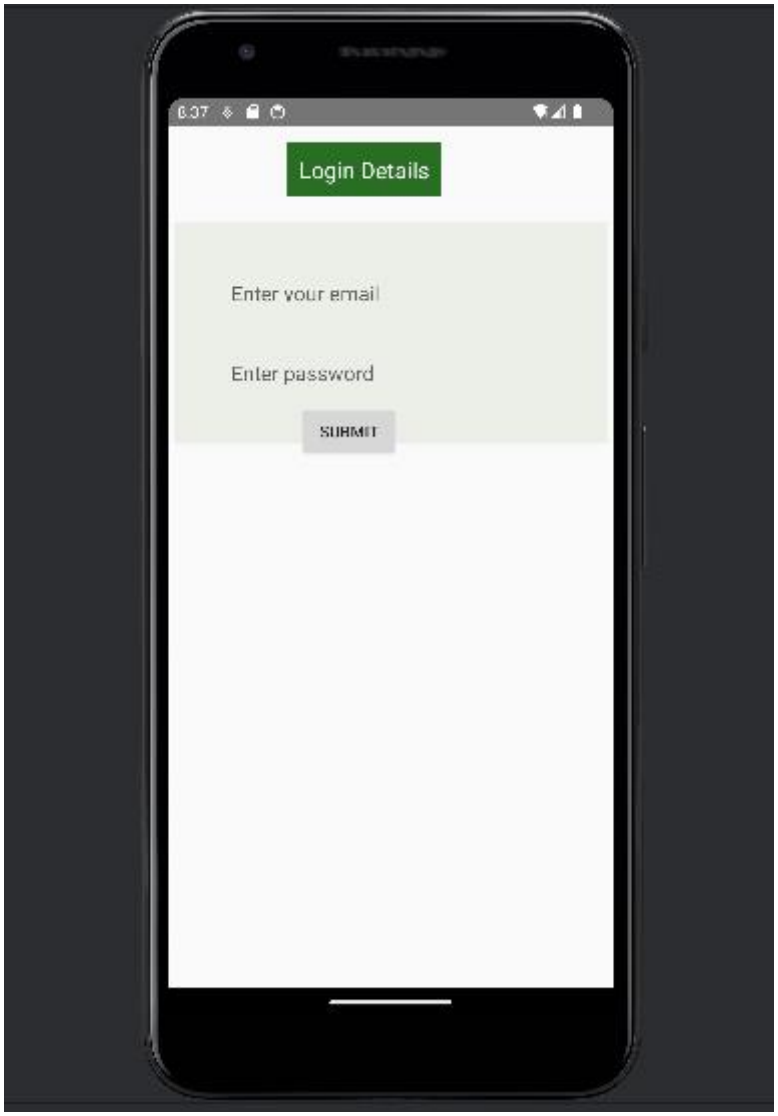
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:id="@+id/txtvw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="10dp"
        android:background="#286F24"
        android:padding="10dp"
        android:text="Login Details"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="80dp"
        android:background="#ECEEE8"
        android:hint="Enter your email"
        android:padding="10dp" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:background="#ECEEE8"
        android:hint="Enter password"
        android:padding="10dp" />

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="110dp"
        android:layout_marginTop="240dp"
        android:text="Submit" />
</FrameLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {
    private lateinit var textView: TextView
    private lateinit var editText1: EditText
    private lateinit var editText2: EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        textView = findViewById(R.id.txtvw1)
        editText1 = findViewById(R.id.editText1);
        editText2 = findViewById(R.id.editText2);

    }
}
```

Output:



List View:

Mainactivity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ListView
        android:id="@+id/userlist"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.ListView
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        val arrayAdapter: ArrayAdapter<*>
        val users = arrayOf(
            "Virat Kohli", "Rohit Sharma", "Steve Smith",
            "Kane Williamson", "Ross Taylor"
        )

        // access the listView from xml file
        var mListView = findViewById<ListView>(R.id.userlist)
        arrayAdapter = ArrayAdapter(this,
            android.R.layout.simple_list_item_1, users)
        mListView.adapter = arrayAdapter

    }
}
```

AndroidManifest.xml

```
<dist:module dist:instant="true" />
```

Output:

Grid View:

Mainactivity.xml:

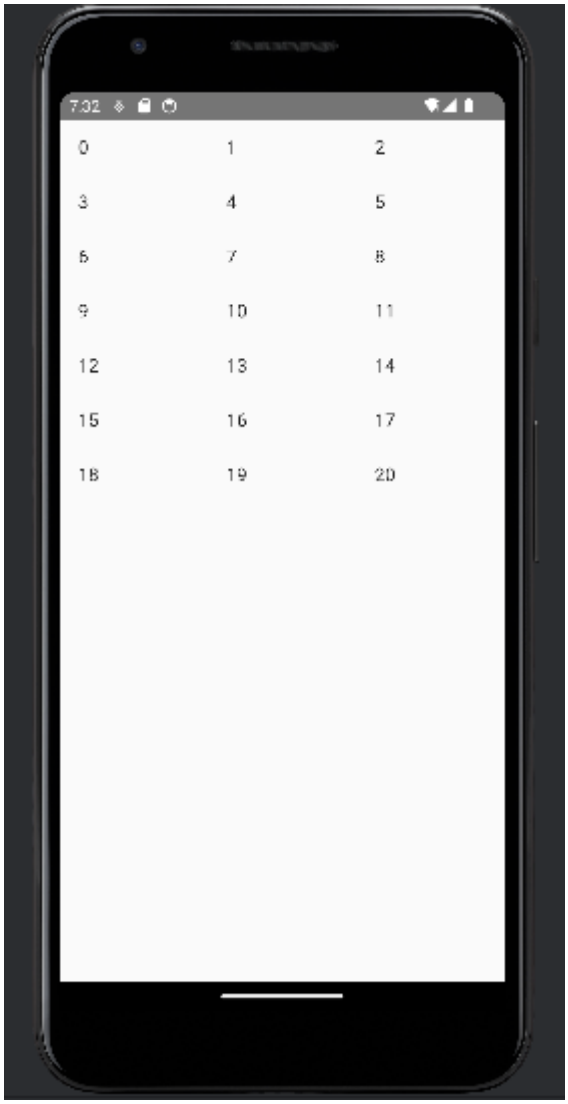
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <GridView
        android:id="@+id/gridView"
        android:numColumns="3"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Mainactivity.kt:

```
import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.GridView
import android.widget.ListView
import android.widget.TextView
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.layout.ui.theme.LayoutTheme

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        //Find View By Id for GridView
        val gridView = findViewById<GridView>(R.id.gridView) as GridView

        //Define ArrayList
        var arrayList = ArrayList<Int>()

        //While Loop to Insert value in ArrayList
        var i = 0
        while (i <= 20) {
            arrayList.add(i)
            i++
        }

        //Create Adapter

        val myAdapter: ArrayAdapter<Int> = ArrayAdapter(this@MainActivity,
```

```
android.R.layout.simple_list_item_1, arrayList)

        //Set Adapter
        gridView.adapter = myAdapter

        //GridView On Click

        gridView.setOnItemClickListener { adapterView, view, position, id
->

            Toast.makeText(this@MainActivity, "You have Click " +
arrayList.get(position), Toast.LENGTH_LONG).show()
        }

    }
}
```

Output:



Practical 5 : Menu with Action Bar

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

How to define Menu in XML File?

Android Studio provides a standard XML format for the type of menus to define menu items. We can simply define the menu and all its items in XML menu resource instead of building the menu in the code and also load menu resource as menu object in the activity or fragment used in our android application. Here, we should create a new folder menu inside of our project directory (res/menu) to define the menu and also add a new XML file to build the menu with the following elements. Below is the example of defining a menu in the XML file (menu_example.xml).

Android Different Types of Menus

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

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

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

Way to create menu directory and menu resource file:

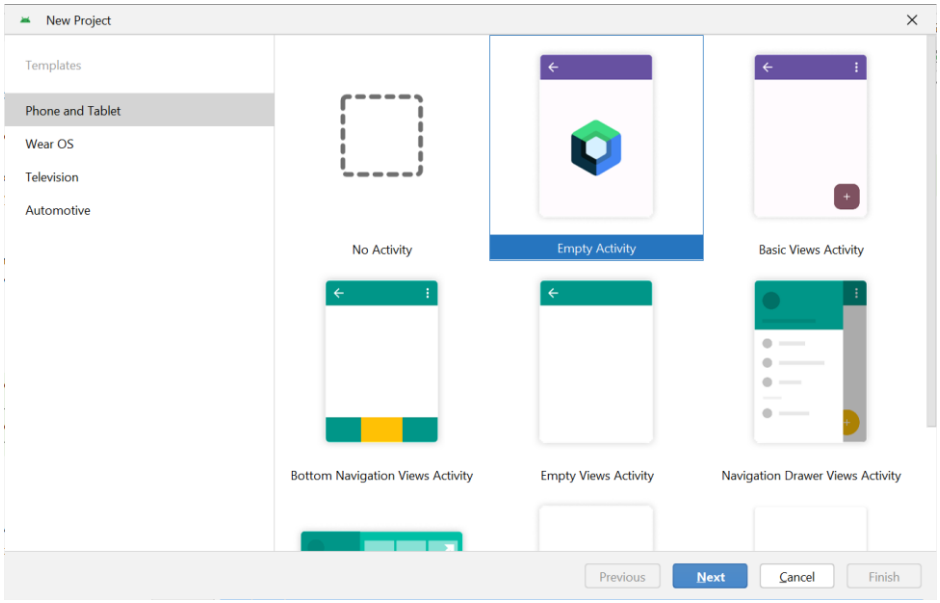
To create the menu directory just right-click on res folder and navigate to res->New->Android Resource Directory. Give resource directory name as menu and resource type also menu. one directory will be created under res folder.

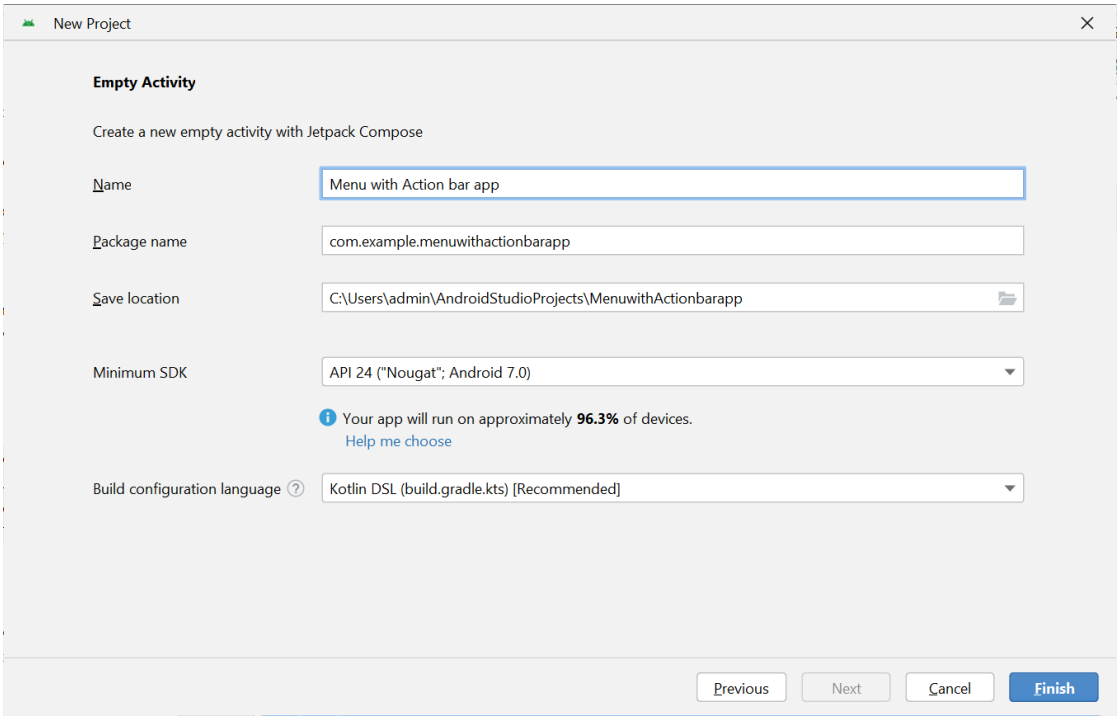
To create xml resource file simply right-click on menu folder and navigate to New->Menu Resource File. Give name of file as menu_example. One menu_example.xml file will be created under menu folder.

- <menu> It is the root element that helps in defining Menu in XML file and it also holds multiple elements.
- <item> It is used to create a single item in the menu. It also contains nested <menu> element in order to create a submenu.
- <group> It is optional and invisible for <item> elements to categorize the menu items so they can share properties like active state, and visibility.

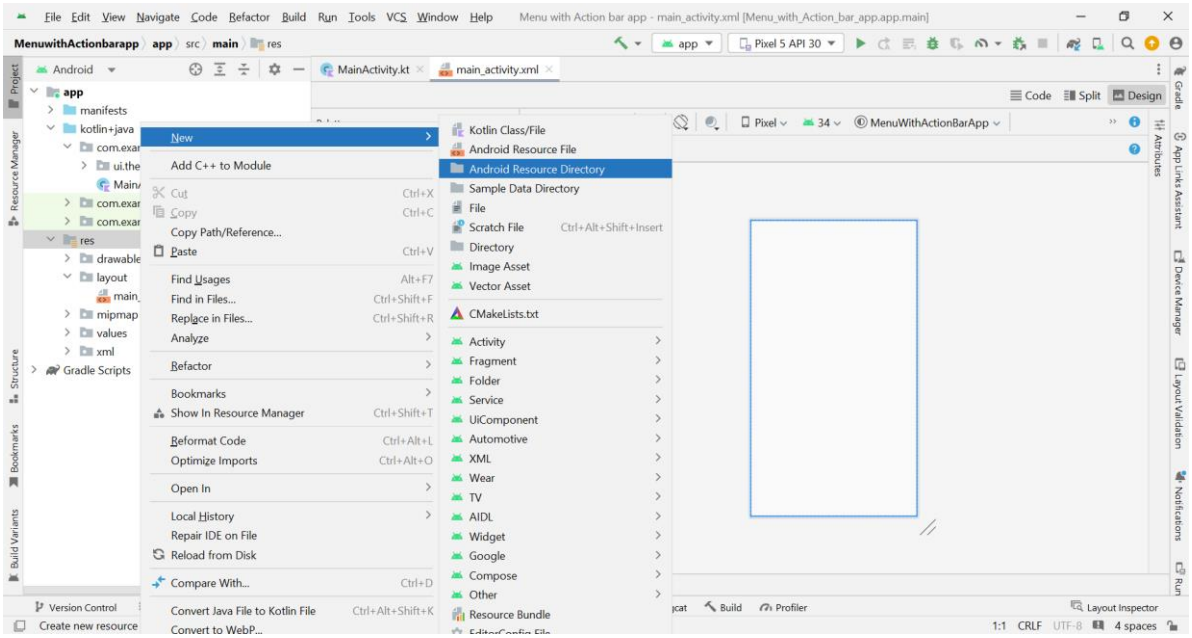
Demonstratting Options Menu

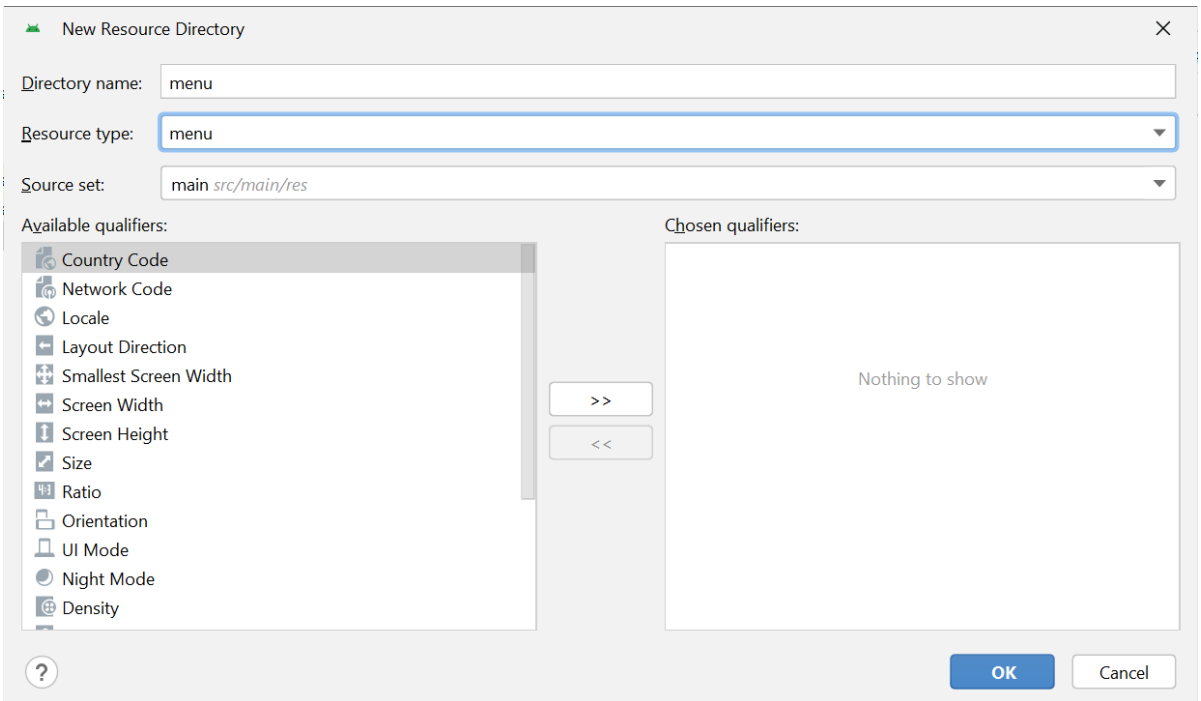
Step 1: Create an android project with empty activity and give name of your preference and add a main_activity.xml file to it.



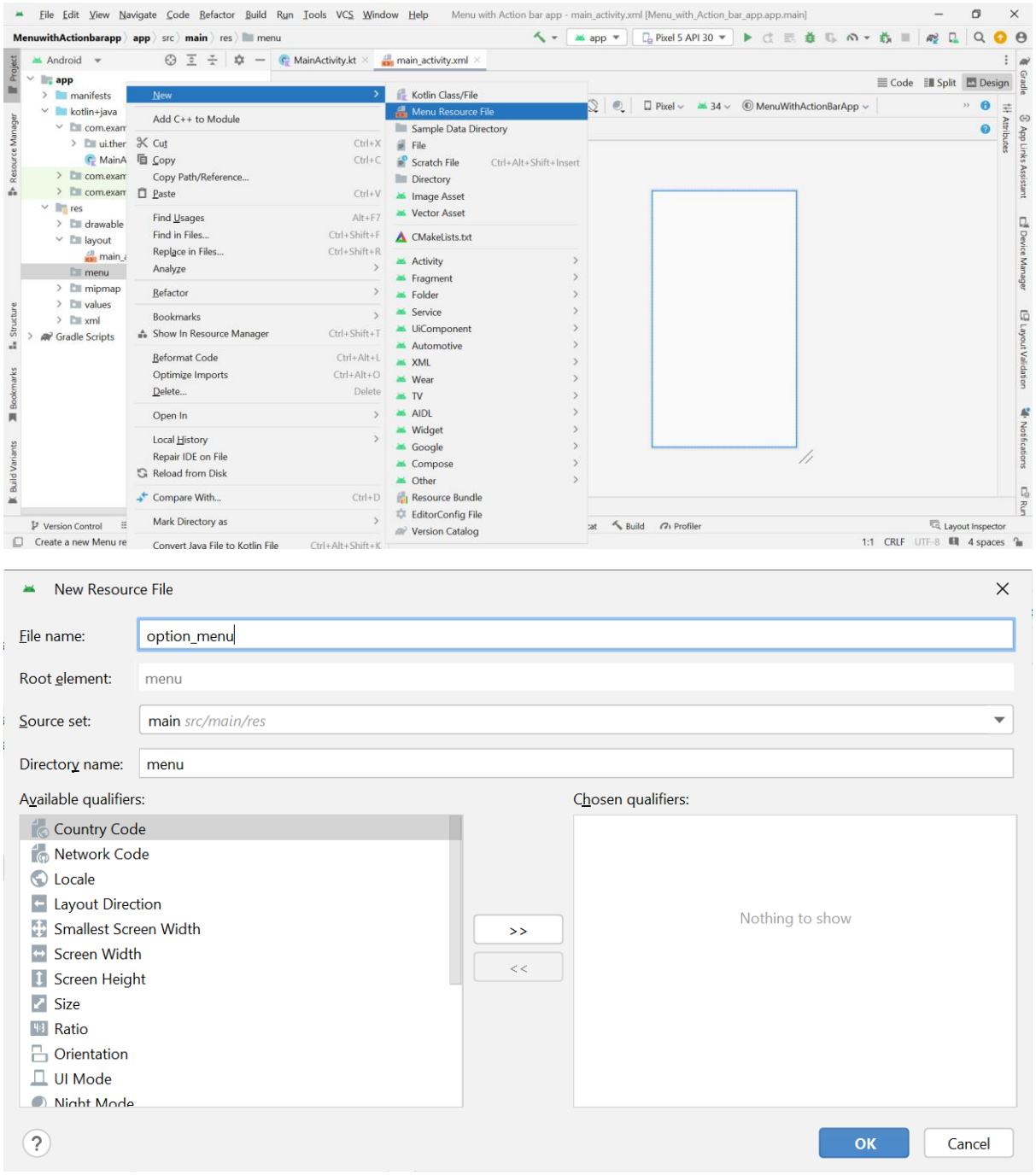


Step 2: Now we have to create a directory for menu, the steps are same as that which are use to create layout directory. Only difference is while selecting the resource type while creating directory select **menu** from the dropdown.

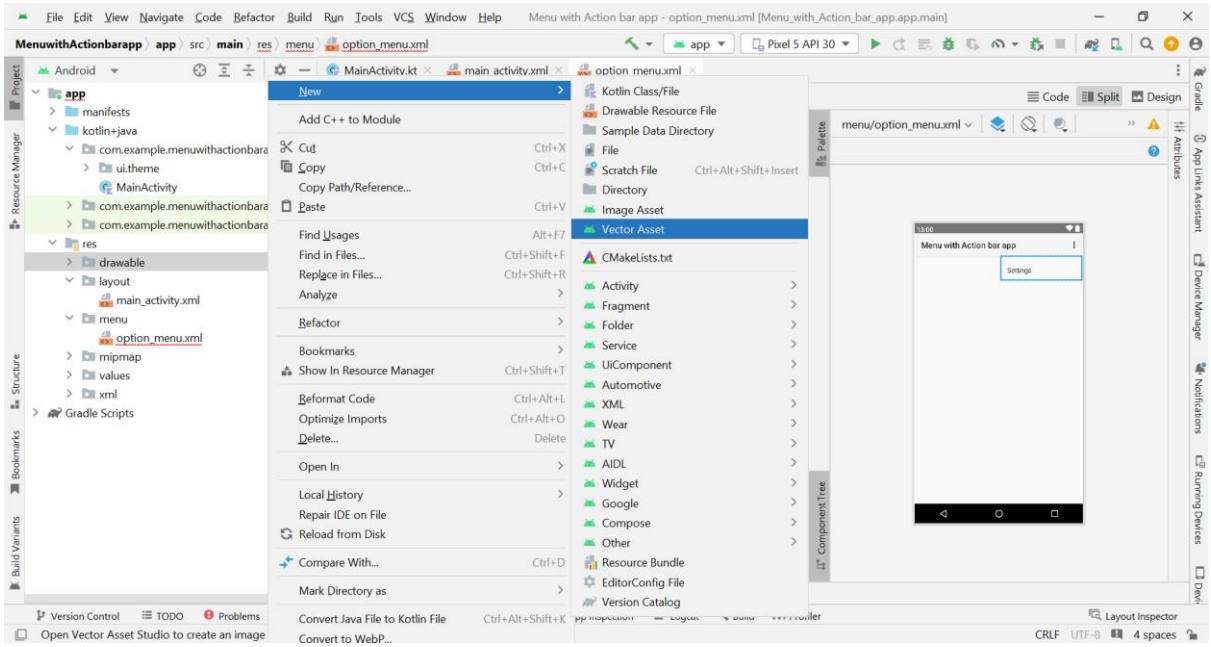


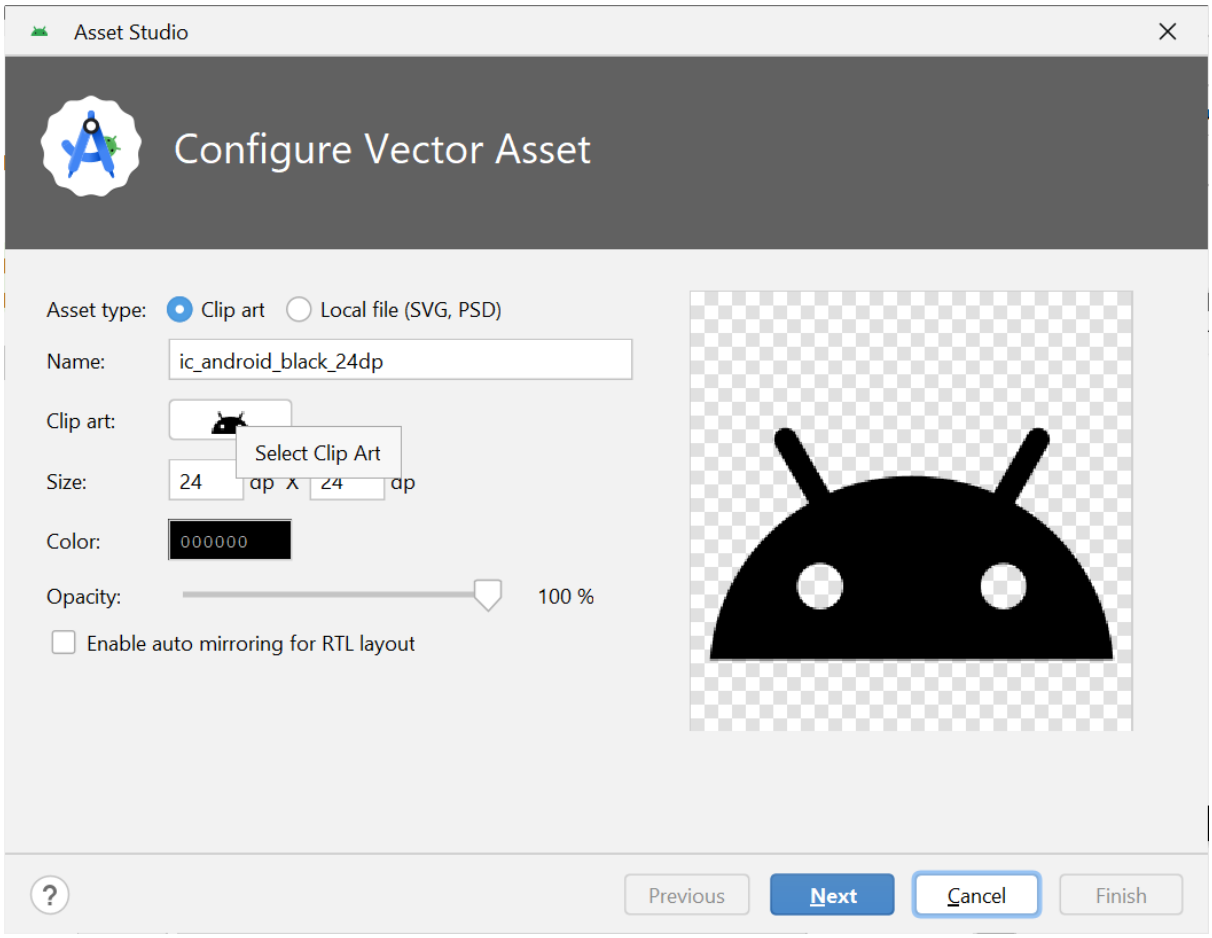


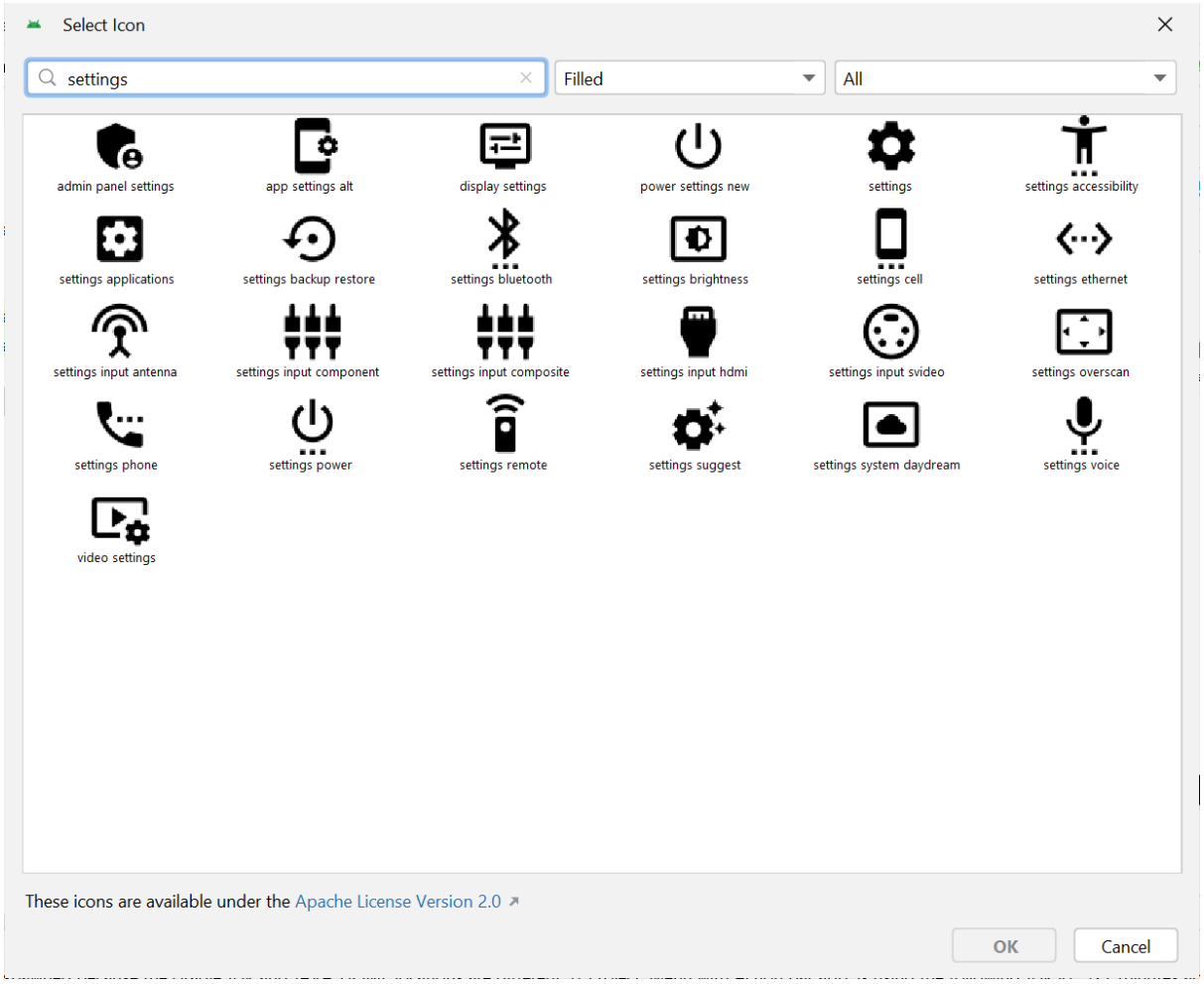
Step 3: Once directory is created add Menu Resource file to the newly created menu folder.

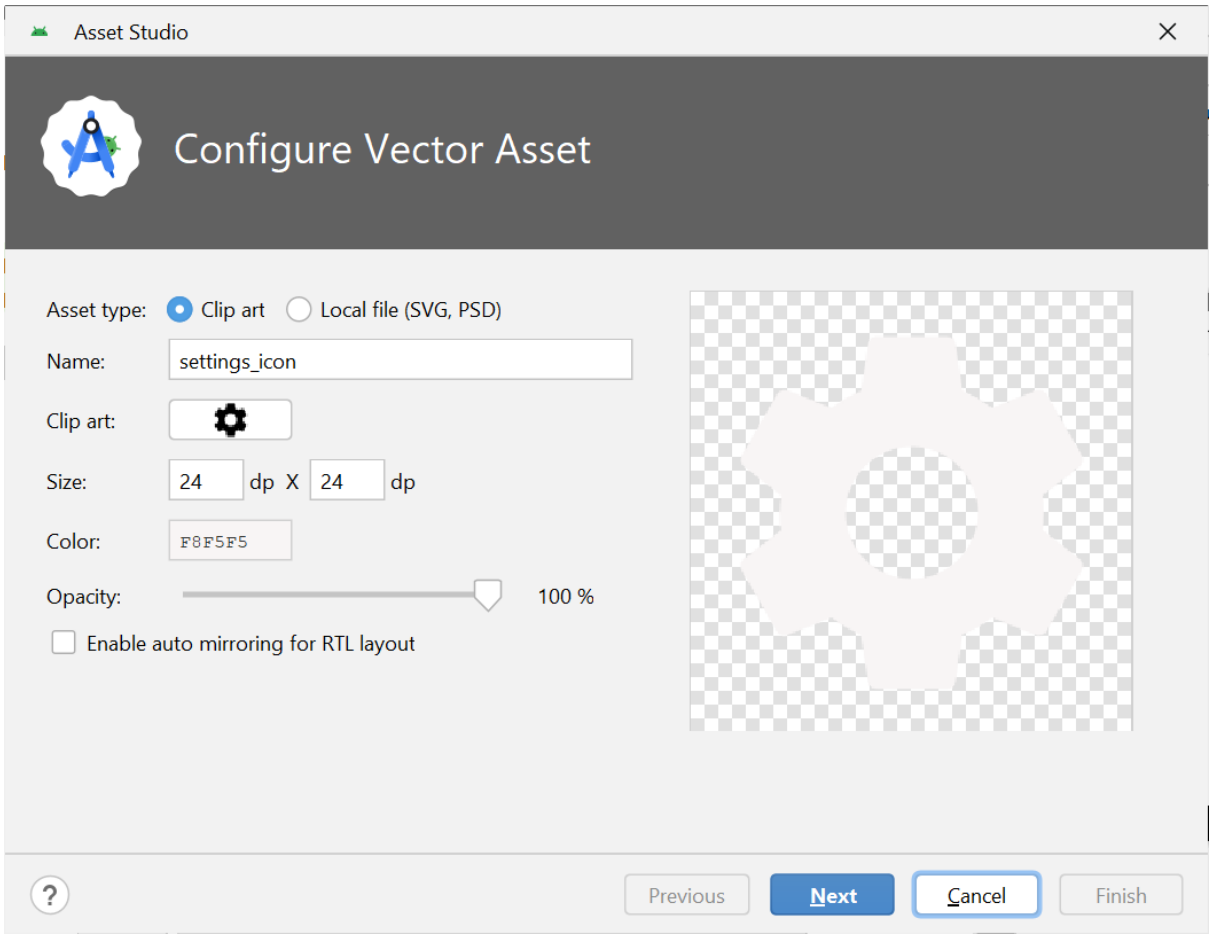


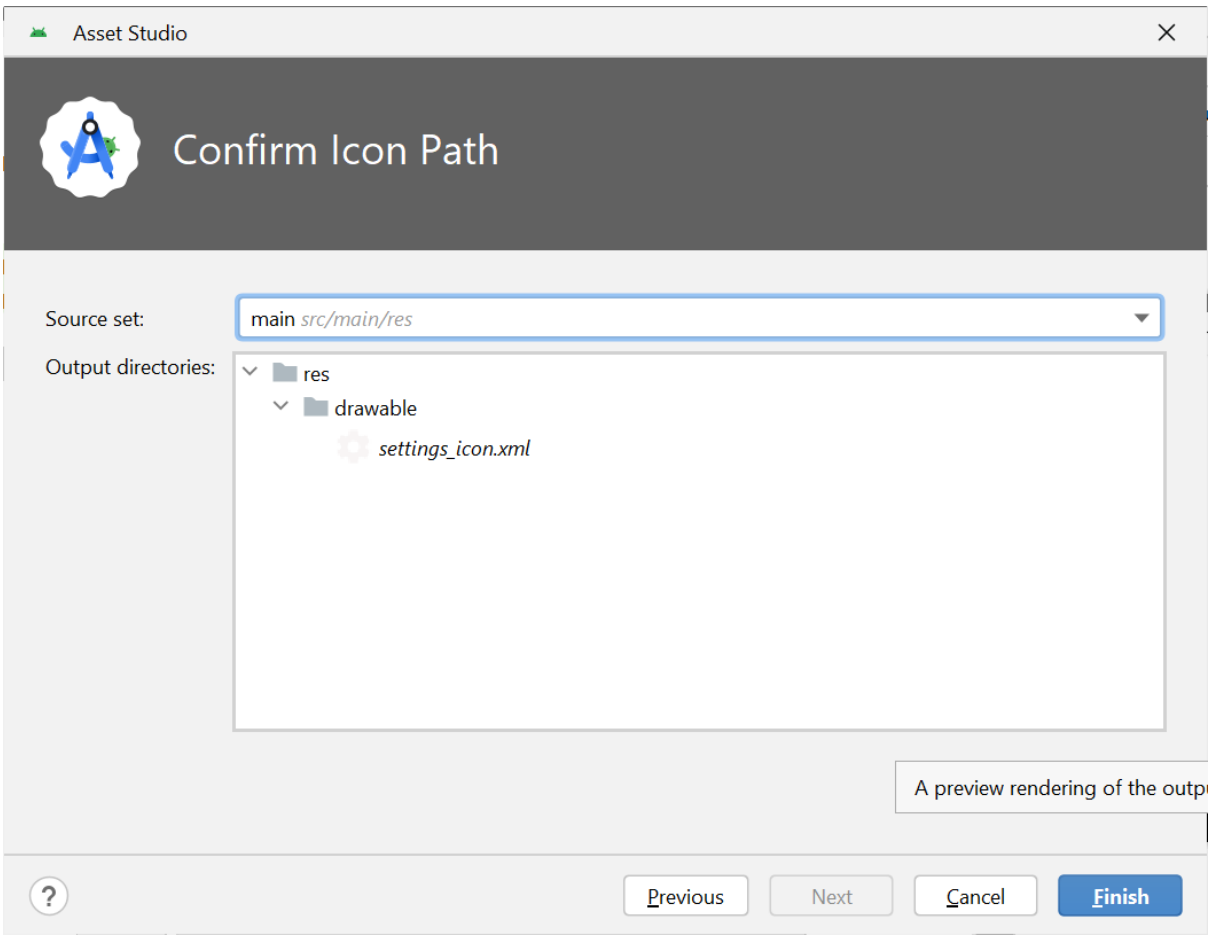
Step 4: To add new vector image to drawable folder, follow the steps below.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Menu with Action bar app - option_menu.xml [Menu_with_Action_bar_app.app.main]
MenuwithActionbarapp app src main res menu option_menu.xml
app
Pixel 5 API 30
Android
MainActivity.kt
main activity.xml
option menu.xml
app
manifests
kotlin+java
com.example.menuwithactionbara
ui.theme
MainActivity
com.example.menuwithactionbara
com.example.menuwithactionbara
res
drawable
layout
main_activity.xml
menu
option_menu.xml
mipmap
values
xml
Gradle Scripts
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Convert to WebP...
Kotlin Class/File
Drawable Resource File
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
Image Asset
Vector Asset
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
TV
AIDL
Widget
Google
Compose
Other
Resource Bundle
EditorConfig File
Version Catalog
Code Split Design
menu/option_menu.xml
Menu with Action bar app
Settings
Version Control TODO Problems
Open Vector Asset Studio to create an image
Layout Inspector
CRLF UTF-8 4 spaces


Asset Studio
Configure Vector Asset
Asset type: Clip art Local file (SVG, PSD)
Name: ic_android_black_24dp
Clip art:
Select Clip Art
Size: 24 dp x 24 dp
Color: 000000
Opacity: 100 %
Enable auto mirroring for RTL layout
Previous
Next
Cancel
Finish


Select Icon
settings
Filled
All
admin panel settings
app settings alt
display settings
power settings new
settings
settings accessibility
settings applications
settings backup restore
settings bluetooth
settings brightness
settings cell
settings ethernet
settings input antenna
settings input component
settings input composite
settings input hdmi
settings input svideo
settings overscan
settings phone
settings power
settings remote
settings suggest
settings system daydream
settings voice
video settings
These icons are available under the Apache License Version 2.0
OK
Cancel


Asset Studio
Configure Vector Asset
Asset type:
Clip art
Local file (SVG, PSD)
Name:
settings_icon
Clip art:
Size:
24
dp X
24
dp
Color:
F8F5F5
Opacity:
100 %
Enable auto mirroring for RTL layout
Previous
Next
Cancel
Finish



Step 5: Add the below code in options_menu.xml file

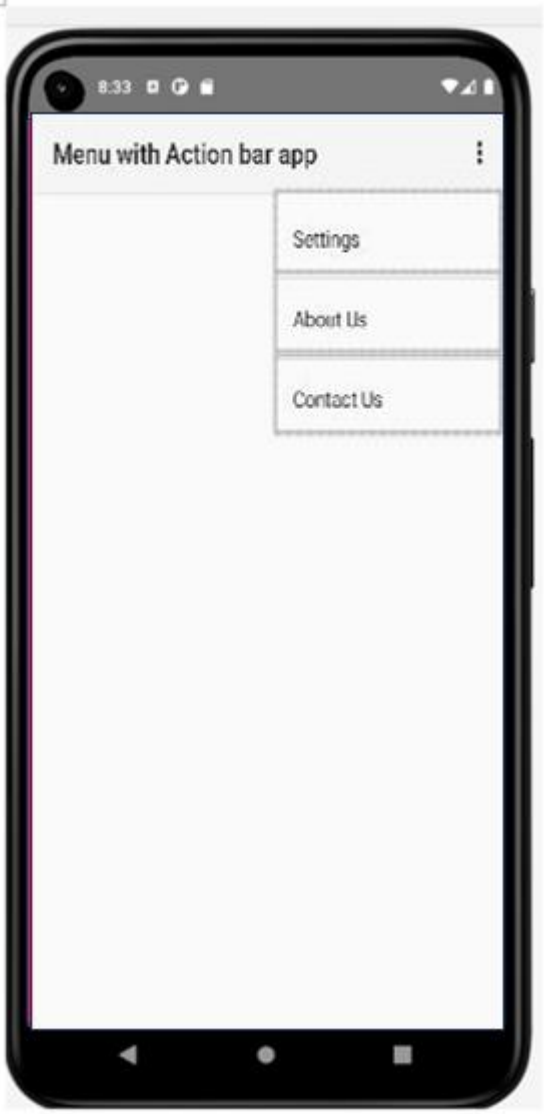
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/settings"
        android:title="Settings"
        android:icon="@drawable/settings_icon"/>
    <item android:id="@+id/about"
        android:title="About Us"
        android:icon="@drawable/info_icon"/>
    <item android:id="@+id/contact"
        android:title="Contact Us"
        android:icon="@drawable/contact_icon"/>

</menu>
```

Step 6: Add the following in MainActivity.kt file:

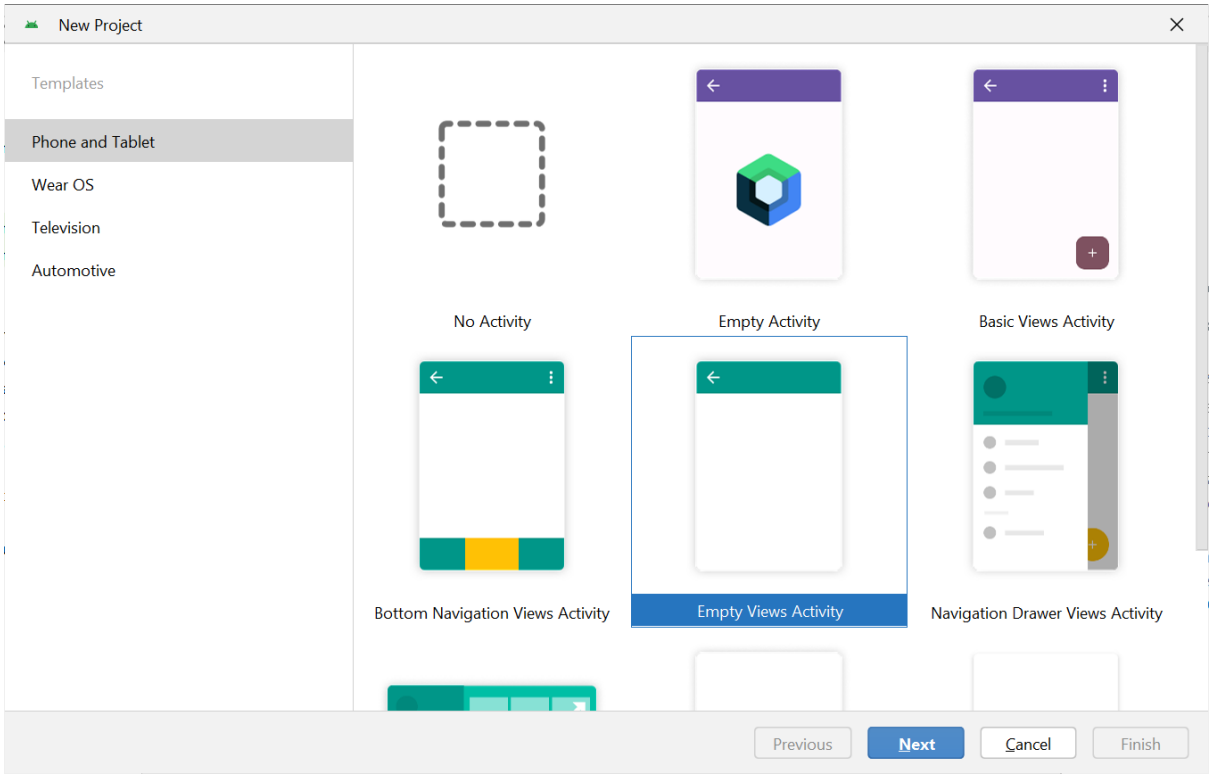
```
package com.example.menuwithactionbarapp

import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
```

```
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.menuwithactionbarapp.ui.theme.MenuWithActionBarAppTheme

class MainActivity<MenuBuilder> : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {

        menuInflater.inflate(R.menu.option_menu,menu)
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when(item.itemId){
            R.id.settings ->{
                Toast.makeText(this,"Settings
Clicked",Toast.LENGTH_LONG).show()
            }
            R.id.about ->{
                Toast.makeText(this,"About Us
Clicked",Toast.LENGTH_LONG).show()
            }
            R.id.contact ->{
                Toast.makeText(this,"Contact Us
Clicked",Toast.LENGTH_LONG).show()
            }
        }
        return super.onOptionsItemSelected(item)
    }
}
```

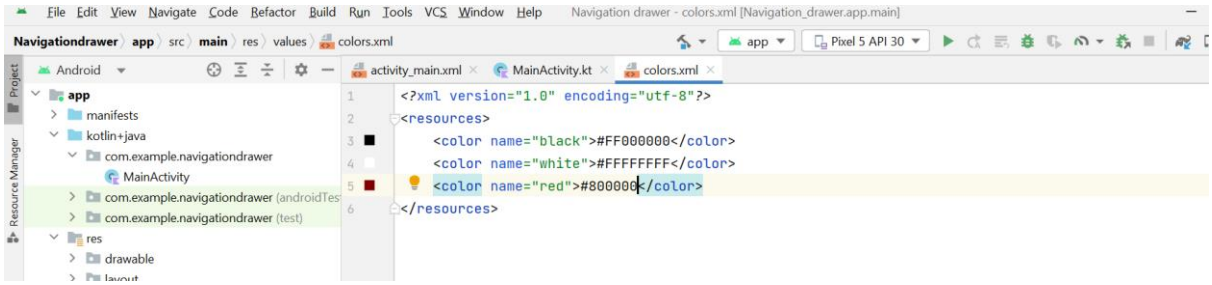
Output:


8:33
Menu with Action bar app
Settings
About Us
Contact Us

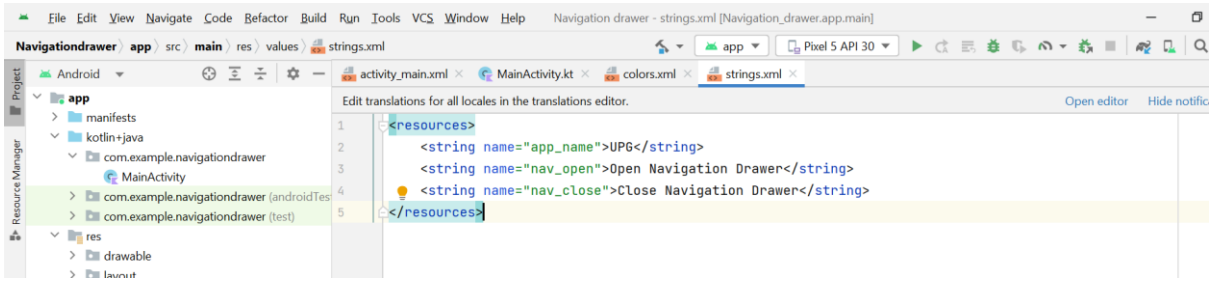
Practical 6: Navigation drawer and Bottom Navigation

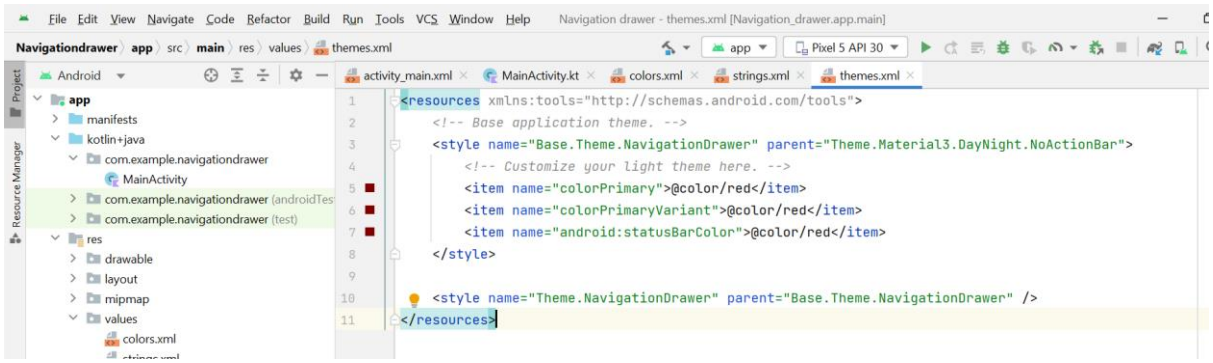


Goto colors.xml file and add few colors.



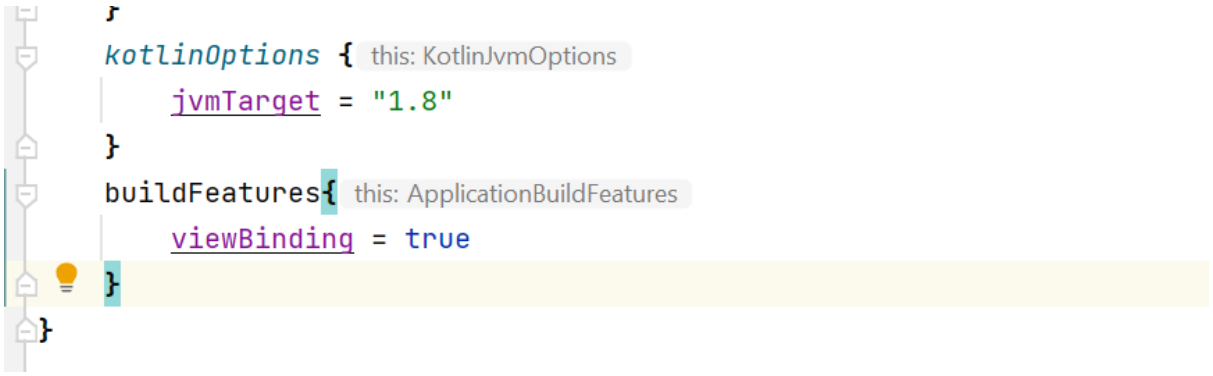
Goto strings.xml file and add few strings:



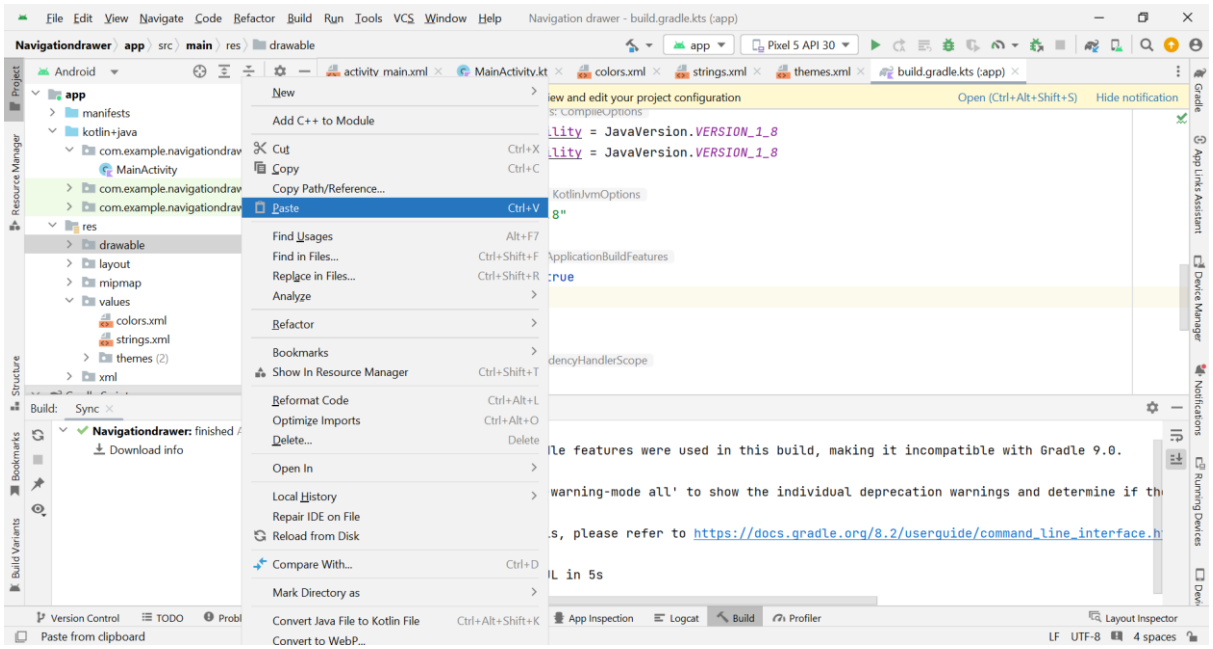


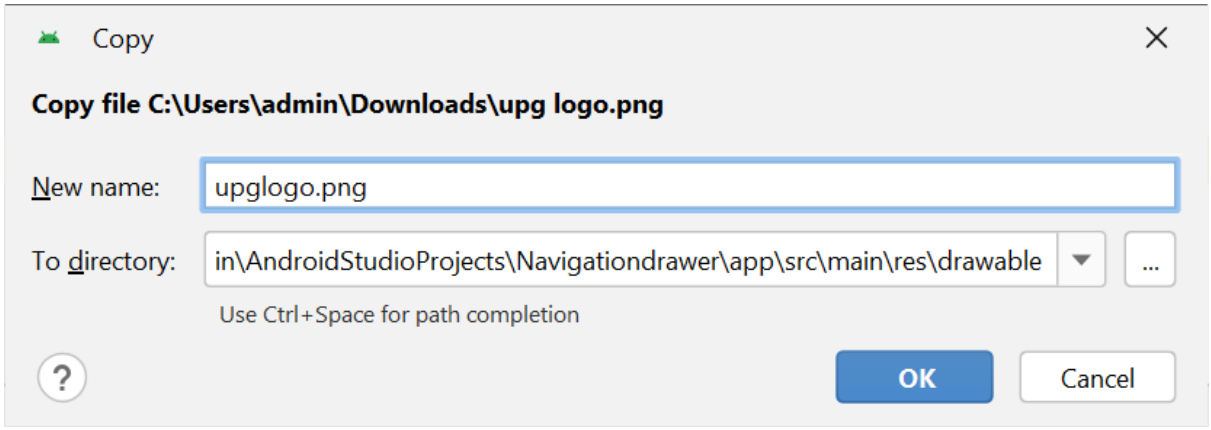
Now goto build.gradle and enable view binding feature by add the code mentioned below in buildFeatures function

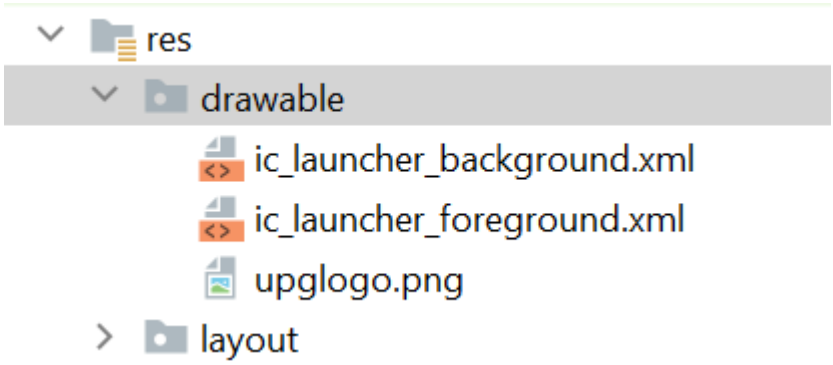
```
viewBinding = true
```



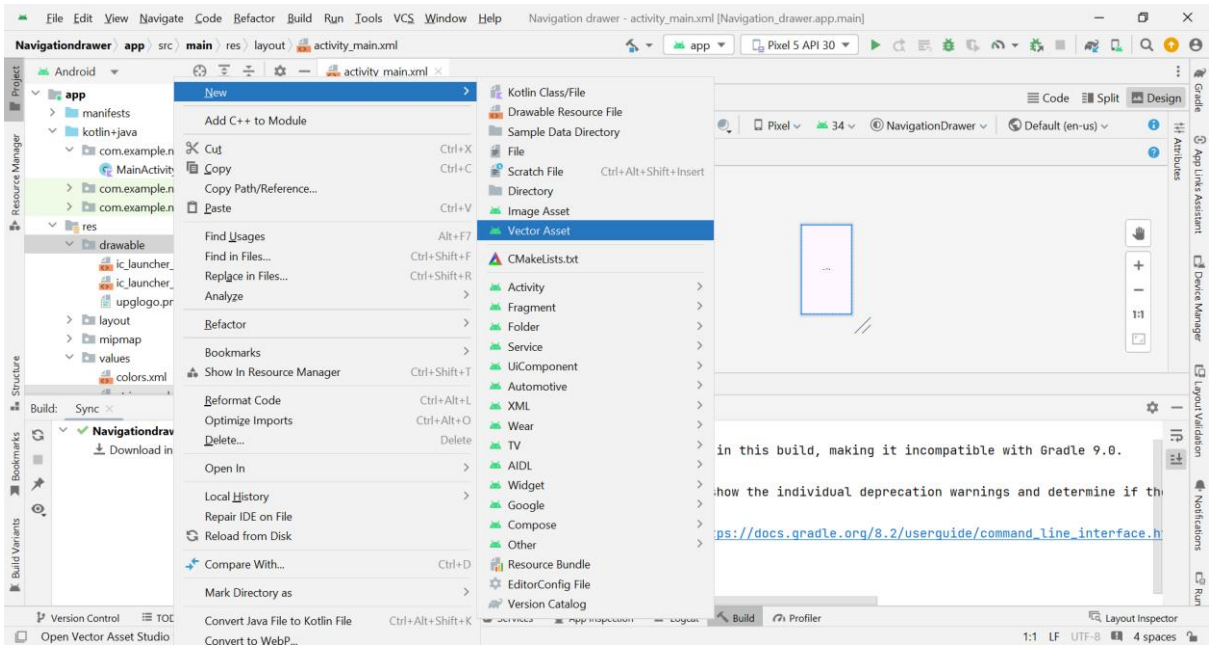
Now we need to add 11 vector images and 1 image in the project so we first copy paste the image

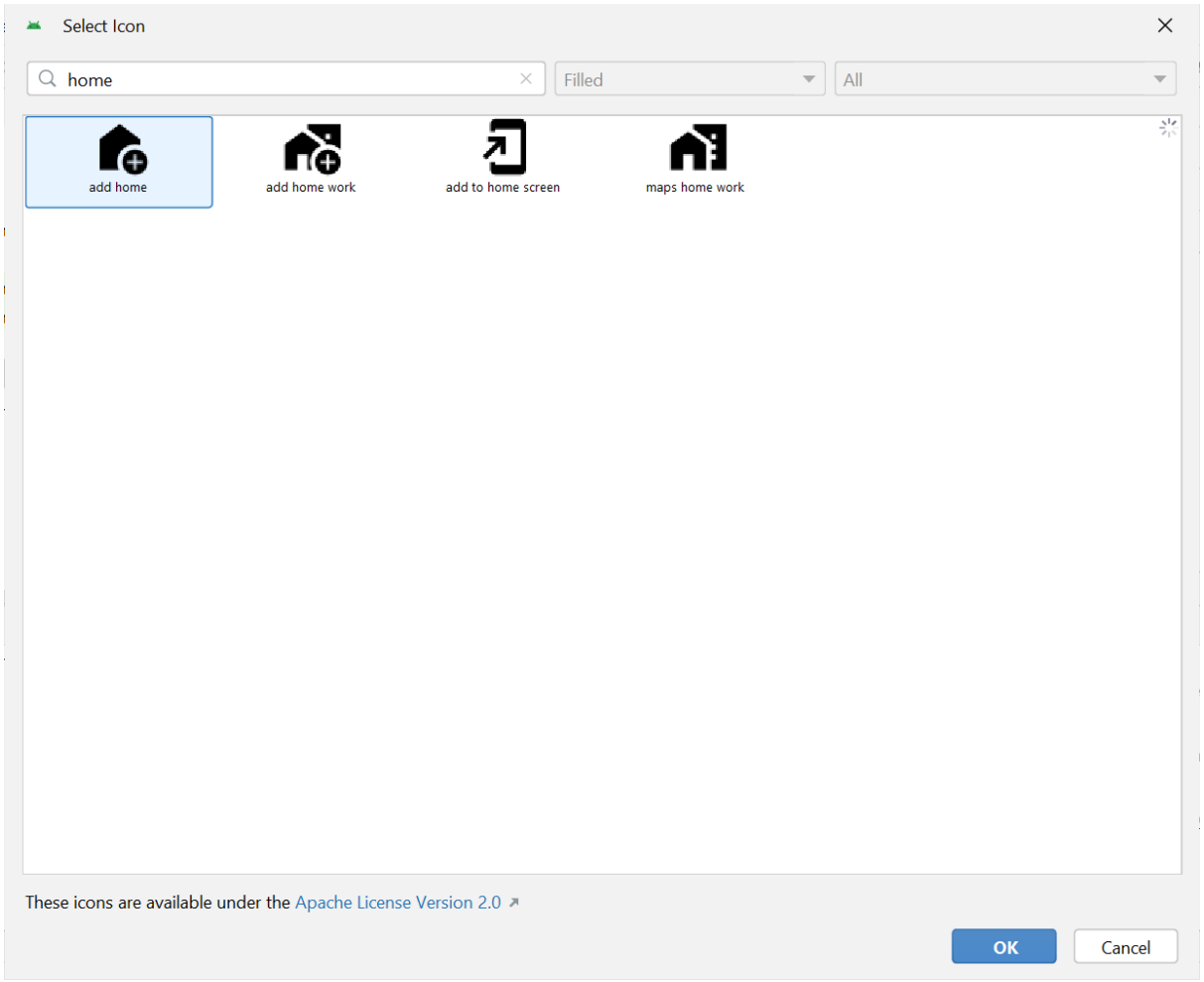


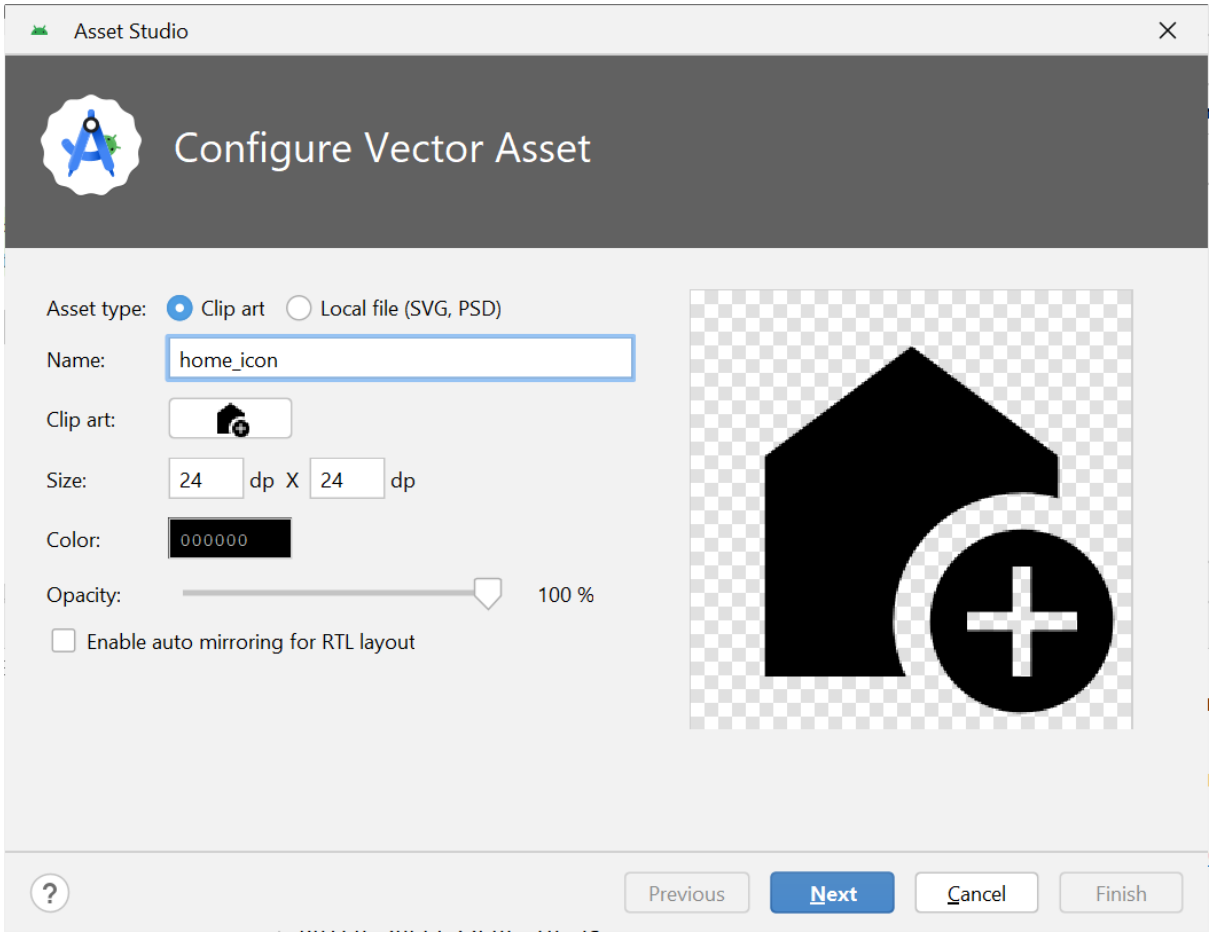


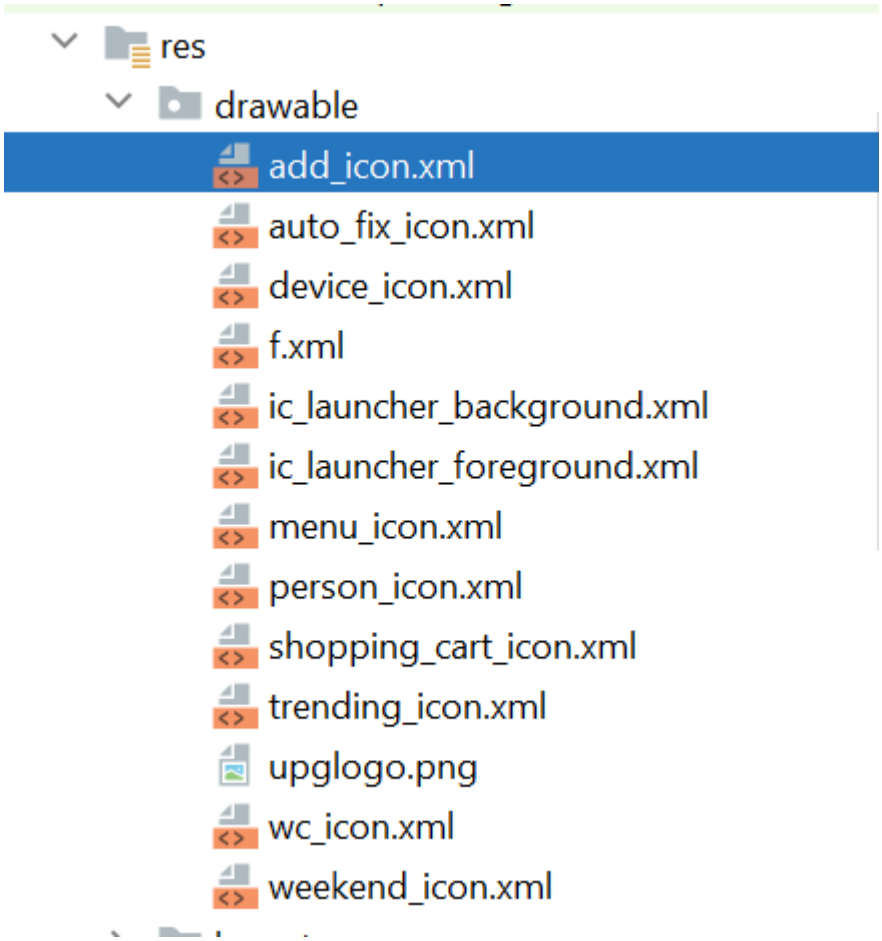


Now we will go on to add vector asset

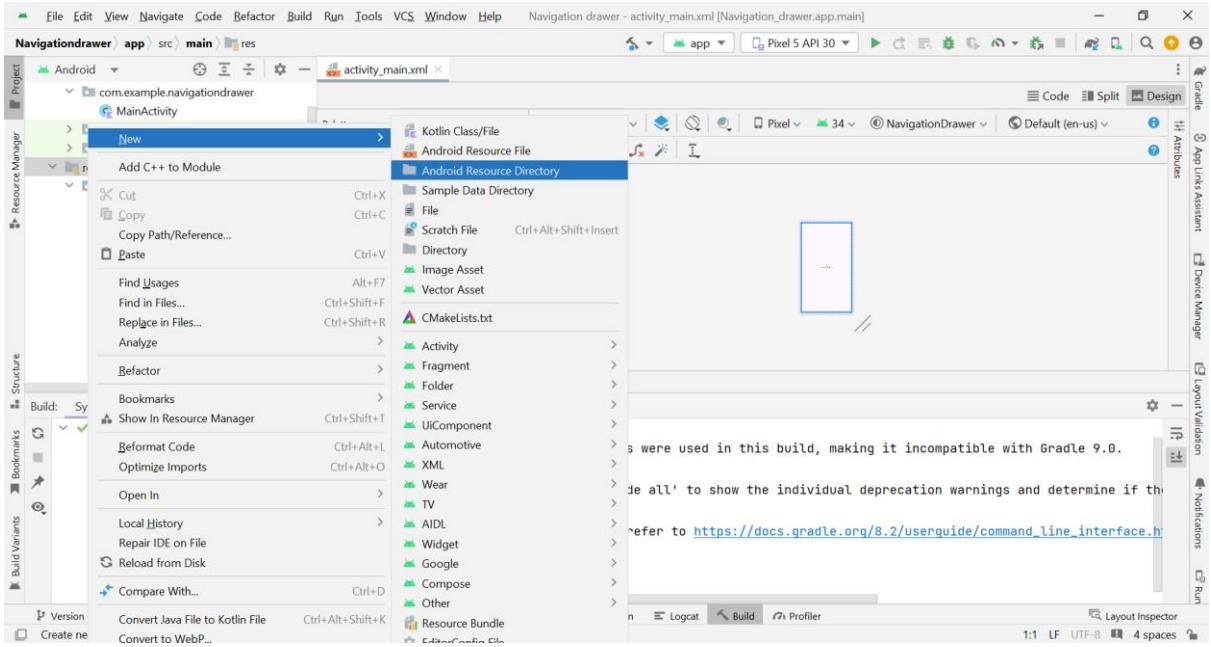


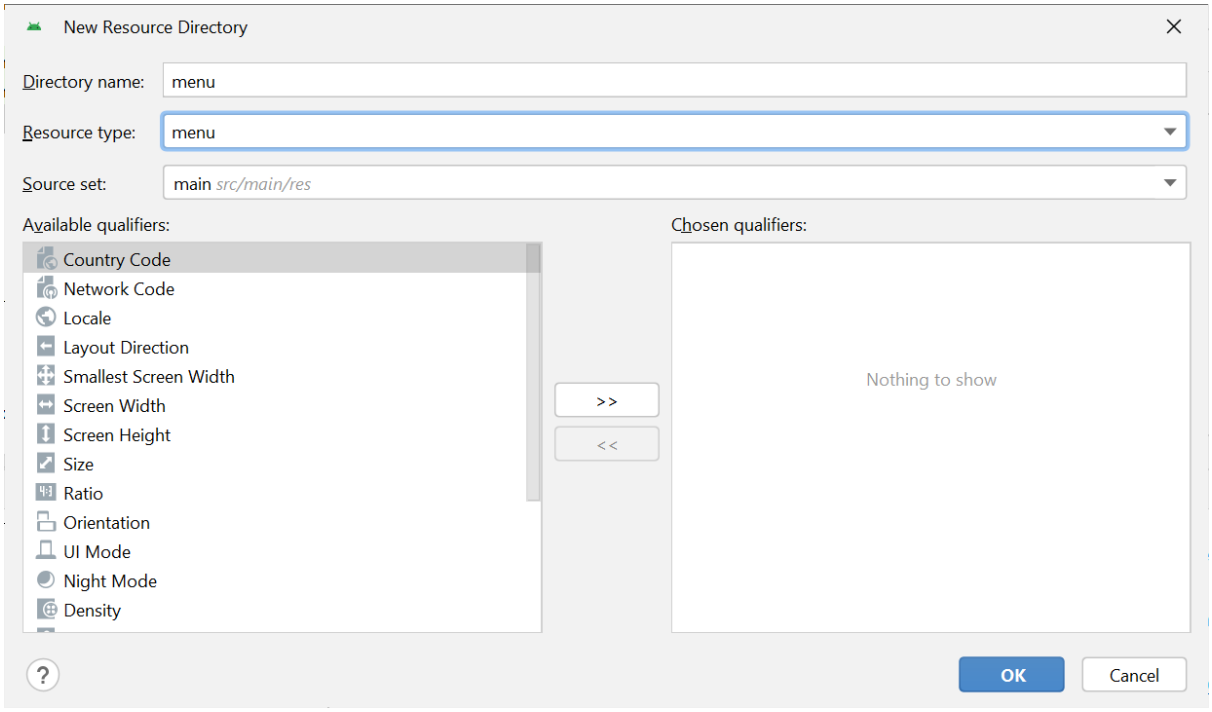

Select Icon
home
Filled
All
add home
add home work
add to home screen
maps home work
These icons are available under the Apache License Version 2.0
OK
Cancel


Asset Studio
Configure Vector Asset
Asset type:
Clip art
Local file (SVG, PSD)
Name:
home_icon
Clip art:
Size:
24
dp X
24
dp
Color:
000000
Opacity:
100 %
Enable auto mirroring for RTL layout
Previous
Next
Cancel
Finish

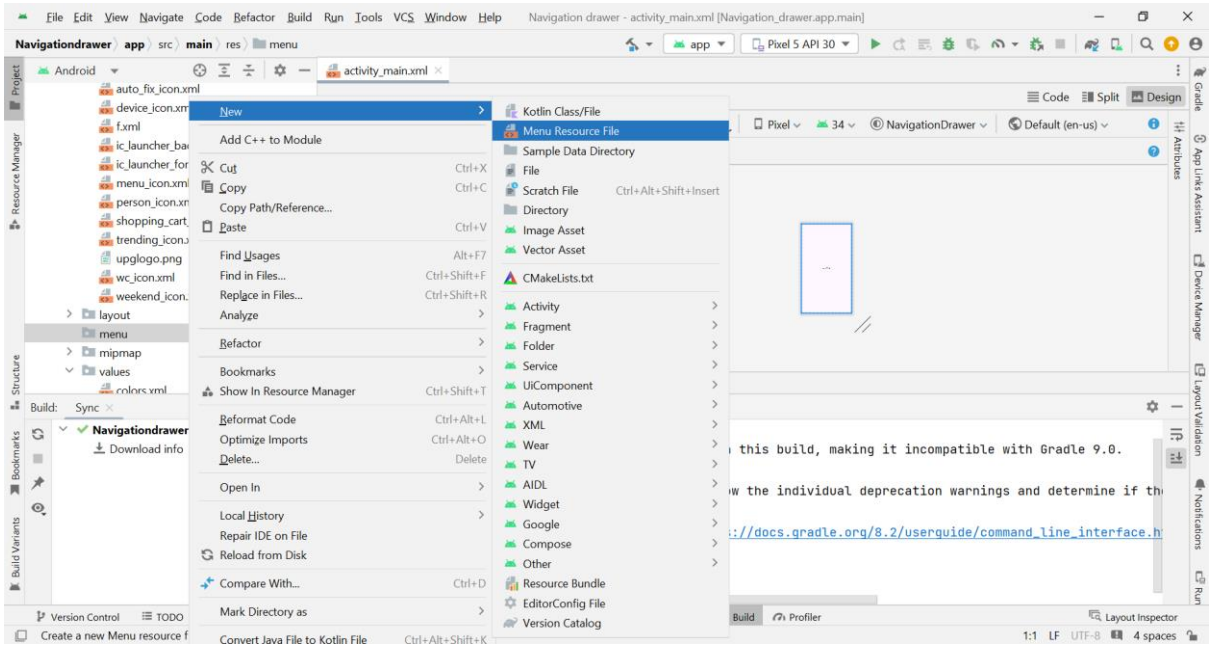


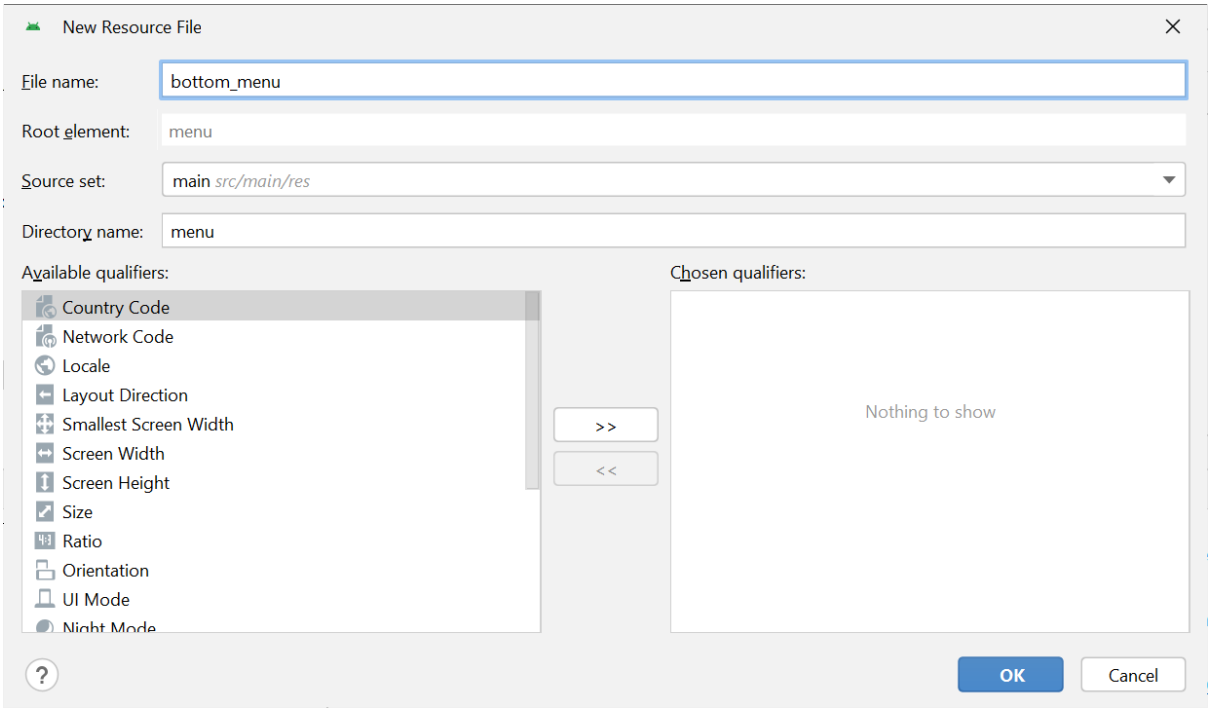
Now we create a dedicated menu directory





Now we create menu for bottom

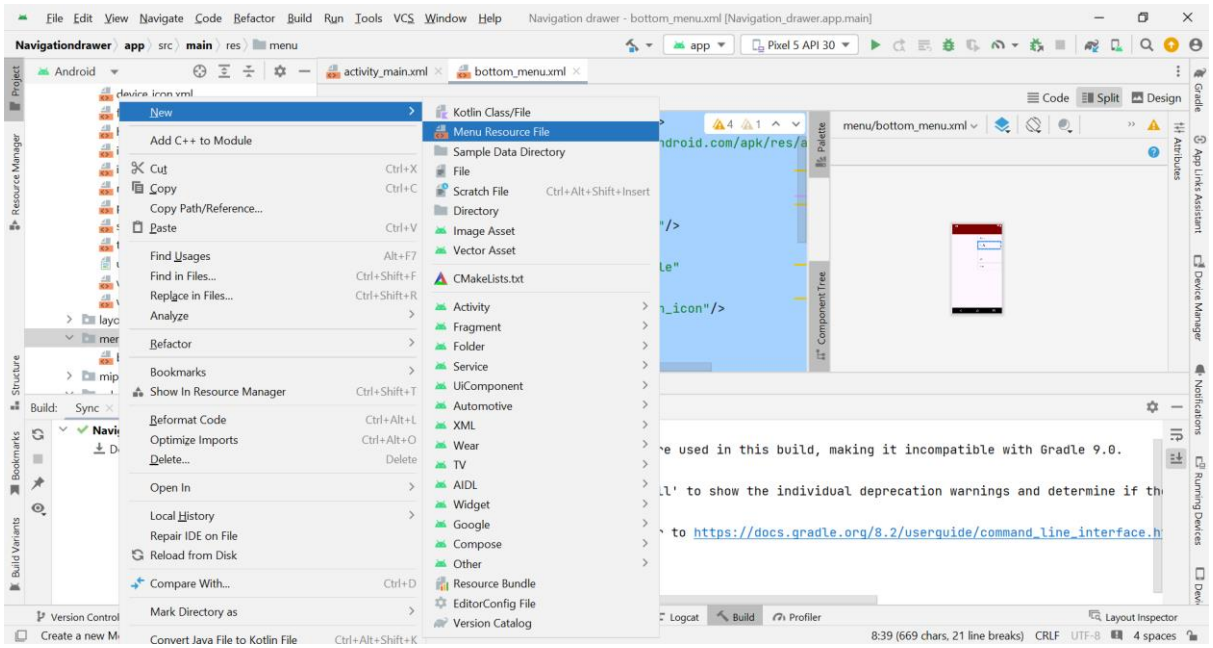


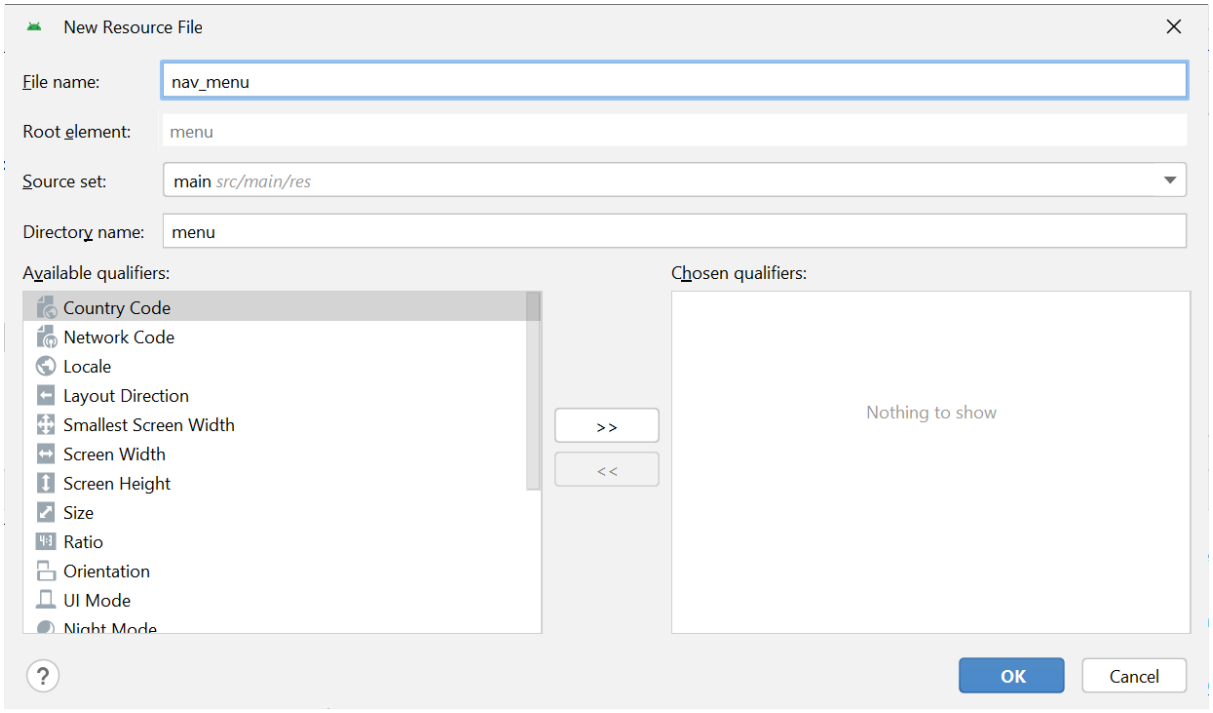


Add the following code

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
    android:id="@+id/bottom_home"
    android:title="Home"
    android:icon="@drawable/home_icon"/>
    <item
        android:id="@+id/bottom_profile"
        android:title="Profile"
        android:icon="@drawable/person_icon"/>
    <item
        android:title=""
        android:enabled="true"/>
    <item
        android:id="@+id/bottom_cart"
        android:title="Cart"
        android:icon="@drawable/shopping_cart_icon"/>
    <item
        android:id="@+id/bottom_menu"
        android:title="Menu"
        android:icon="@drawable/menu_icon"/>
</menu>
```

Now we will add navigation drawer

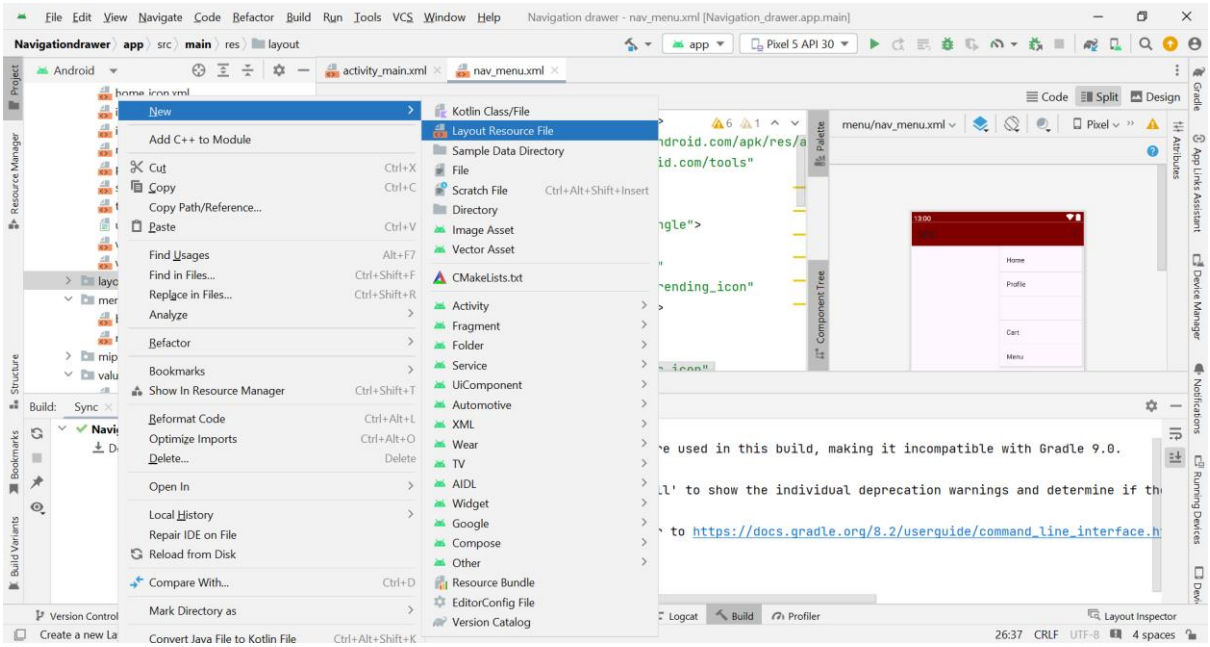


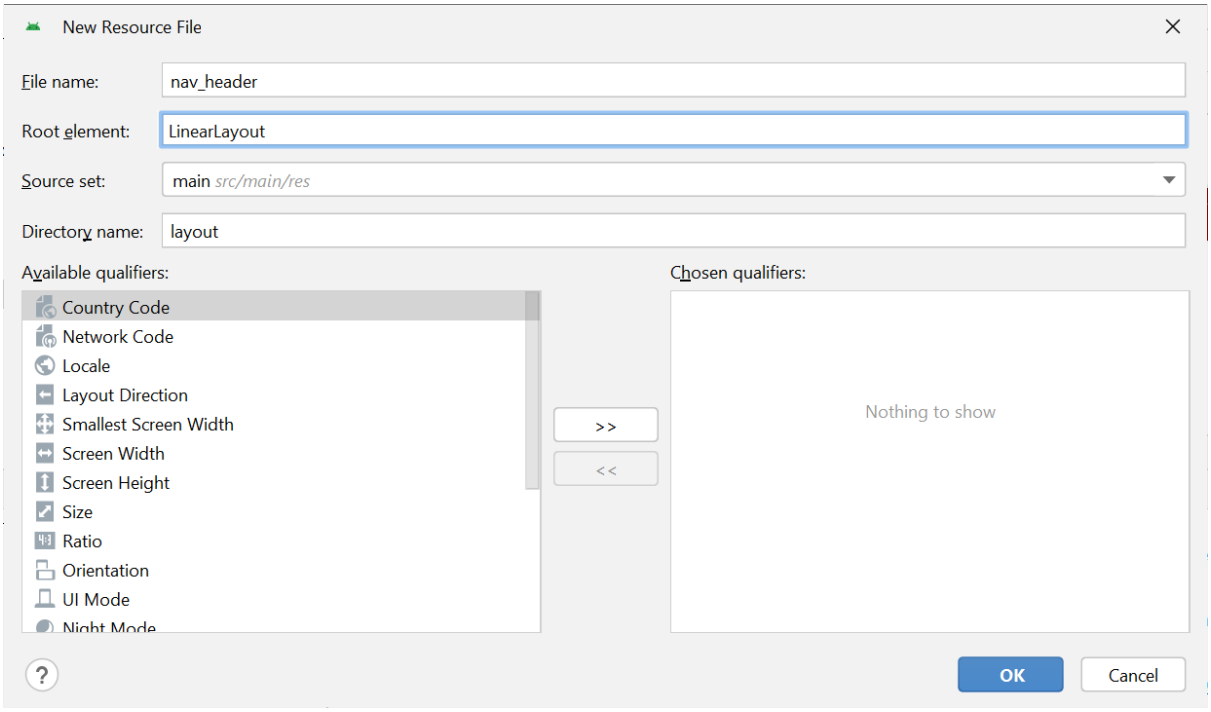



```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:ShowIn = "navigation_view">
    <group
        android:checkableBehavior="single">
        <item
            android:id="@+id/trending"
            android:icon="@drawable/trending_icon"
            android:title="Trending"/>
        <item
            android:id="@+id/fashion"
            android:icon="@drawable/wc_icon"
            android:title="Fashion"/>
        <item
```

```
            android:id="@+id/device"
            android:icon="@drawable/device_icon"
            android:title="Electronics"/>
        <item
            android:id="@+id/fresh"
            android:icon="@drawable/f"
            android:title="Food"/>
        <item
            android:id="@+id/beauty"
            android:icon="@drawable/auto_fix_icon"
            android:title="Beauty"/
        <item
            android:id="@+id/furniture"
            android:icon="@drawable/weekend_icon"
            android:title="Furniture"/>

    </group>


</menu>
```

Now as navigation drawer is ready now we need to create navigation header for that follow the steps below





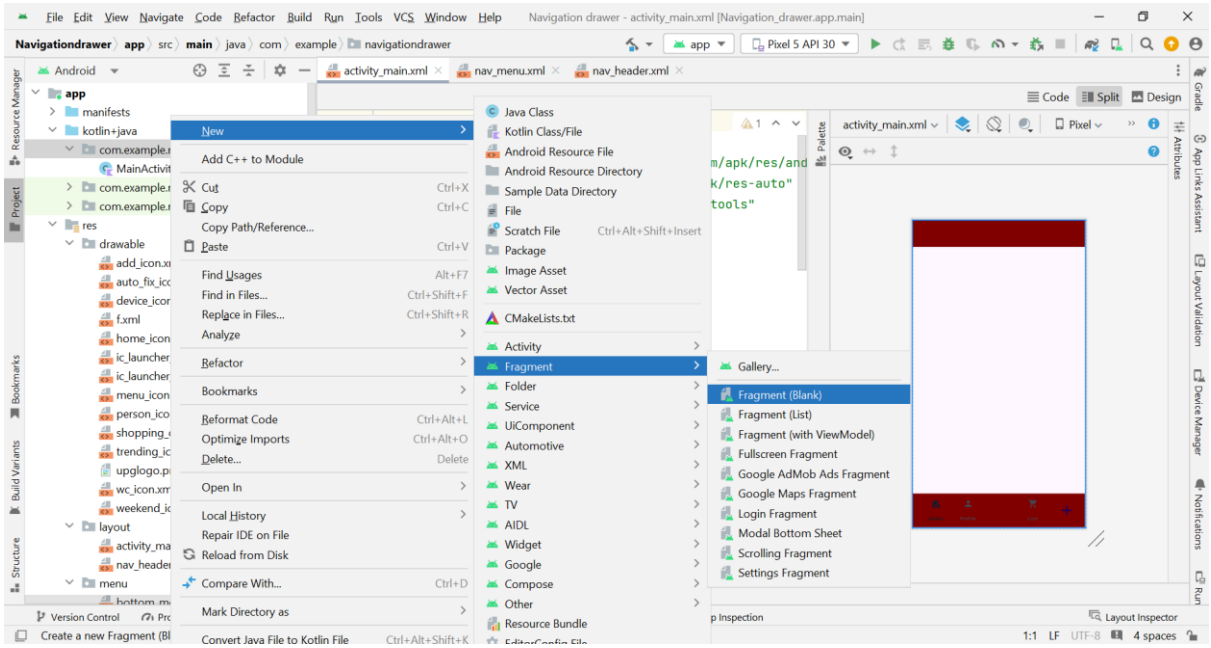

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="176dp"
    android:gravity="top"
    android:padding="16dp"
    android:background="@color/red"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:src="@drawable/upglogo"
        android:scaleType="center"/>

</LinearLayout>
```

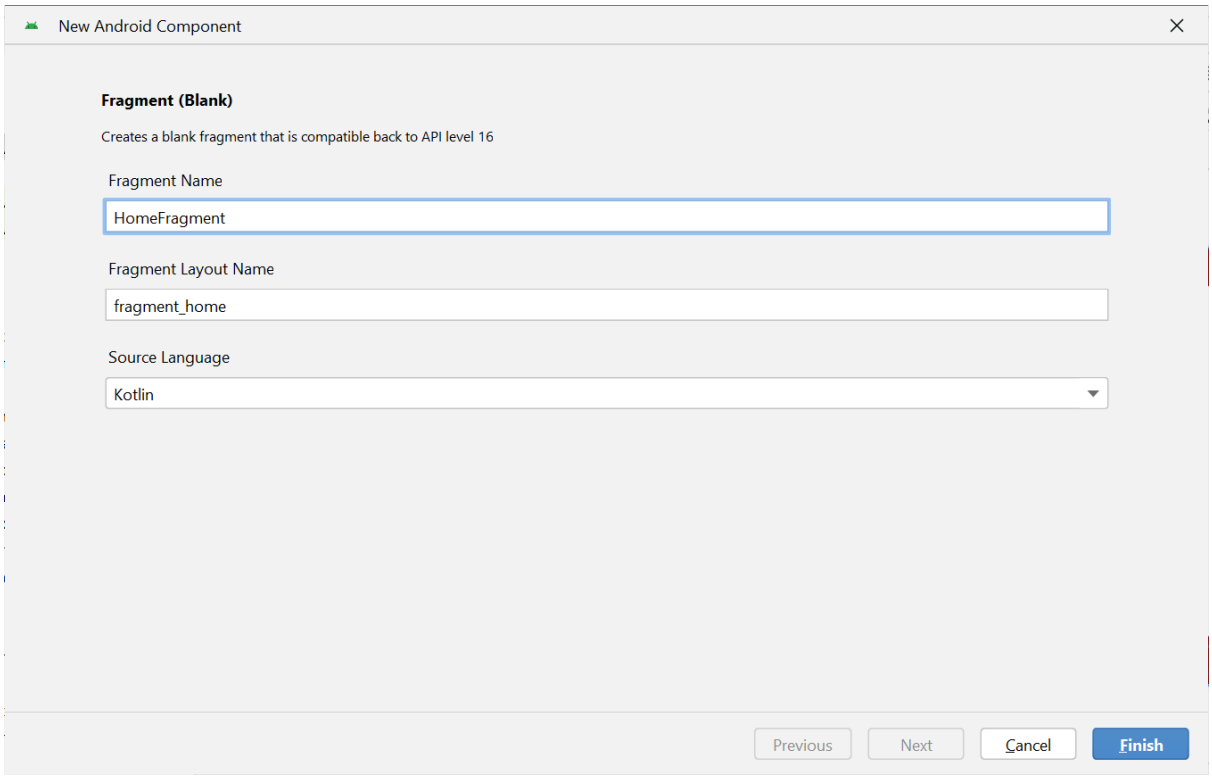
Now add the following code in activity_main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    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:id="@+id/drawer_layout">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <Toolbar
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
```

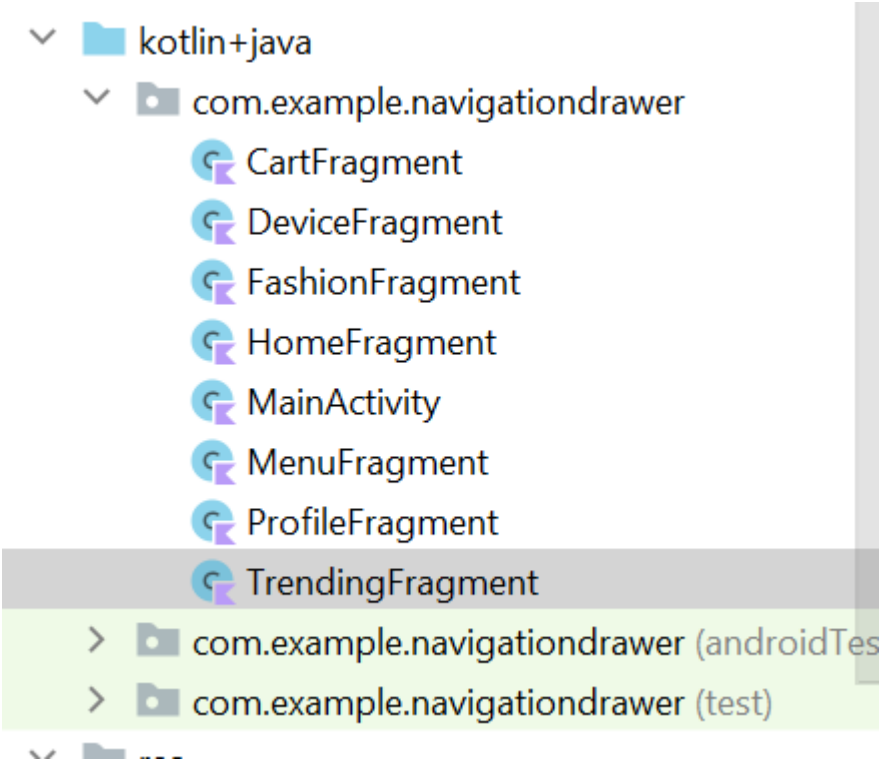
```
            android:id="@+id/toolbar"
            android:background="@color/red"
            android:elevation="4dp"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            android:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>
        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/fragment_container"/>

        <androidx.coordinatorlayout.widget.CoordinatorLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <com.google.android.material.bottomappbar.BottomAppBar
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/bottomAppBar"
                android:layout_gravity="bottom"
                android:backgroundTint="@color/red"
                app:fabCradleMargin="10dp"
                app:fabCradleRoundedCornerRadius="50dp">

<com.google.android.material.bottomnavigation.BottomNavigationView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:id="@+id/bottom_navigation"
                    app:labelVisibilityMode="labeled"
                    app:menu="@menu/bottom_menu"
                    android:background="@android:color/transparent"/>
            </com.google.android.material.bottomappbar.BottomAppBar>

<com.google.android.material.floatingactionbutton.FloatingActionButton
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:backgroundTint="@color/red"
                android:src="@drawable/add_icon"
                app:layout_anchor="@id/bottomAppBar"
                app:maxImageSize="40dp"
                android:id="@+id/fab"/>


        </androidx.coordinatorlayout.widget.CoordinatorLayout>

    </RelativeLayout>
    <com.google.android.material.navigation.NavigationView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/navigation_drawer"
        android:layout_gravity="start"
        app:itemIconTint="@color/red"
        app:itemTextColor="@color/red"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/nav_menu"/>

</androidx.drawerlayout.widget.DrawerLayout>
```

Now we need to create a 7 fragment for the above layout, 4 will be bottom navigation , 3 fragments for navigation drawer and 3 toast.

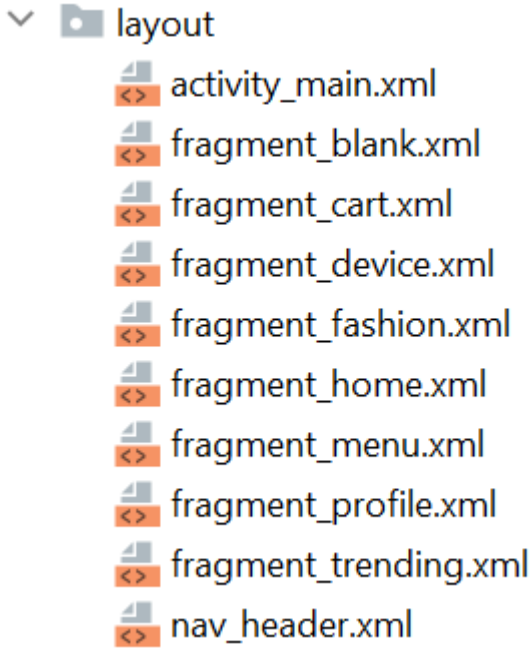




Likewise you will create rest of them



Their supporting layout files have also been created while fragments were getting created.



Now add the following code in fragment named fragment_home.xml file:

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Home"
    android:gravity="center"
    android:textColor="@color/red"
    android:textSize="40sp"/>
```

Similarly add the above code for rest of the fragments and change the text attribute to their respective function, that is if you adding the code in cart fragment the text would be changed to Cart, so on and so forth.

Add the following code for MainActivity.kt file:

```
package com.example.navigationdrawer

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.MenuItem
import android.widget.Toast
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.core.view.GravityCompat
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentTransaction
import com.example.navigationdrawer.databinding.ActivityMainBinding
import com.google.android.material.navigation.NavigationView

class MainActivity : AppCompatActivity() ,
NavigationView.OnNavigationItemSelectedListener{

    private lateinit var fragmentManager: FragmentManager
    private lateinit var binding:ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        setSupportActionBar(binding.toolbar)
        val toggle = ActionBarDrawerToggle(this,
binding.drawerLayout,binding.toolbar,R.string.nav_open,R.string.nav_close)
        binding.drawerLayout.addDrawerListener(toggle)
        toggle.syncState()

        binding.navigationDrawer.setNavigationItemSelectedListener(this)
        binding.bottomNavigation.background=null
        binding.bottomNavigation.setOnItemReselectedListener{item->
            when(item.itemId){
                R.id.bottom_home -> openFragment(HomeFragment())
                R.id.bottom_profile-> openFragment(ProfileFragment())
                R.id.bottom_menu -> openFragment(MenuFragment())
                R.id.bottom_cart-> openFragment(CartFragment())
            }
            true
        }
        fragmentManager=supportFragmentManager
        openFragment(HomeFragment())
        binding.fab.setOnClickListener{
            Toast.makeText(this, "Categories, Toast.LENGTH_SHORT").show()
        }
    }

    override fun onNavigationItemSelected(item: MenuItem): Boolean {
        when(item.itemId){
            R.id.trending -> openFragment(TrendingFragment())
            R.id.fashion -> openFragment(FashionFragment())
            R.id.device -> openFragment(DeviceFragment())
            R.id.fresh -> Toast.makeText(this,"Food
Clicked",Toast.LENGTH_SHORT)
            R.id.beauty -> Toast.makeText(this,"Beauty
```

```
Clicked",Toast.LENGTH_SHORT)
            R.id.furniture->Toast.makeText(this,"Furniture
Clicked",Toast.LENGTH_SHORT)
        }
        binding.drawerLayout.closeDrawer(GravityCompat.START)
        return true
    }

    override fun onBackPressed() {
        if (binding.drawerLayout.isDrawerOpen(GravityCompat.START)){
            binding.drawerLayout.closeDrawer(GravityCompat.START)
        }else{
            super.getOnBackPressedDispatcher().onBackPressed()
        }
    }
    private fun openFragment(fragment: Fragment){
        val fragmentTransaction: FragmentTransaction=
fragmentManager.beginTransaction()
        fragmentTransaction.replace(R.id.fragment_container,fragment)
        fragmentTransaction.commit()
    }
}
```

Practical 7 : Permissions and Security

Starting from Android 6.0 (API 23), users are not asked for permissions at the time of installation rather developers need to request the permissions at the run time. Only the permissions that are defined in the manifest file can be requested at run time.

Types of Permissions

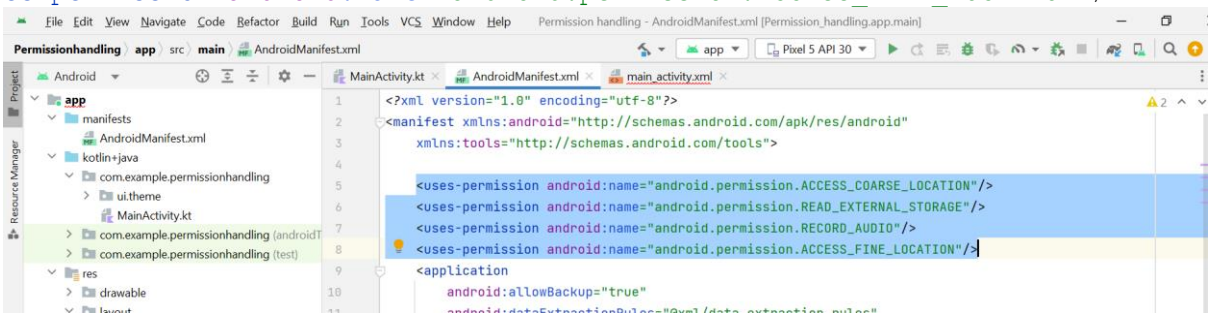
1. Install-Time Permissions: If the Android 5.1.1 (API 22) or lower, the permission is requested at the installation time at the Google Play Store. If the user Accepts the permissions, the app is installed. Else the app installation is cancelled.
2. Run-Time Permissions: If the Android 6 (API 23) or higher, the permission is requested at the run time during the running of the app.

If the user Accepts the permissions, then that feature of the app can be used. Else to use the feature, the app requests permission again.

So, now the permissions are requested at runtime. In this article, we will discuss how to request permissions in an Android Application at run time.

Step 1: Goto AndroidManifest.xml file and add following code:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

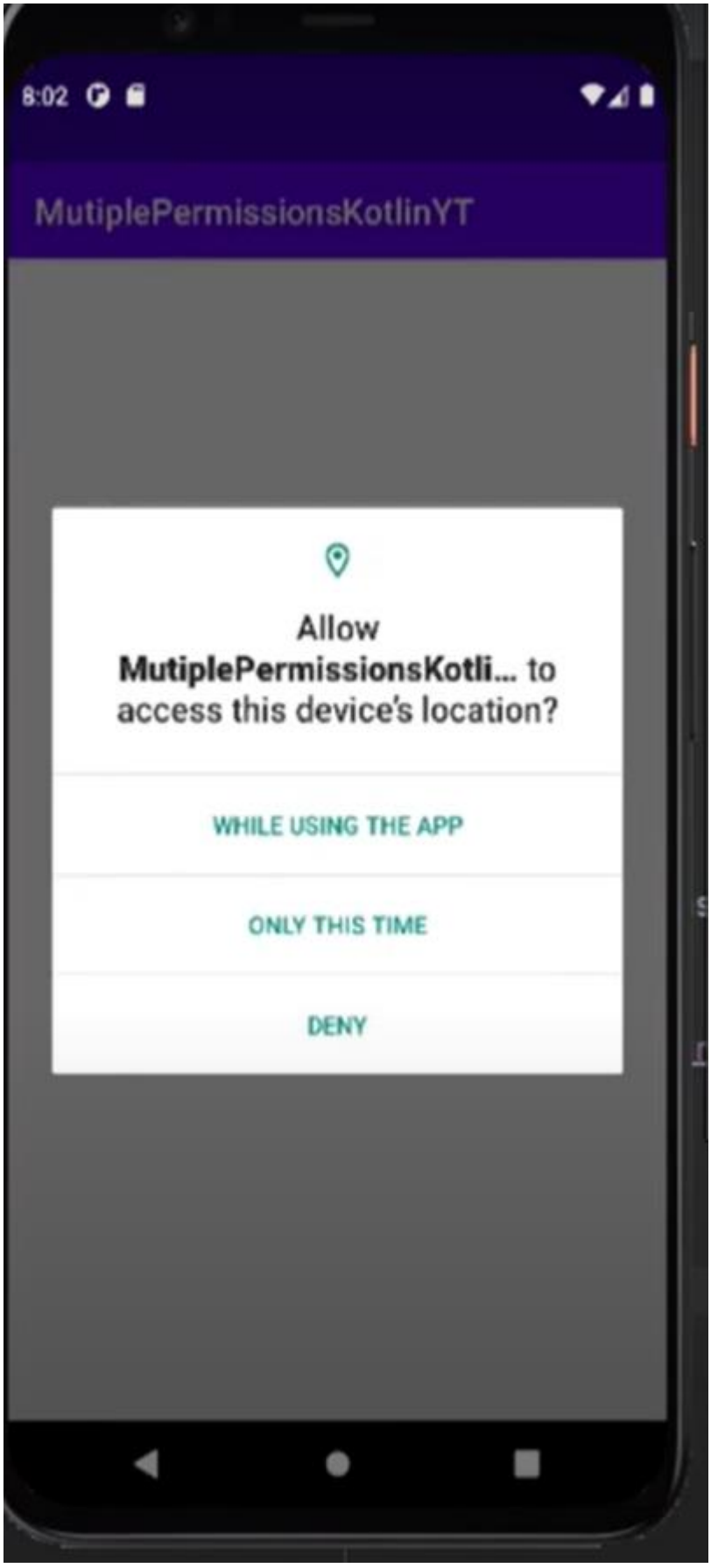


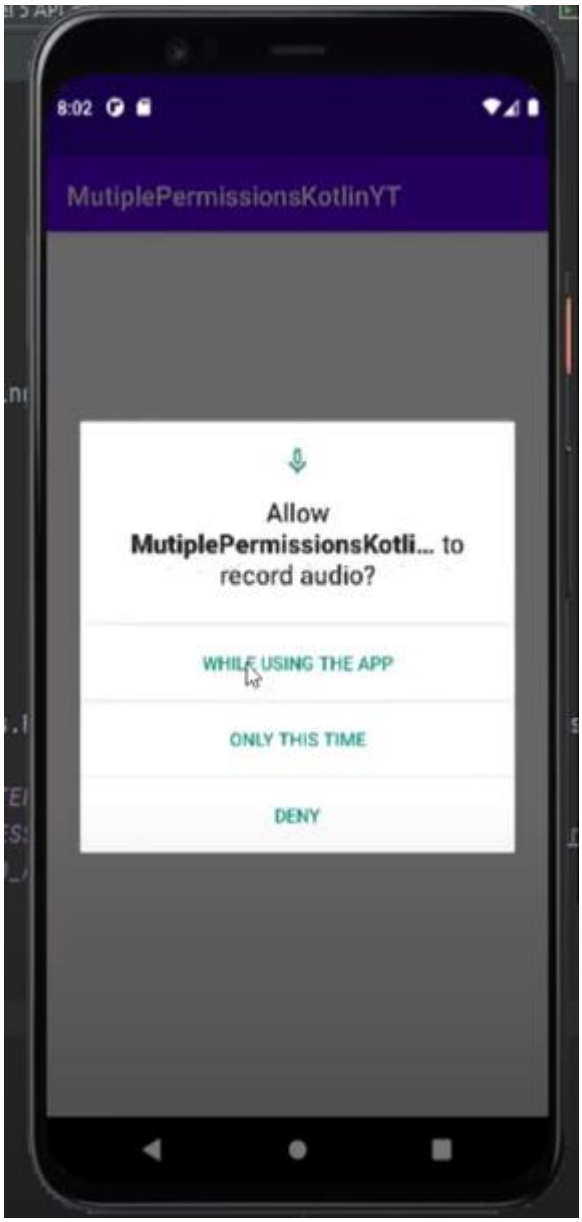
Step 2: Goto MainActivity.kt file and add the following code:

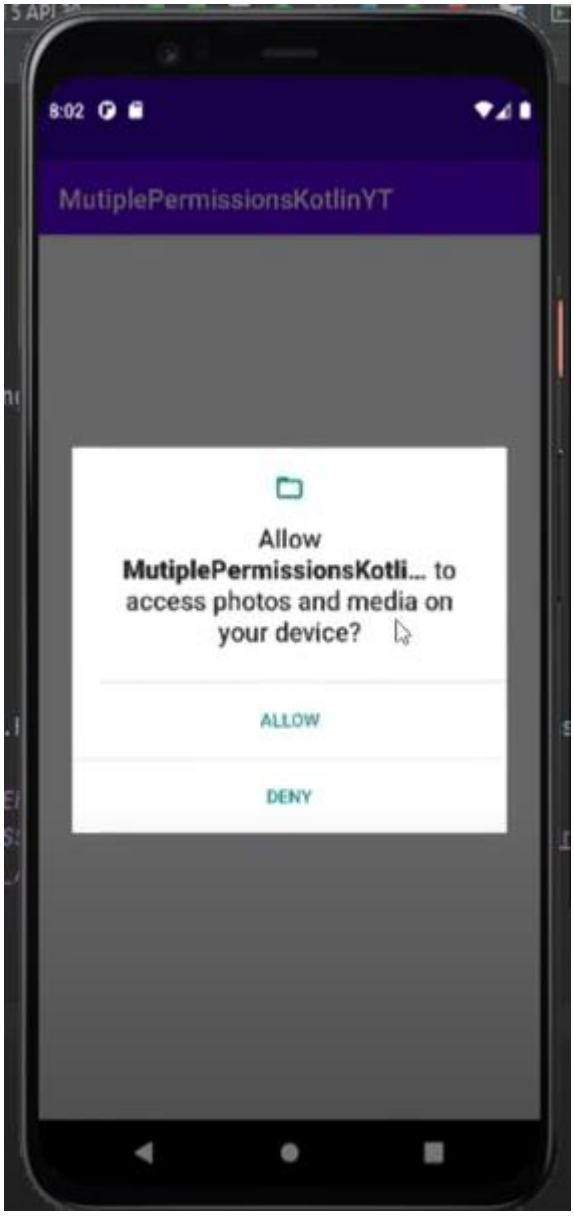
```
package com.example.permissionhandling

import android.content.pm.PackageManager
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.ManagedActivityResultLauncher
import androidx.activity.compose.setContent
import androidx.activity.result.ActivityResultLauncher
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.core.content.ContextCompat
import com.example.permissionhandling.ui.theme.PermissionHandlingTheme
```

```
class MainActivity : ComponentActivity() {

    private lateinit var permissionLauncher:
ActivityResultLauncher<Array<String>>
    private var isReadPermissionGranted = false
    private var isLocationPermissionGranted = false
    private var isRecordPermissionGranted = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

permissionLauncher=registerForActivityResult(ActivityResultContracts.Reques
tMultiplePermissions()){
            permission->
            isReadPermissionGranted =
permission[Manifest.permission.READ_EXTERNAL_STORAGE]?:isReadPermissionGran
ted
            isRecordPermissionGranted =
permission[Manifest.permission.RECORD_AUDIO]?:isRecordPermissionGranted
            isLocationPermissionGranted =
permission[Manifest.permission.ACCESS_FINE_LOCATION]?:isLocationPermissionG
ranted
        }
        requestPermission()
    }
    private fun requestPermission(){
        isReadPermissionGranted = ContextCompat.checkSelfPermission(
            this,Manifest.permission.READ_EXTERNAL_STORAGE
        )==PackageManager.PERMISSION_GRANTED
        isLocationPermissionGranted=ContextCompat.checkSelfPermission(
                this,Manifest.permission.ACCESS_FINE_LOCATION
        )== PackageManager.PERMISSION_GRANTED
        isRecordPermissionGranted = ContextCompat.checkSelfPermission(
                this, Manifest.permission.RECORD_AUDIO
                )==PackageManager.PERMISSION_GRANTED

        val permissionRequest: MutableList<String> = ArrayList()
        if(!isReadPermissionGranted){

permissionRequest.add(Manifest.permission.READ_EXTERNAL_STORAGE)
        }
        if(!isLocationPermissionGranted){
            permissionRequest.add(Manifest.permission.ACCESS_FINE_LOCATION)
        }
        if(!isRecordPermissionGranted){
            permissionRequest.add(Manifest.permission.RECORD_AUDIO)
        }
        if(permissionRequest.isNotEmpty()){
            permissionLauncher.launch(permissionRequest.toTypedArray())
        }
    }
}
```

Output:


8:02
MutiplePermissionsKotlinYT
Allow
MutiplePermissionsKotli... to
access this device's location?
WHILE USING THE APP
ONLY THIS TIME
DENY


8:02
MutiplePermissionsKotlinYT
Allow
MutiplePermissionsKotli... to
record audio?
WHILE USING THE APP
ONLY THIS TIME
DENY


8:02
MutiplePermissionsKotlinYT
Allow
MutiplePermissionsKotli... to
access photos and media on
your device?
ALLOW
DENY

Practical 8 : Including Telephone API

Intent is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers, and services, here use to make phone calls. This application basically contains one activity with edit text to write the phone number on which you want to make a call and a button to call that number.

Step 1: Create a New Project in Android Studio

Step 2: Add Permission to AndroidManifest.xml File

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

Step 3 : Add the following code in activity_main.xml file:

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

        <!-- Edit text for phone number -->
        <EditText
                android:id="@+id/editText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentTop="true"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="30dp" />

        <!-- Button to make call -->
        <Button
                android:id="@+id/button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentTop="true"
```

```
        android:layout_centerHorizontal="true"
        android:layout_marginTop="115dp"
        android:padding="5dp"
        android:text="Make Call!!" />
</RelativeLayout>
```

Step 4: Make the following changes in MainActivity.kt file:

```
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    // define objects for edit text and button
    private lateinit var edittext: EditText
    private lateinit var button: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Getting instance of edittext and button
        button = findViewById(R.id.button)
        edittext = findViewById(R.id.editText)

        // Attach set on click listener to the button for initiating intent
        button.setOnClickListener(View.OnClickListener {
            // getting phone number from edit text and changing it to String
```

```
                val phone_number = edittext.text.toString()

                // Getting instance of Intent with action as ACTION_CALL
                val phone_intent = Intent(Intent.ACTION_CALL)

                // Set data of Intent through Uri by parsing phone number
                phone_intent.data = Uri.parse("tel:$phone_number")

                // start Intent
                startActivity(phone_intent)
            })
        }
}
```

Output: