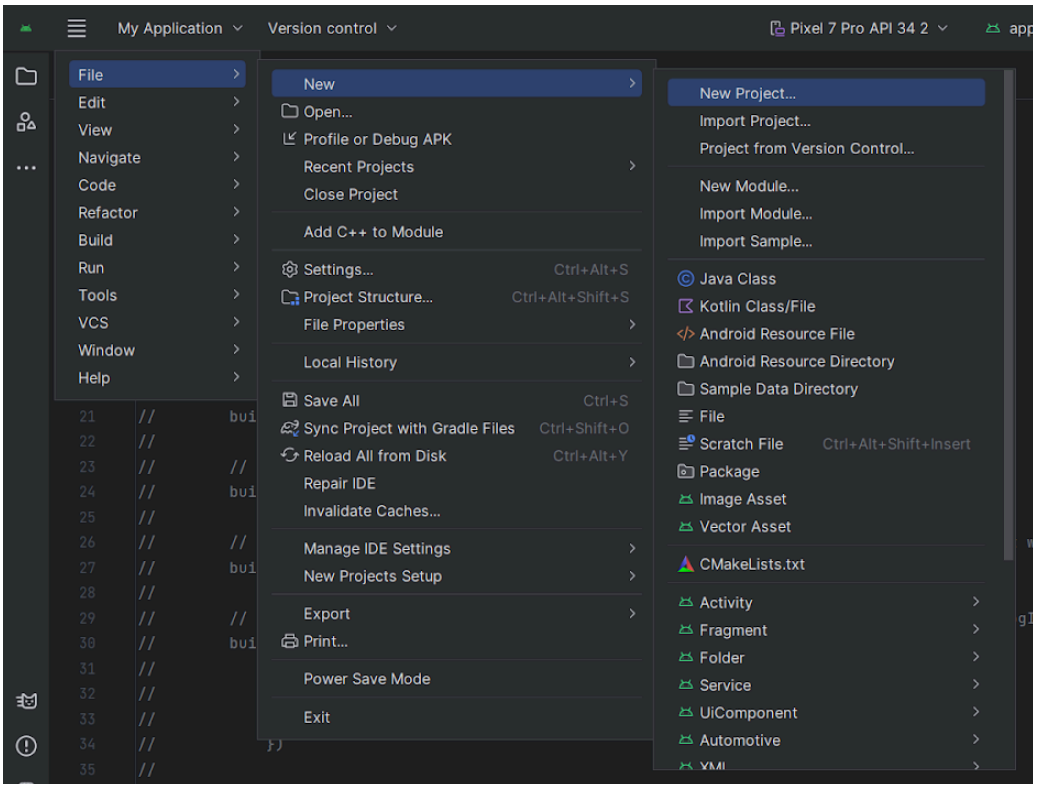
PRACTICAL 1

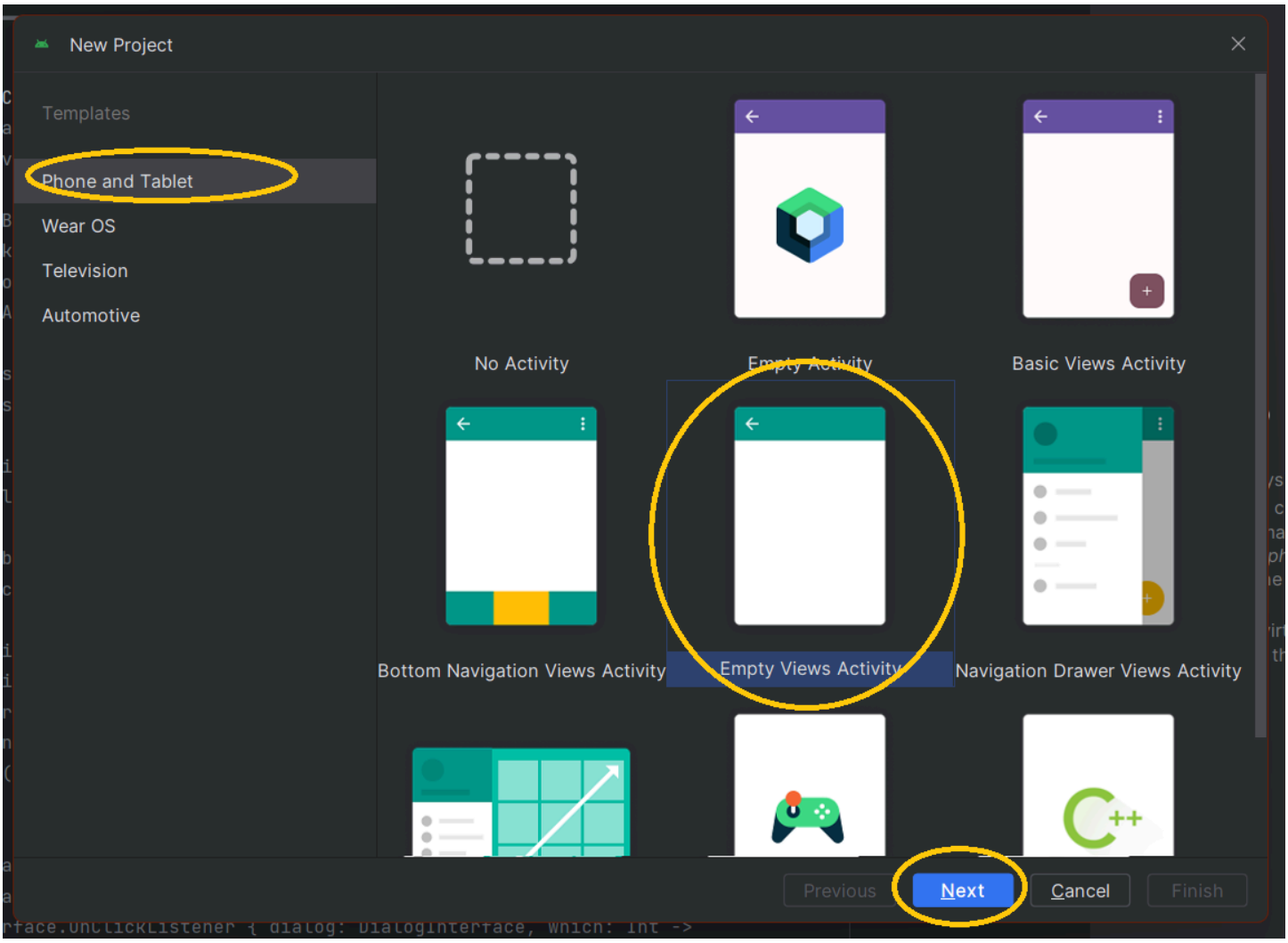
Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals: Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple “Hello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

Creating a project using Android Studio Hedgehog.

Go to File -> New -> New Project

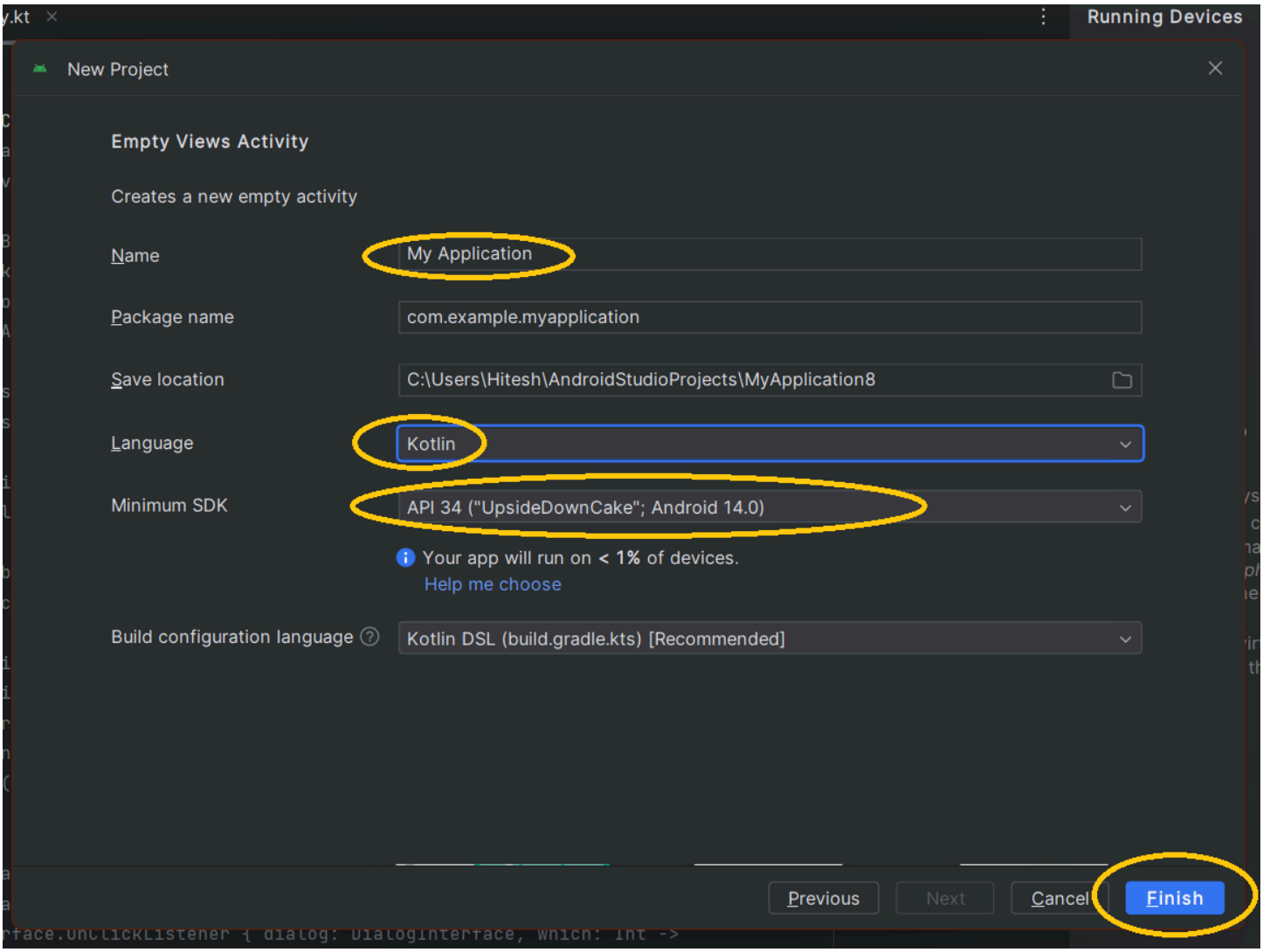


Select Empty View Activity



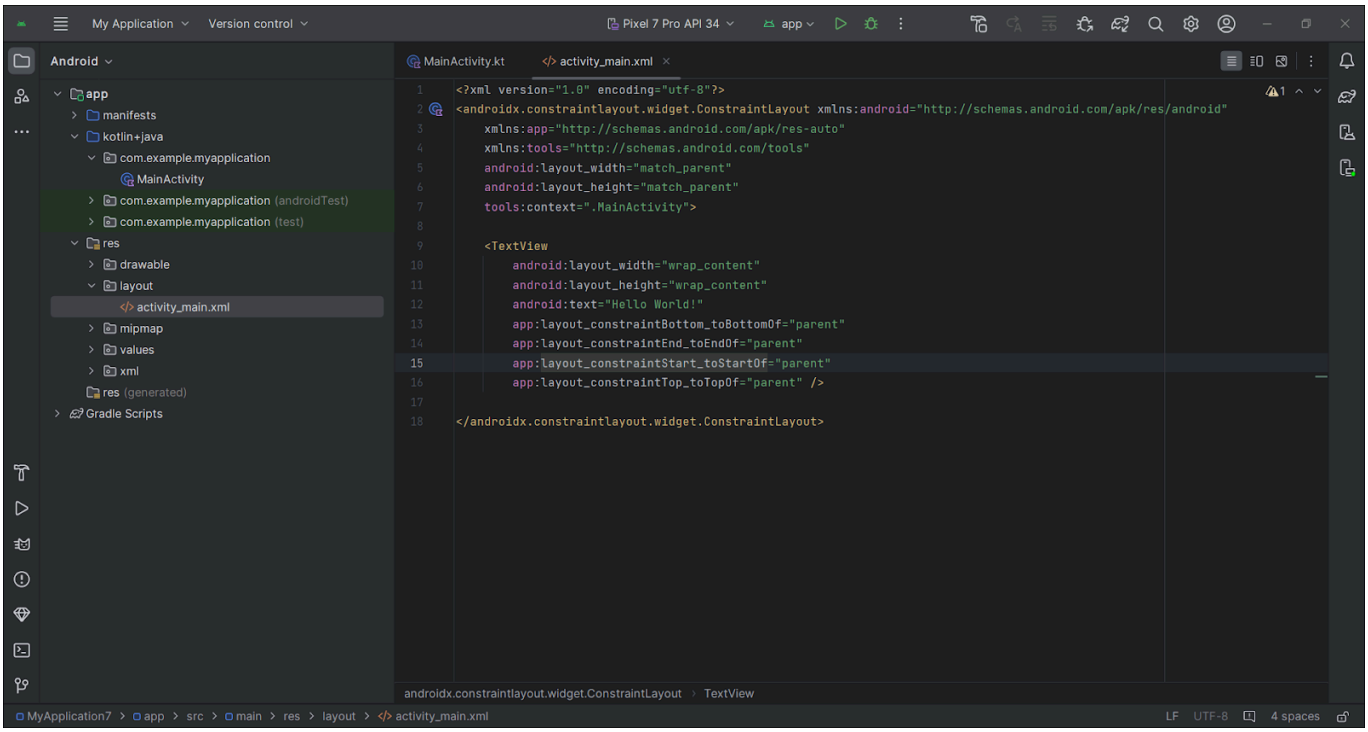
Change the name of the project

select language from kotlin the language drop down



On finishing the project will load in some time.

The below is the window you will be left with.



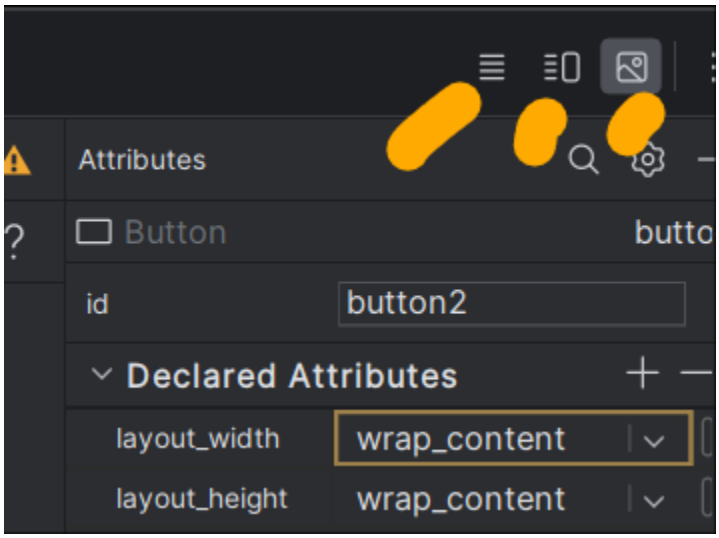
open main activity to view the code

MainActivity.kt (no changes)

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

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

}
```

open activity_main.xml which will be present inside the res->layouts->activity_main.xml

(there are 2 buttons for viewing code, split , design)



(Code Split Design View Available in Studio)

activity_main.xml (no change)

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

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

</android.support.constraint.ConstraintLayout>
```

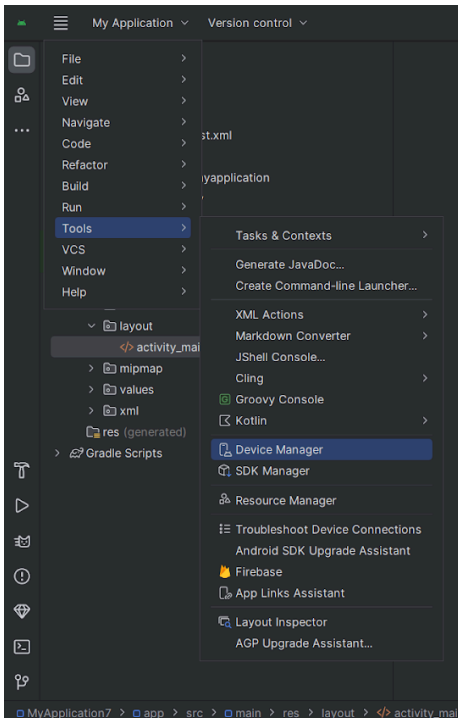
Running the app.

For running the app you will be required a virtual device for that you need to create a device. Let’s create one.

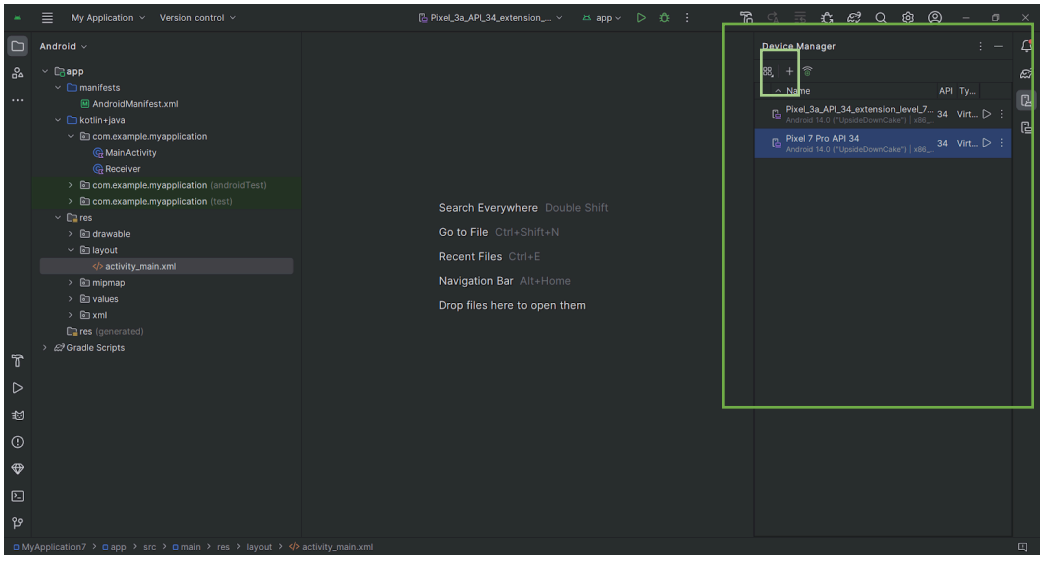
Create and manage virtual devices:

To open the AVD Manager, do one of the following:

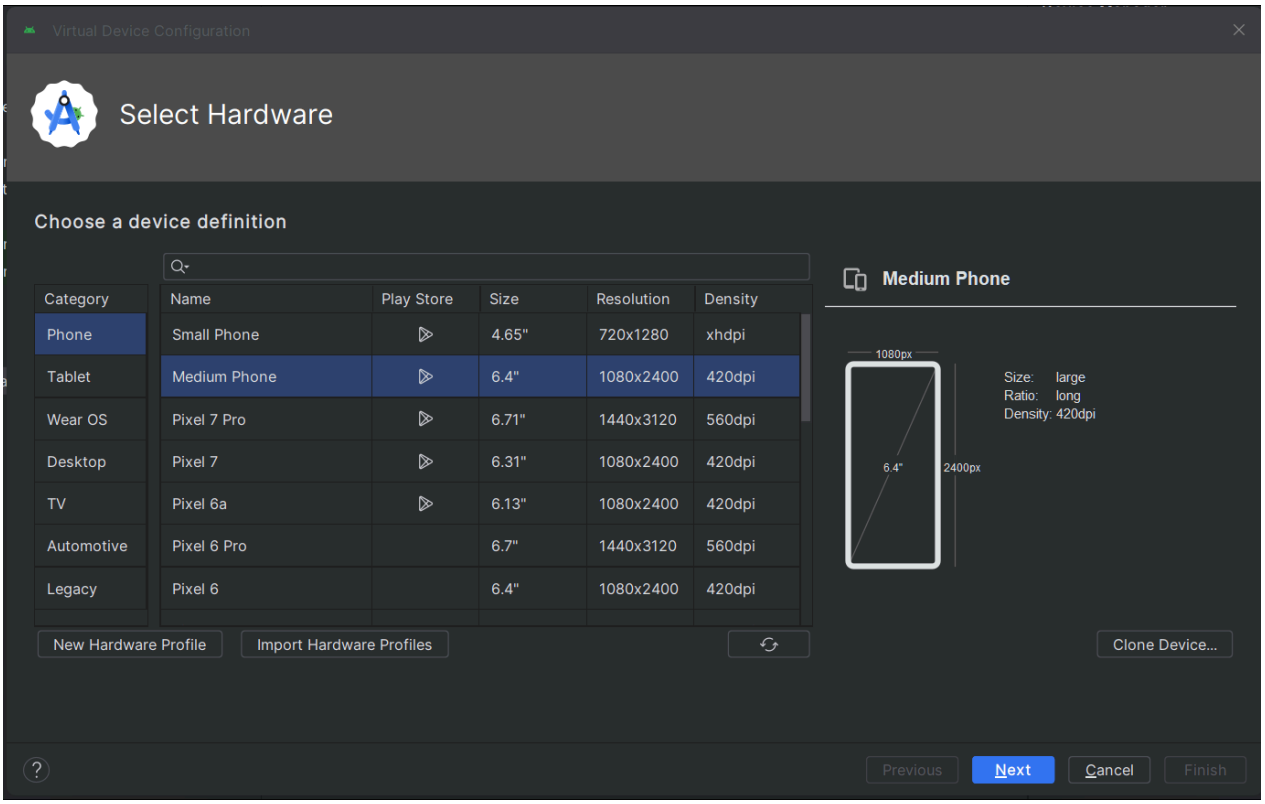
1. Select Tools > Device Manager.



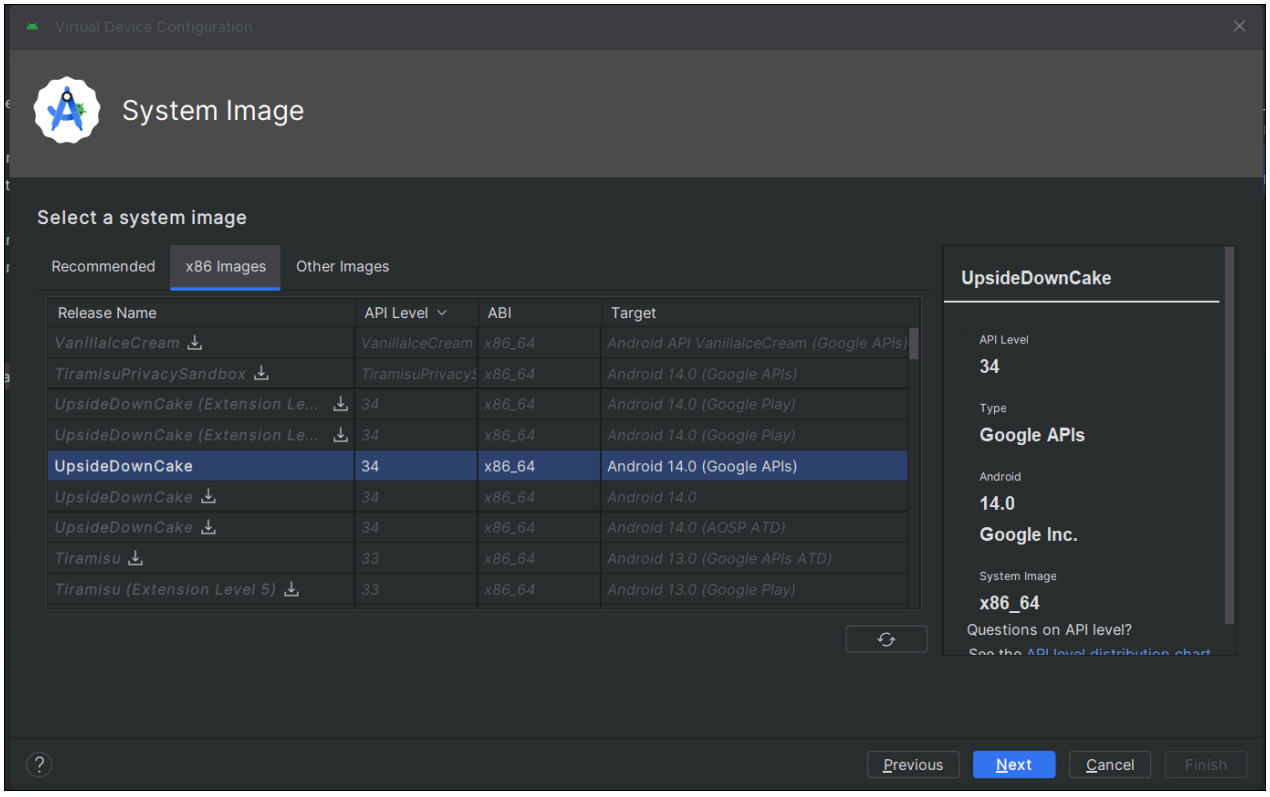
It will be shown in the right-hand side



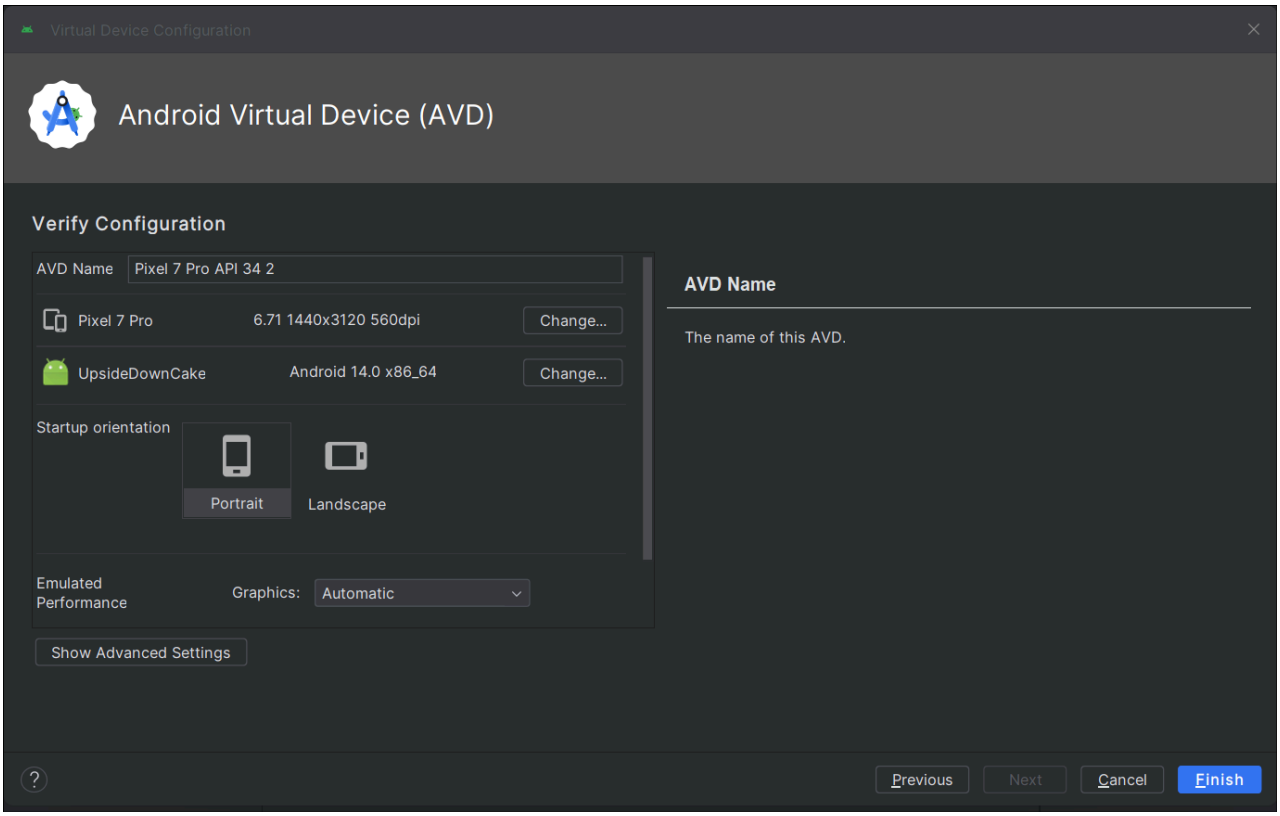
2. Click on the + icon on the Device Manager tab.



Select Api 34 Android 14 only in Target tab

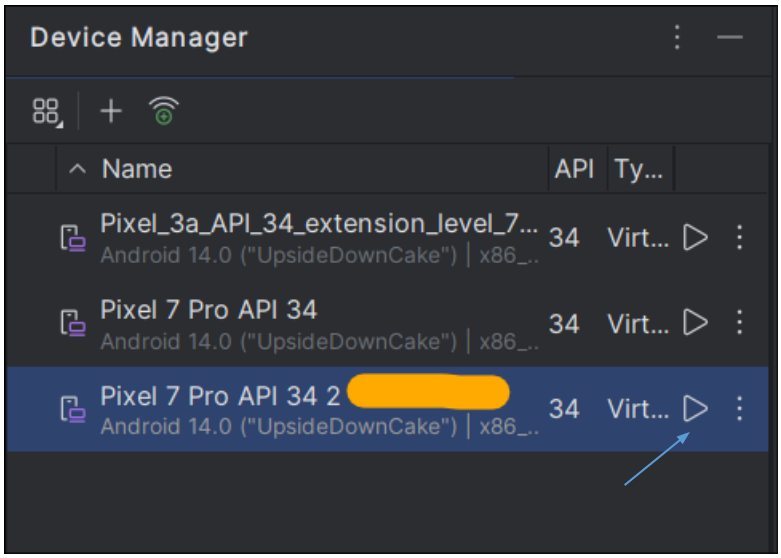


Enter a new name in the AVD Name



Click Finish

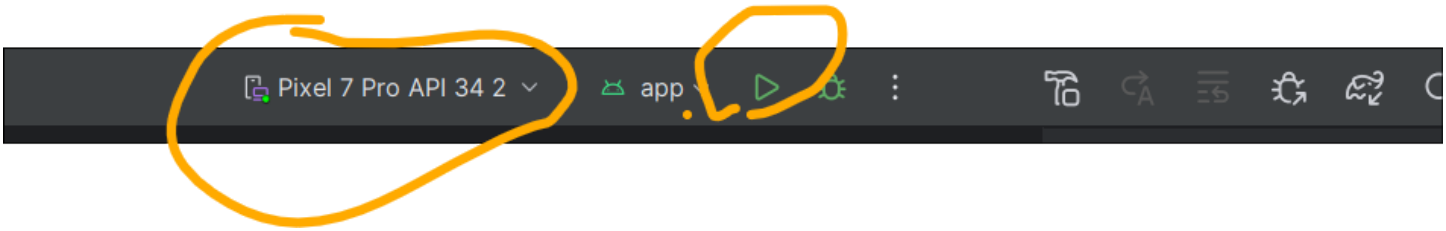
A new device added in the Device Manager list.



Run the device with the play button.

For running the Project.

On top of the android studio window you will find a Green play button. Before that you will see a device name written in there. click the dropdown to select the virtual device you recently created then click on the green run button to run your android project.



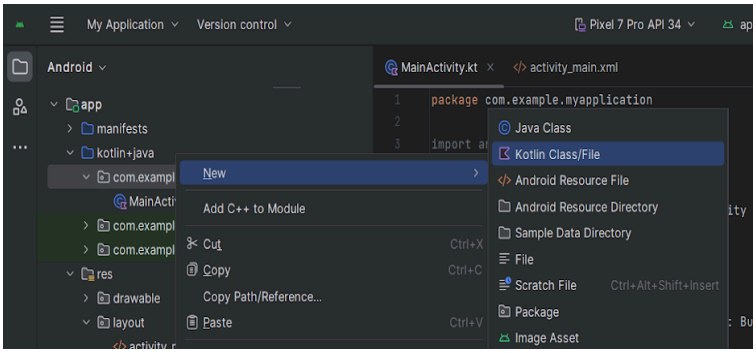
Broadcast Activity

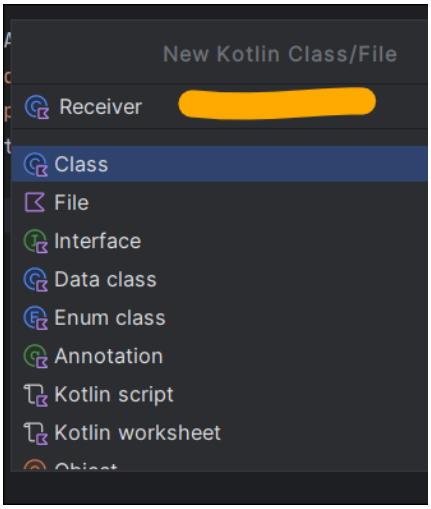
Apps can receive a Broadcast Receiver activity in two ways:

1) Through manifest-declared receivers
2) Context-registered receivers

In this example, we are approaching manifest-declared Receivers.

1. Create an android app, for creating an Android app.
2. Creating a class named Receiver.





3. Extend the class with “BroadcastReceiver”. The code for this is given in the code below.
4. Then implement onReceive(Context, Intent) where onReceive method receives each message as an Intent object parameter.

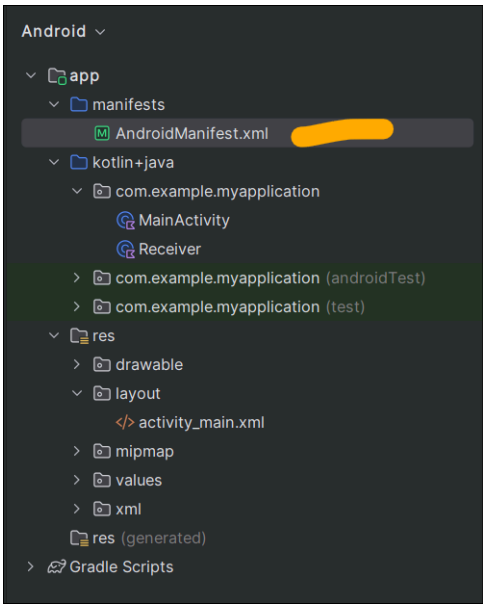
Receiver.kt

```
package com.example.myapplication
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.widget.Toast
class Receiver :BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val isAirplaneModeEnabled = intent?.getBooleanExtra("state",
false) ?: return

if (isAirplaneModeEnabled) {

   Toast.makeText(context, "Airplane Mode Enabled",
Toast.LENGTH_LONG).show()

} else {

   Toast.makeText(context, "Airplane Mode Disabled",
Toast.LENGTH_LONG).show()

} }
}
```

5. Declare a broadcast receiver in the manifest file

Add the element<receiver> in your app's manifest. Here is a code snap.

you will find the file inside the manifest folder.



The code to add is written in bold

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".Receiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.AIRPLANE_MODE" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

6. MainActivity code add the receiver Listener

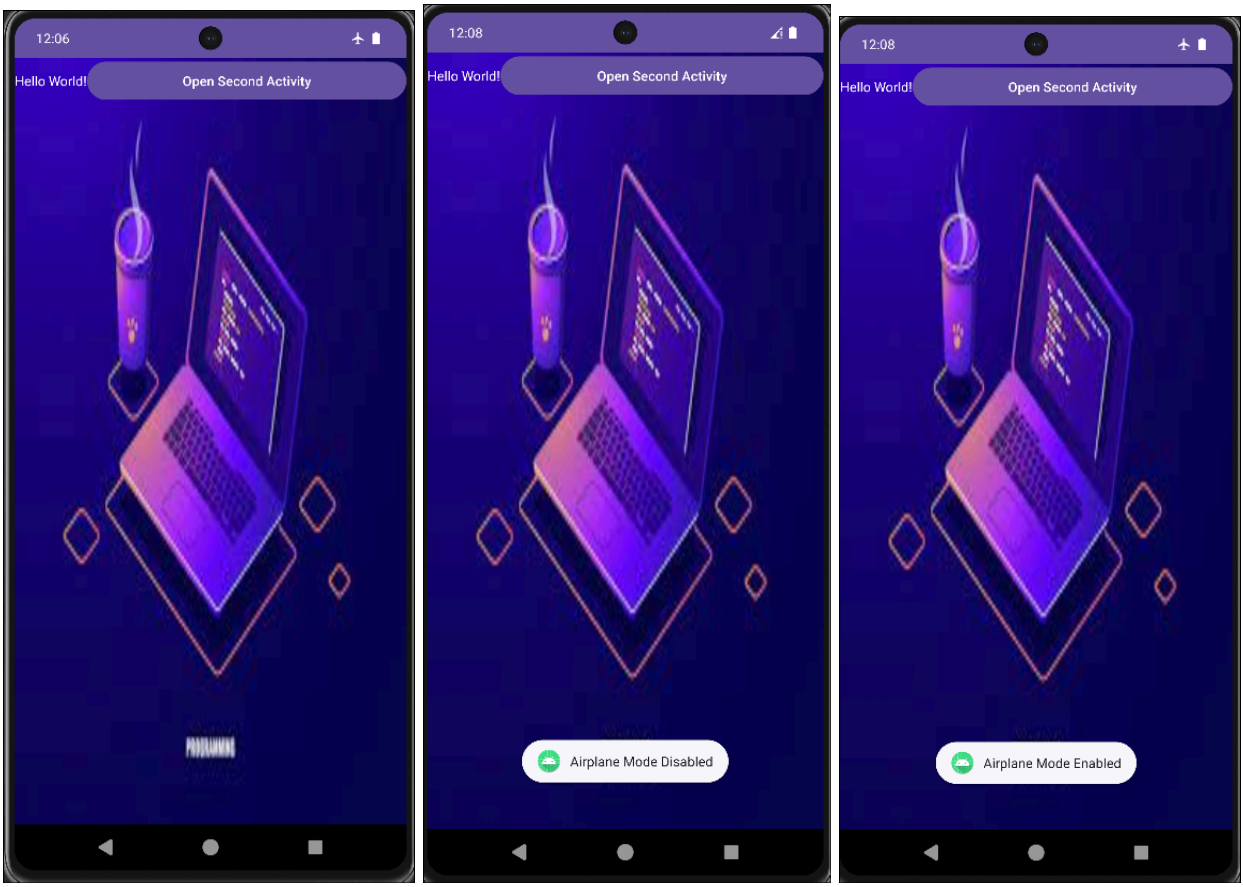
MainActivity.kt: (Code to add is written in bold)

```
package com.example.myapplication

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast

class MainActivity : AppCompatActivity() {

   val  tag = "StateChange"
lateinit var receiver: Receiver
```

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

    receiver = Receiver()
    IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED).also {

        registerReceiver(receiver, it)
    }
}

override fun onStop() {
    super.onStop()
    Log.i(tag, "onStop")
    unregisterReceiver(receiver)
}

}
```

upon running the app from the run button.

You will get the output as below.

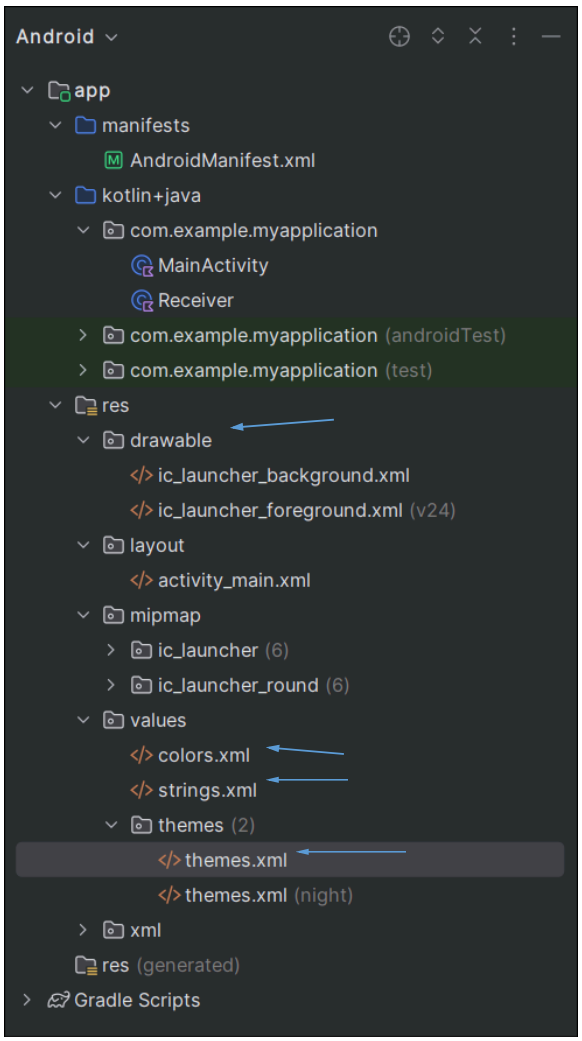


PRACTICAL 2

Programming Resources

Android Resources: (Color, Theme, String, Drawable, Dimension, Image).

Android Folders to view in the project.



open colors.xml and see the code below present

colors.xml (no change) (if you want to add your code you can add).

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
```

```
    <color name="white">#FFFFFFFF</color>
</resources>
```

open themes.xml to view only

values/themes.xml (no change)

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.MyApplication"
parent="Theme.Material3.DayNight.NoActionBar">
    </style>

    <style name="Theme.MyApplication" parent="Base.Theme.MyApplication" />
</resources>
```

open strings.xml to view only

strings.xml (no change)

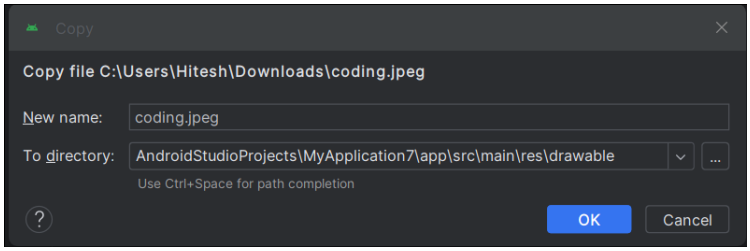
here the app_name is the variable name that is used at multiple places inside the project

```
<resources>
    <string name="app_name">My Application</string>
</resources>
```

Drawable

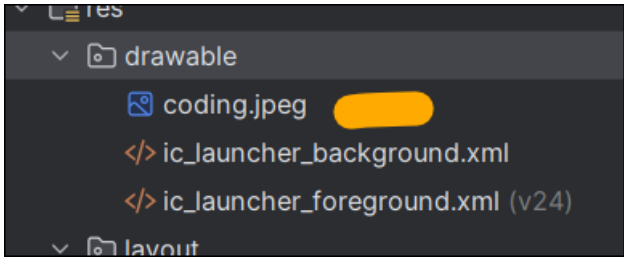
1. Copy the image from the file system if you want to add an image in the project. (Image name should be lowercase without any special characters and the format supported to copy image is majorly jpg image format.
2. Then Paste that image file inside the drawable folder. by right clicking then clicking paste.

3. You will get a pop up to change the image name here.



update name then click ok

as you can see below the image will be added into the drawable folder



MainActivity.kt (no changes)

```
package com.example.myapplication

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

activity_main.xml: (write the code below)

here we have added the same image to view as a background in the layout code

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

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

</LinearLayout>
```

Output: run the app you will see the output as this.

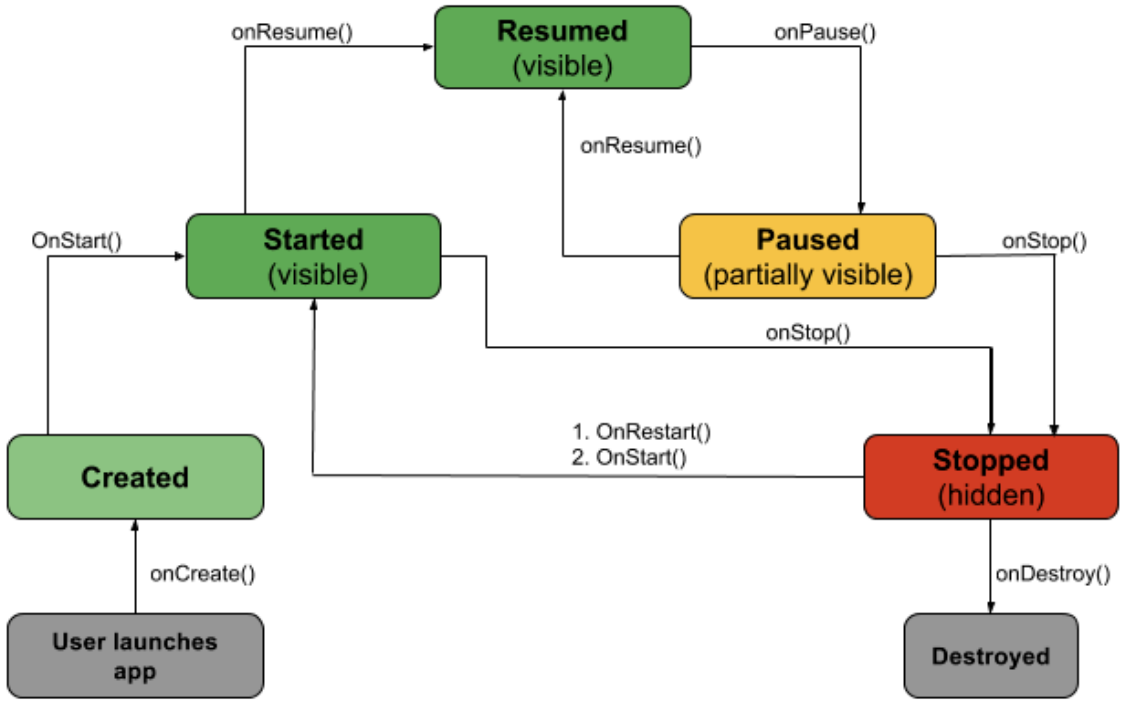

Hello World!

PRACTICAL 3

Programming Activities and fragments

Activity Life Cycle, Activity methods, Multiple Activities, Life Cycle of fragments and multiple fragments.

Activity Lifecycle:



onCreate(): Called by the OS when the activity is first created. This is where you initialize any UI elements or data objects. You also have the savedInstanceState of the activity that contains its previously saved state, and you can use it to recreate that state.

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

onStart(): Just before presenting the user with an activity, this method is called. It's always followed by onResume(). In here, you generally should start UI animations, audio based content or anything else that requires the activity's contents to be on screen.

onResume(): As an activity enters the foreground, this method is called. Here you have a good place to restart animations, update UI elements, restart camera previews, resume audio/video playback or initialize any components that you release during onPause().

onPause(): This method is called before sliding into the background. Here you should stop any visuals or audio associated with the activity such as UI animations, music playback or the camera. This method is followed by onResume() if the activity returns to the foreground or by onStop() if it becomes hidden.

onStop(): This method is called right after onPause(), when the activity is no longer visible to the user, and it's a good place to save data that you want to commit to the disk. It's followed by either onRestart(), if this activity is coming back to the foreground, or onDestroy() if it's being released from memory.

onRestart(): Called after stopping an activity, but just before starting it again. It's always followed by onStart().

onDestroy(): This is the final callback you'll receive from the OS before the activity is destroyed. You can trigger an activity's destruction by calling finish(), or it can be triggered by the system when the system needs to recoup memory. If your activity includes any background threads or other long-running resources, destruction could lead to a memory leak if they're not released, so you need to remember to stop these processes here as well.

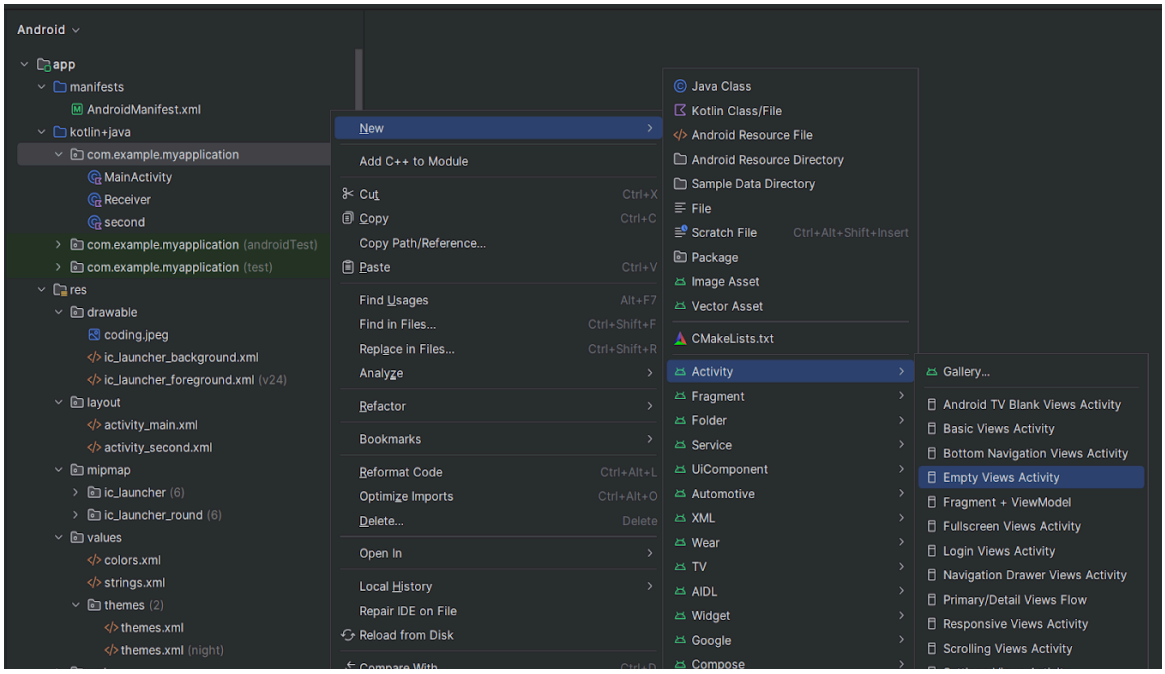
here is the sample code of activity_main.xml

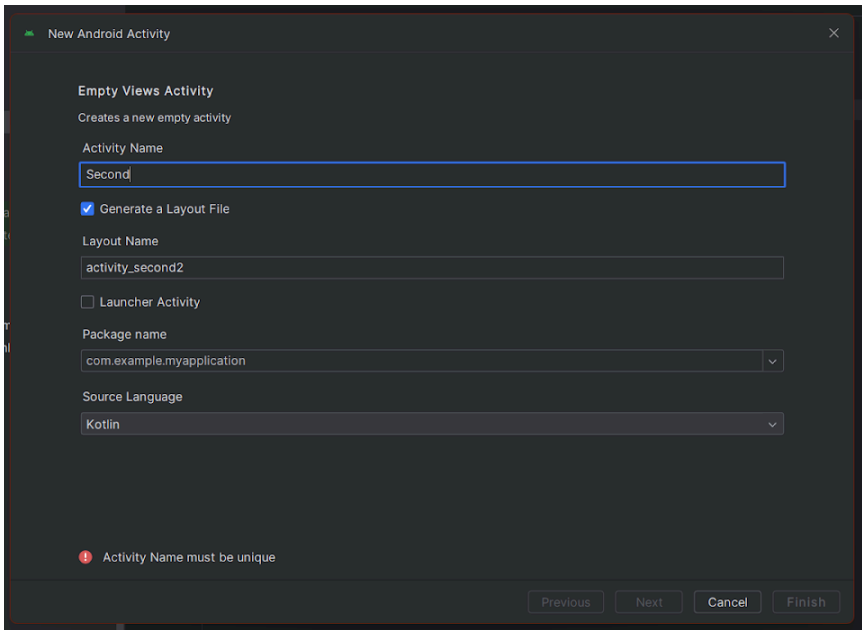
```
package com.example.myapplication

import android.os.Bundle
import android.os.PersistableBundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
   val  tag = "StateChange";
   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)

   }

   override fun onStart() {
       super.onStart()
       Log.i(tag, "onStart");
   }

   override fun onResume() {
       super.onResume()
```

```
        Log.i(tag, "onResume");
    }

    override fun onPause() {
        super.onPause()
        Log.i(tag, "onPause");
    }

    override fun onStop() {
        super.onStop()
        Log.i(tag, "onStop");
unregisterReceiver(receiver)

    }

    override fun onRestart() {
        super.onRestart()
        Log.i(tag, "onRestart");
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.i(tag, "onDestroy");
    }

}
```

Multiple Activities:

Creating Second Activity





Second.kt

```
package com.example.myapplication


import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle


class Second : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
    }
}
```

activity_second.xml

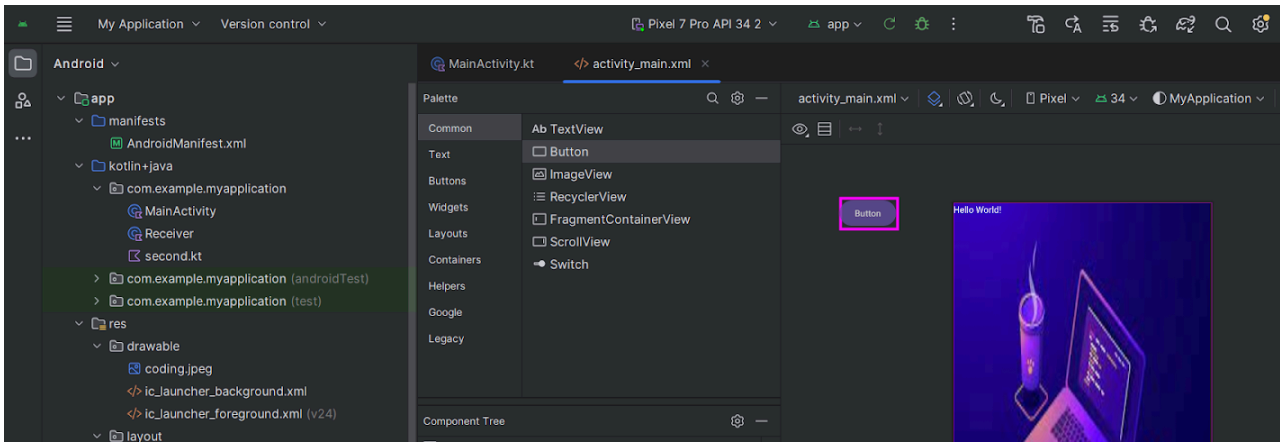
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

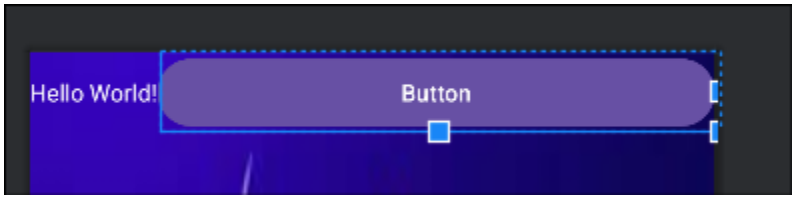
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Second">

    <TextView
        android:id="@+id/textView"
        android:layout_width="116dp"
        android:layout_height="90dp"
        android:text="Second Activity"
        tools:layout_editor_absoluteX="0dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Open Main Activity then add a button to have a click event on it to open the second activity

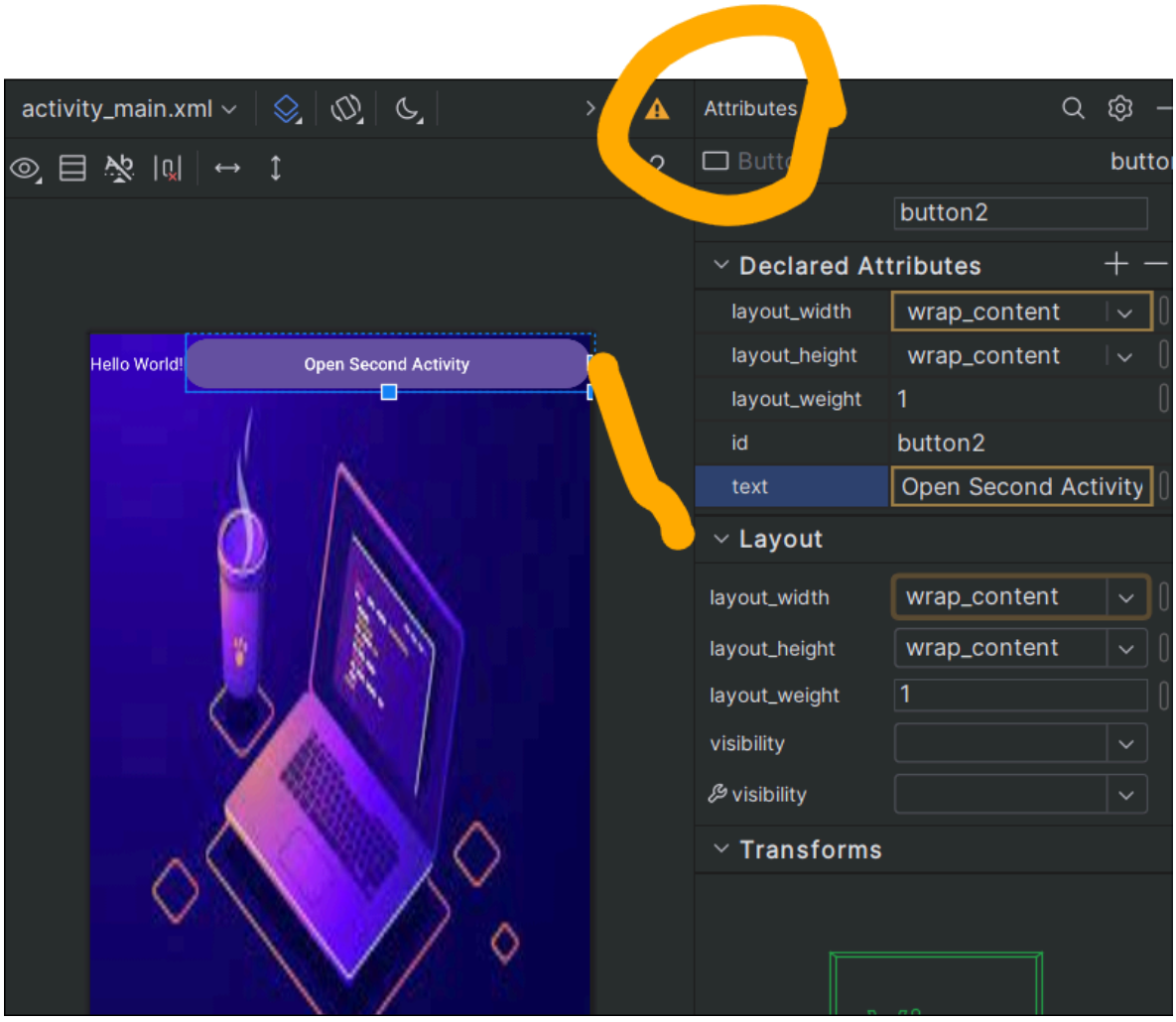
activity_main.xml

Open Design Section now





Change button text



(open Code Section to view the button id)

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




<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
```

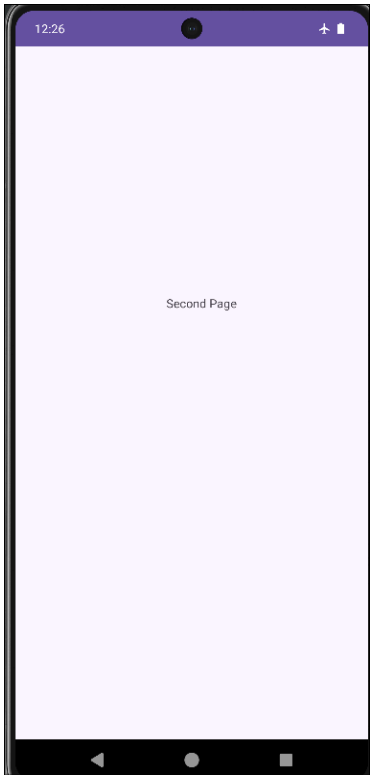
```
    android:background="@drawable/coding">

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

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Open Second Activity" />

</LinearLayout>
```

Code of MainActivity.kt (here type the code which is in bold)

change the button id based on the xml code

```
package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity

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

        val buttonClick = findViewById<Button>(R.id.button2)
```

```
        buttonClick.setOnClickListener {
            val intent = Intent(this, Second::class.java)
            startActivity(intent)
        }
    }
}
```



PRACTICAL 4

Programs related to different Layouts

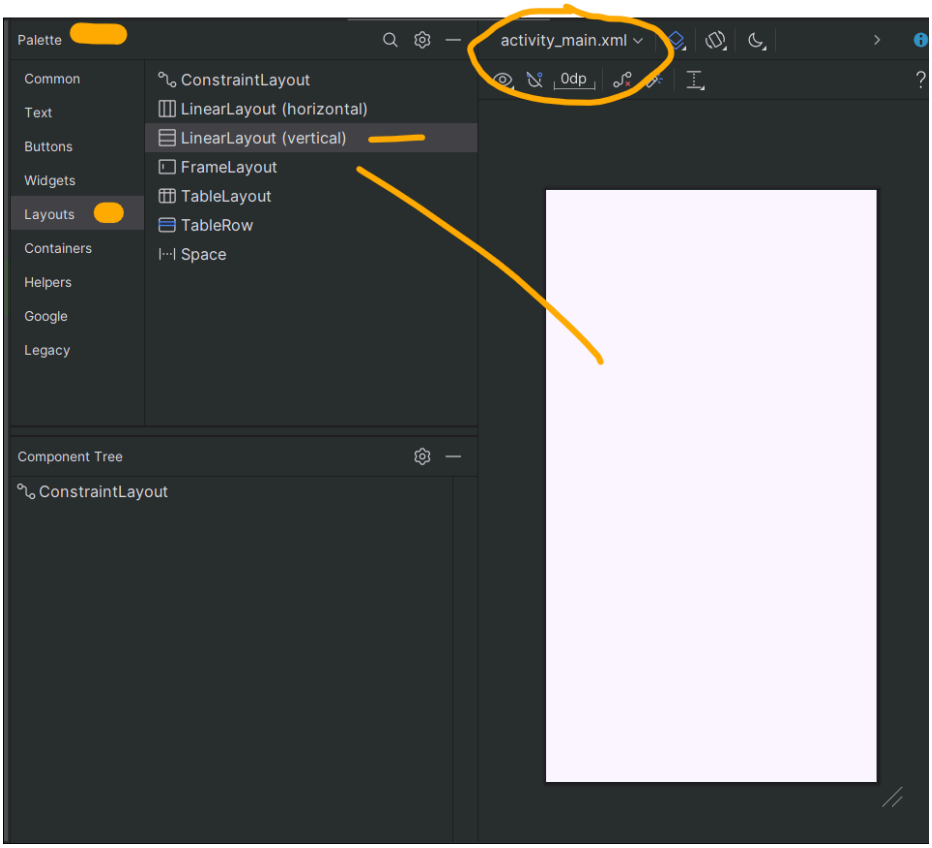
Linear, Relative, Table, Absolute, Frame, List View, Grid View.

1. Linear layout:

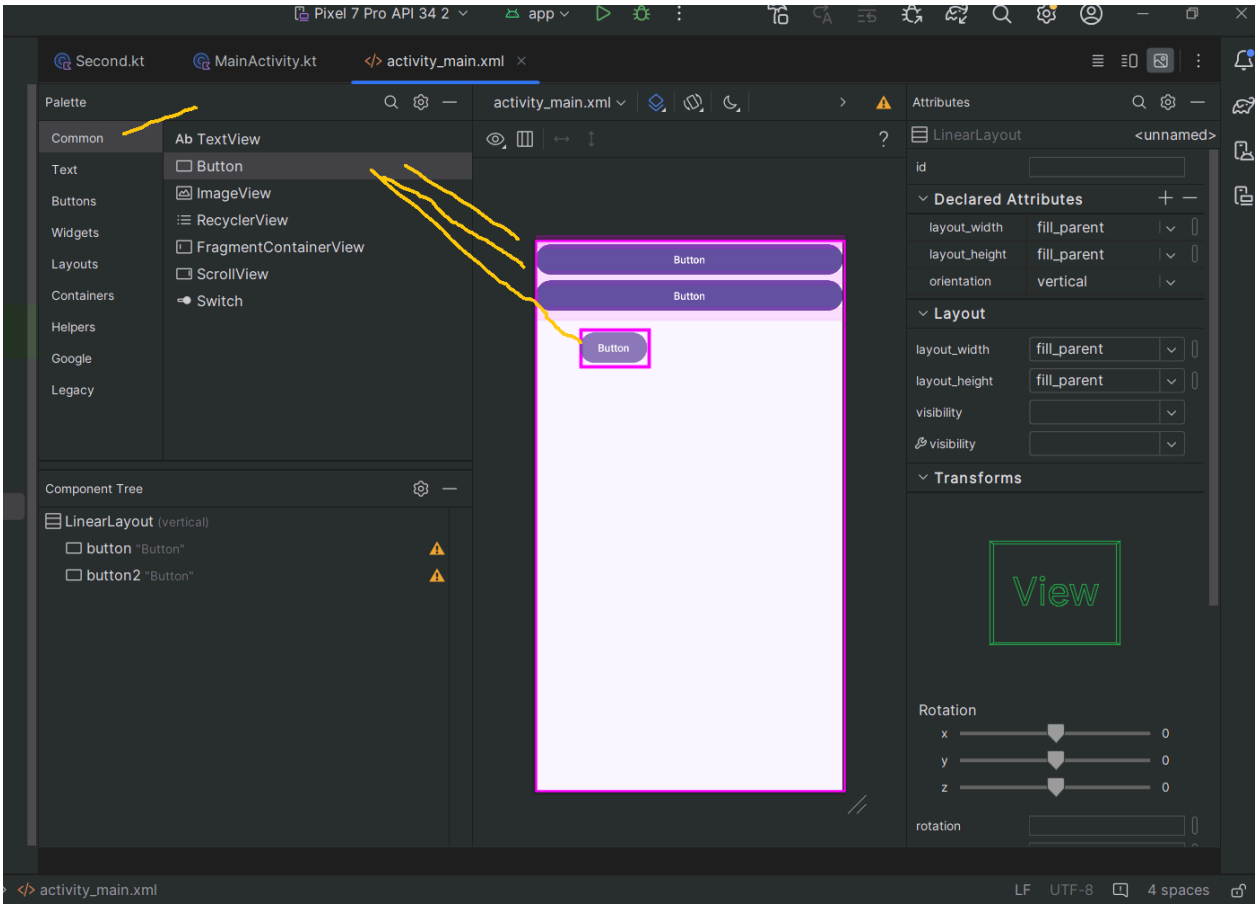
For Linear Layout go to the XML. You will find a ConstraintLayout already present there. Remove it and then add a Linear Layout code as mentioned below.

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


</LinearLayout>
```

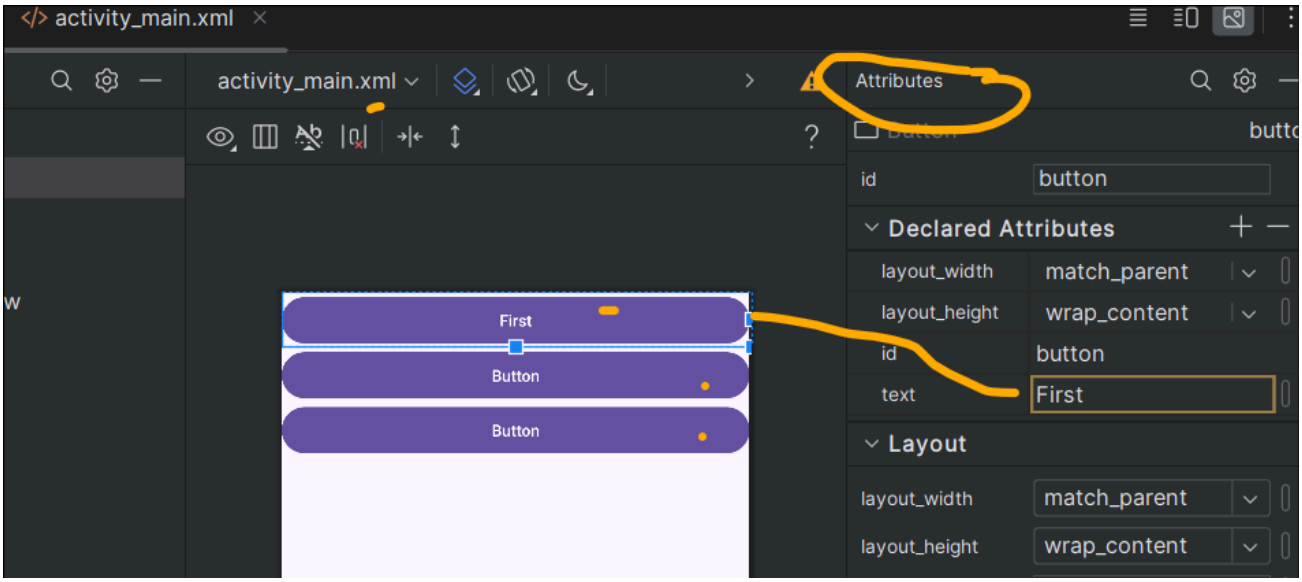
Drag and drop of layout won't happen as there is no layout present to put items from the Palette.

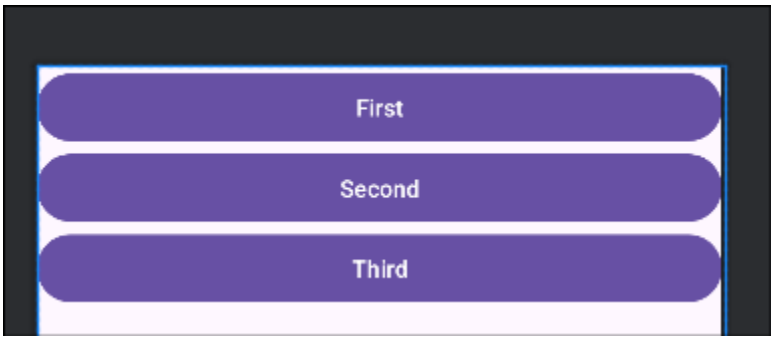


Then drag and drop the 3 buttons in the design by going in the common section then select the button to drag.



Rename the buttons by clicking the specific button and changing its text attributes at the right hand side in the Attributes tab.



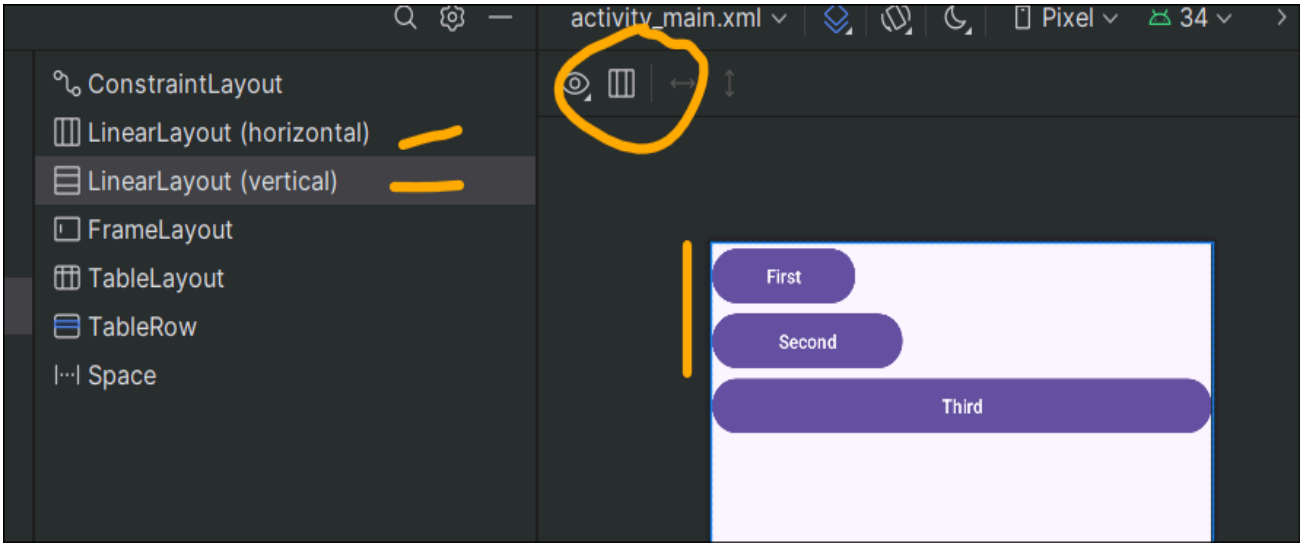


Linear Layout has 2 variant

1. LinearLayout (horizontal)
2. LinearLayout (vertical)

   You can change the layout to horizontal to vertical and vice versa by clicking on the button given in the image below.

   Change the button size so that u can view the UI working properly



Final XML Code with horizontal Layout will look like this

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">

    <Button
```

```
        android:id="@+id/button"
        android:layout_width="118dp"
        android:layout_height="wrap_content"
        android:text="First" />

    <Button
        android:id="@+id/button2"
        android:layout_width="157dp"
        android:layout_height="wrap_content"
        android:text="Second" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Third" />

</LinearLayout>
```

2. Relative:

In the relative layout the items are placed relative to each other.

The below code gives a better example in the code as you can see there is an input field called EditText with ID name1. Then below it is a Linear layout which has a special attribute **android:layout_below=”@+id/name1”** which tells the relative layout section to show the linear layout below the Edit Text field which is relatively set.

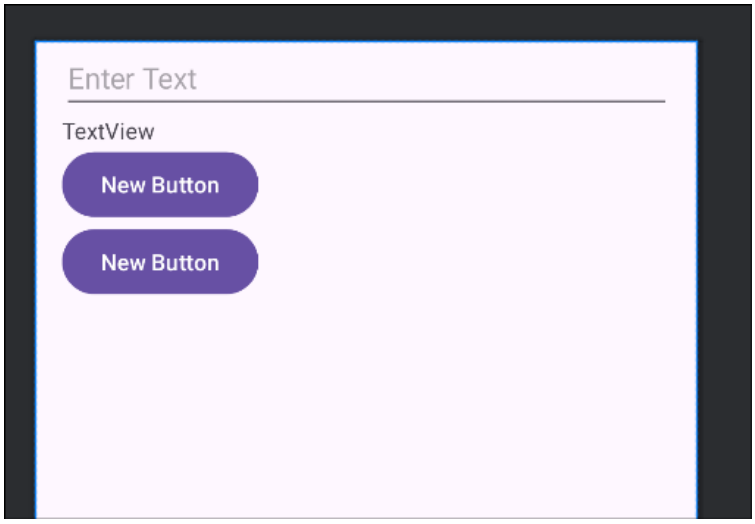
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:paddingLeft="16dp"
android:paddingRight="16dp" >
<EditText
    android:id="@+id/name1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Text" />
```

```
<LinearLayout
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_alignParentStart="true"
   android:layout_below="@+id/name1">

   <TextView
       android:id="@+id/textView3"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="TextView" />

   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="New Button"
       android:id="@+id/button" />

   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="New Button"
       android:id="@+id/button2" />

</LinearLayout>
```

```
</RelativeLayout>
```

Output

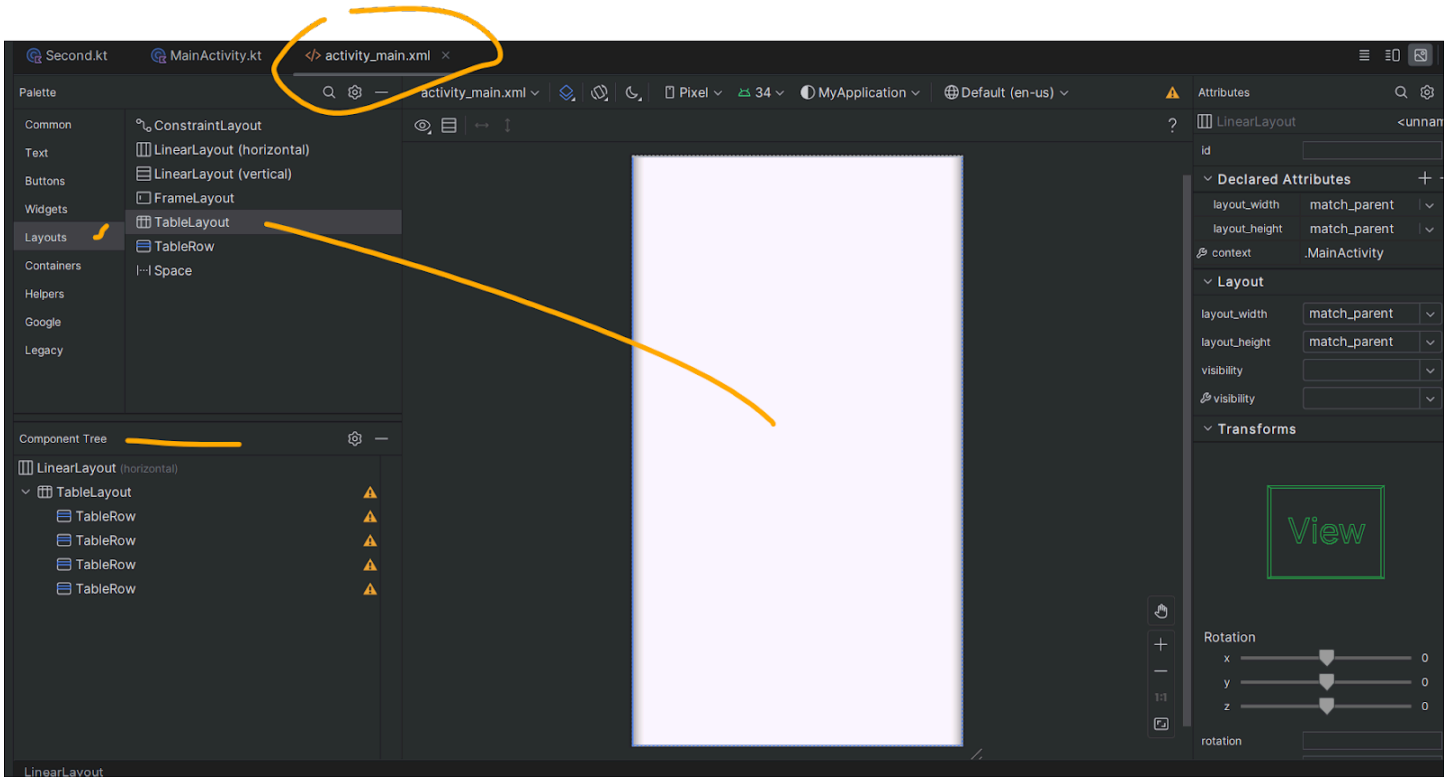
As the code the settings



3. Table:

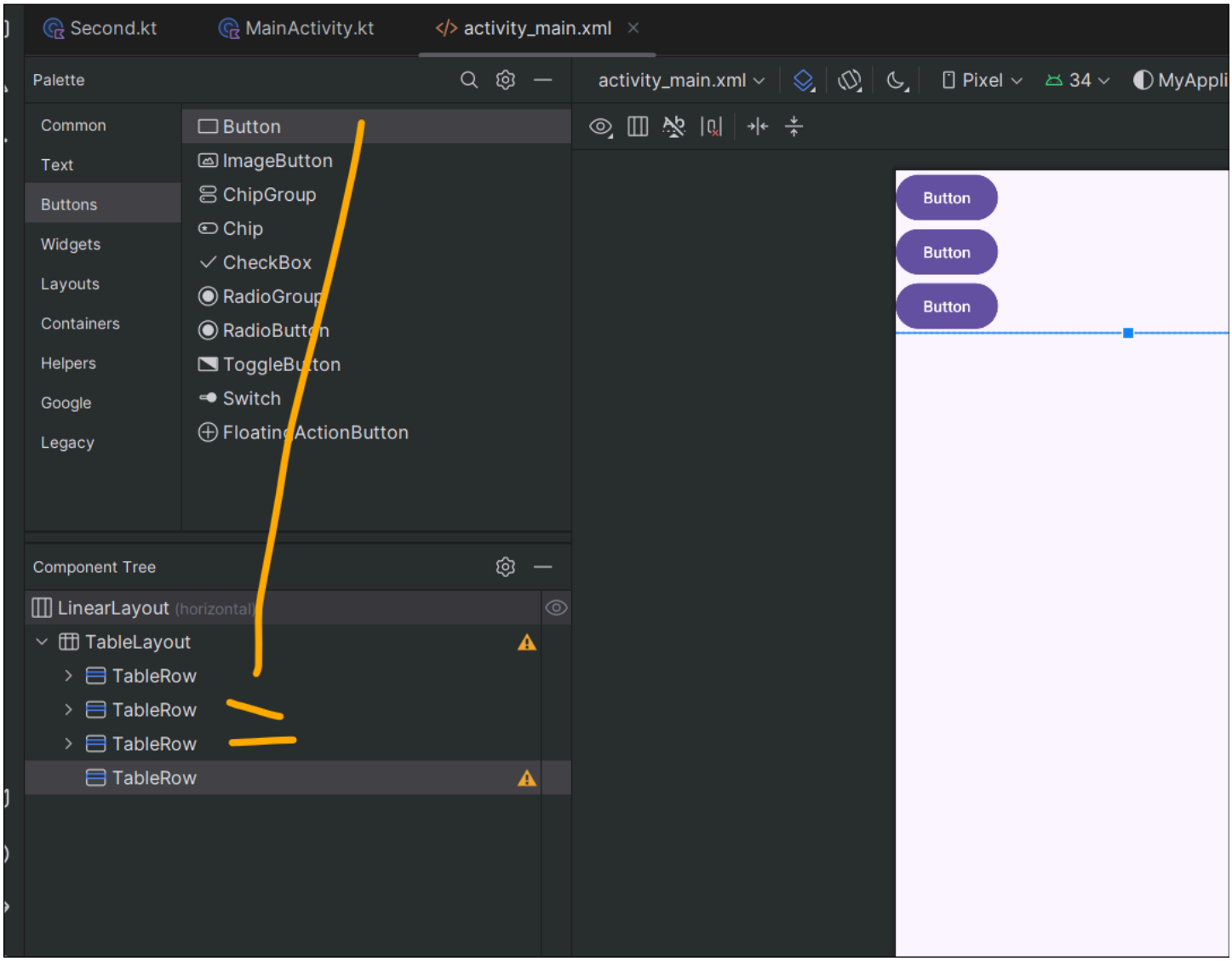
For Table Layout add the Linear Layout as a holding layout for table.
Then Drag and drop the table in White section on the UI.
You will get a Table Layout with some Table Row added by default.

For that see the component in the Component Tree Tab below the Palette Tab.

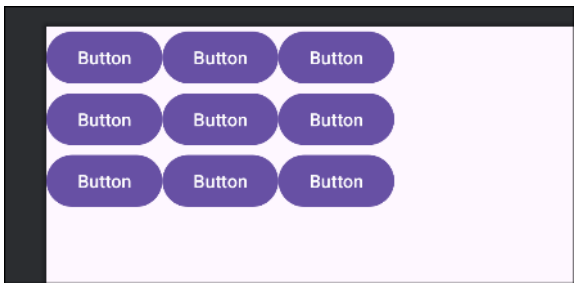


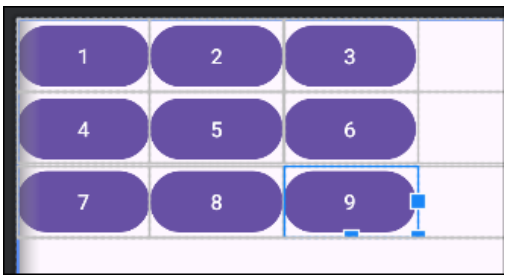
As the UI will not be draggable for adding buttons properly. Use the Component Tree section to drag the button into the Table Row from the Palette and the UI will show the buttons properly.

Drag 3 buttons in each row of a Table Row. Delete other extra table rows.



Drag and drop more buttons and rename buttons to numbers





Activity_main.xml code will look like this.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TableLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <TableRow
           android:layout_width="match_parent"
           android:layout_height="match_parent" >

           <Button
               android:id="@+id/button1"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="1" />

           <Button
               android:id="@+id/button2"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="2" />

           <Button
               android:id="@+id/button3"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="3" />
       </TableRow>

       <TableRow
           android:layout_width="match_parent"
           android:layout_height="match_parent" >

           <Button
               android:id="@+id/button4"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="4" />

           <Button
               android:id="@+id/button5"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="5" />

           <Button
               android:id="@+id/button6"
               android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:text="6" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <Button
                android:id="@+id/button7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="7" />

            <Button
                android:id="@+id/button8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="8" />

            <Button
                android:id="@+id/button9"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="9" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </TableLayout>
</LinearLayout>
```

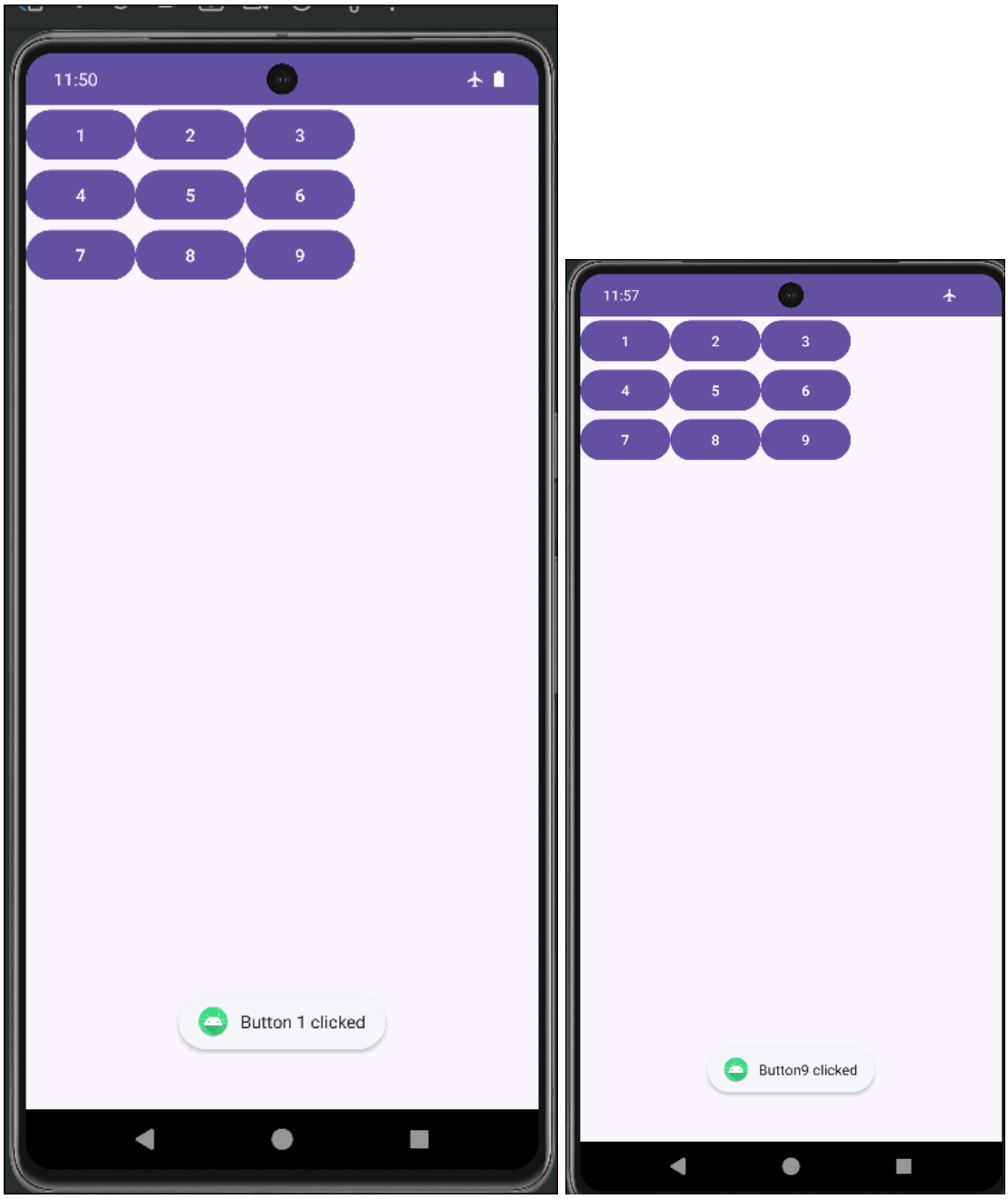
Now will add events to all the button in Activity_main.kt

```
package com.r.table_view

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
import org.jetbrains.anko.toast
import android.widget.Button


class MainActivity : AppCompatActivity() {

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

  val button1 = findViewById<Button>(R.id.button1)
    button1.setOnClickListener {
      Toast.makeText(this, "Button1 clicked", Toast.LENGTH_LONG).show();
    }
    val button2 = findViewById<Button>(R.id.button2)
    button2.setOnClickListener {
```

```
        Toast.makeText(this, "Button2 clicked", Toast.LENGTH_LONG).show();
    }
    val button3 = findViewById<Button>(R.id.button3)
    button3.setOnClickListener {
        Toast.makeText(this, "Button3 clicked", Toast.LENGTH_LONG).show();
    }


    val button4 = findViewById<Button>(R.id.button4)
    button4.setOnClickListener {
        Toast.makeText(this, "Button4 clicked", Toast.LENGTH_LONG).show();
    }

    val button5 = findViewById<Button>(R.id.button5)
    button5.setOnClickListener {
        Toast.makeText(this, "Button5 clicked", Toast.LENGTH_LONG).show();
    }
    val button6 = findViewById<Button>(R.id.button6)
    button6.setOnClickListener {
        Toast.makeText(this, "Button6 clicked", Toast.LENGTH_LONG).show();
    }
    val button7 = findViewById<Button>(R.id.button7)
    button7.setOnClickListener {
        Toast.makeText(this, "Button7 clicked", Toast.LENGTH_LONG).show();
    }
    val button8 = findViewById<Button>(R.id.button8)

    button8.setOnClickListener {
        Toast.makeText(this, "Button8 clicked", Toast.LENGTH_LONG).show();
    }
    val button9 = findViewById<Button>(R.id.button9)
    button9.setOnClickListener {
        Toast.makeText(this, "Button9 clicked", Toast.LENGTH_LONG).show();
    }


  }
}
```

output:

As clicking on the button the toast message will be displayed.



4. Frame Layout: For Frame layout add the code below .

activity_main.xml

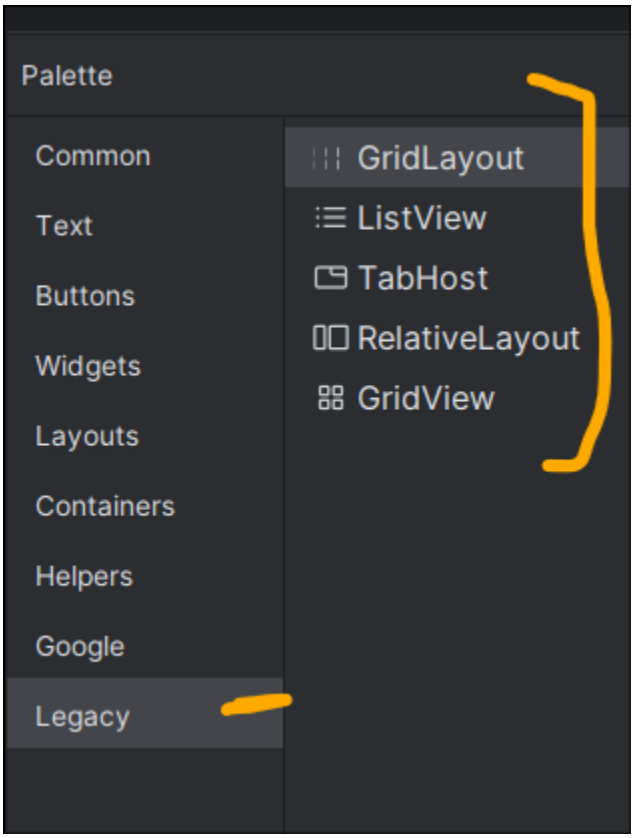
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/coding"
        android:scaleType="centerCrop"/>
    <TextView
        android:textSize="100dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:gravity="center"
        android:textColor="@color/black"
        android:layout_marginTop="220dp"
        />

</FrameLayout>
```

MainActivity.kt no change is required

Output:

5. List View Layout:



As you can see in the above image the List View, Grid View, Relative Layout View are in the legacy tab in Palette.

Here we will try out List view.

First we will have to have a list of items to be displayed. For that open strings.xml file and then add the Array List there and will populate the UI with that list.
String.xml

```
<resources>
  <string name="app_name">list</string>

  <array name="insert_list">
    <item>one</item>
    <item>two</item>
    <item>three</item>
    <item>four</item>
    <item>five</item>
    <item>six</item>
    <item>seven</item>
    <item>eight</item>
```

```
    <item>nine</item>
    <item>ten</item>
  </array>

</resources>
```

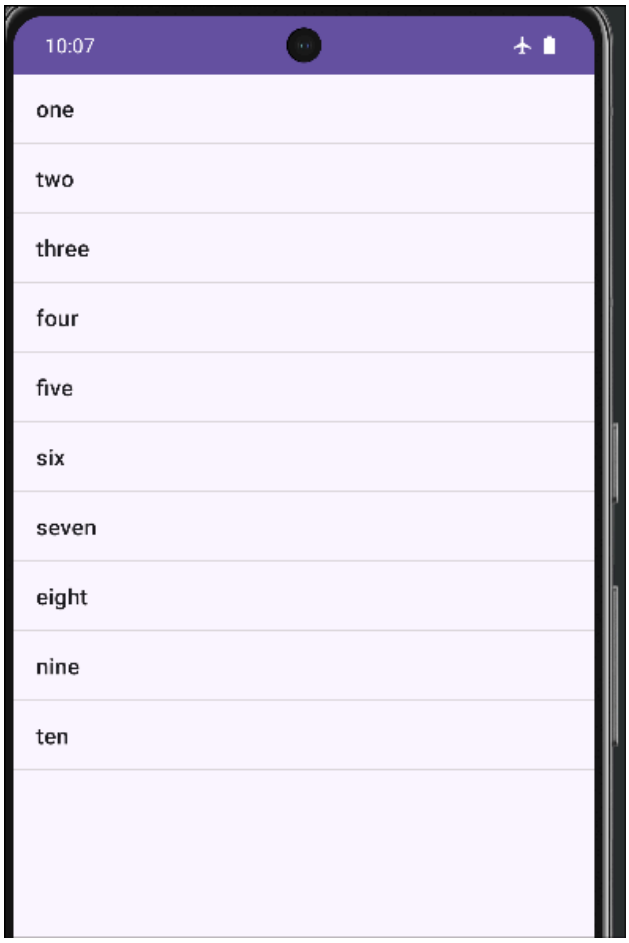
Here the array name is **insert_list.**

Add then add a List View in Activity_main.xml with the below code. Here use the array name to set the list view. Then run the application.

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:entries="@array/insert_list">

</ListView>
```

Output:



6. Grid layout: Grid Layout is similar to Table Layout Here the things are arranged in a grid.

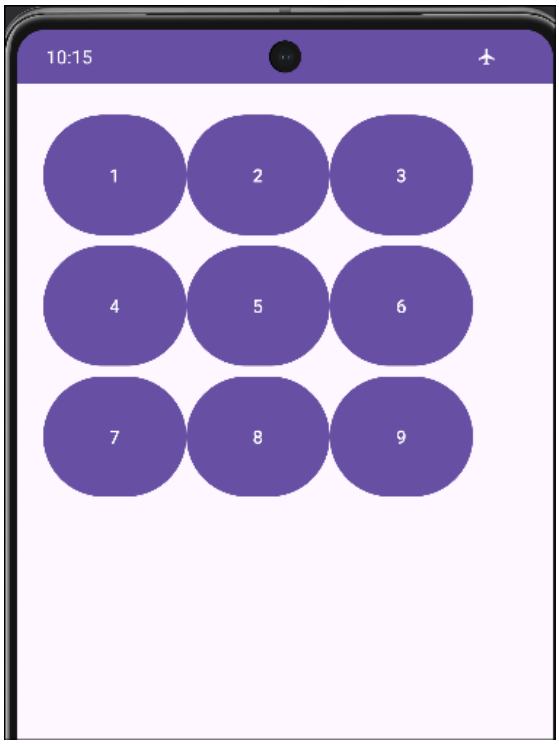
In the below code of activity_main.xml the **rowCount** and **columnCount** is set, which gives the settings of how many rows and columns the grid will contain. Here the items in the grid fill the layout from left top right from top to bottom.

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

```
    android:rowCount="3"
    android:columnCount="3"
    android:padding="20dp">

    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="1"/>

    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="2"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="3"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="4"/>

    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="5"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="6"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="7"/>

    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="8"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="9"/>

</GridLayout>
```

mainActvity.kt: No Code change is required.

output:


10:15
1
2
3
4
5
6
7
8
9

PRACTICAL 5

Programming UI elements

AppBar, Fragments, UI Components

Here to add the App Bar in the app we have to make changes in the AndroidManifest.xml file.

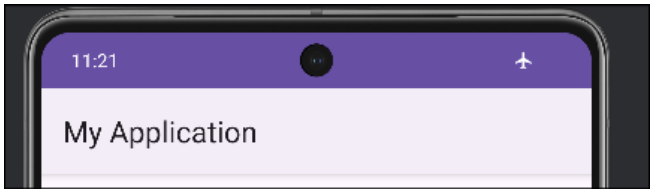
Currently the parent attribute of the app would be having value `Theme.Material3.DayNight.NoActionBar`

change it to or remove the `NoActionBar` from it.

```
<resources xmlns:tools="http://schemas.android.com/tools">
   <!-- Base application theme. -->
   <style name="Base.Theme.MyApplication" parent="Theme.Material3.DayNight">
   </style>
   <style name="Theme.MyApplication" parent="Base.Theme.MyApplication" />
</resources>
```

Run the app then you will be able to see the app bar in output

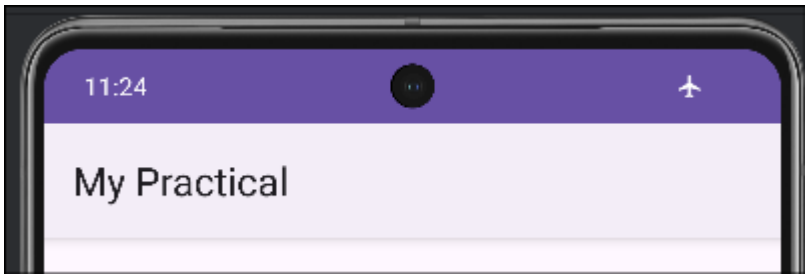
Output:



You can change the name of the app in strings.xml. This change will rename the application that is named. Change it to the desired name.

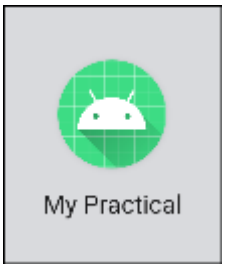
strings.xml

```
<resources>
   <string name="app_name">
<!--        My Application-->
      My Practical
   </string>
</resources>
```

Output: 

Here in the app list of the emulator the name of the App also changes as the strings.xml variable **app_name** is used at one more place which is in AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">

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

Output : 

Fragments

// TODO

Ui Components // Button Text , etc

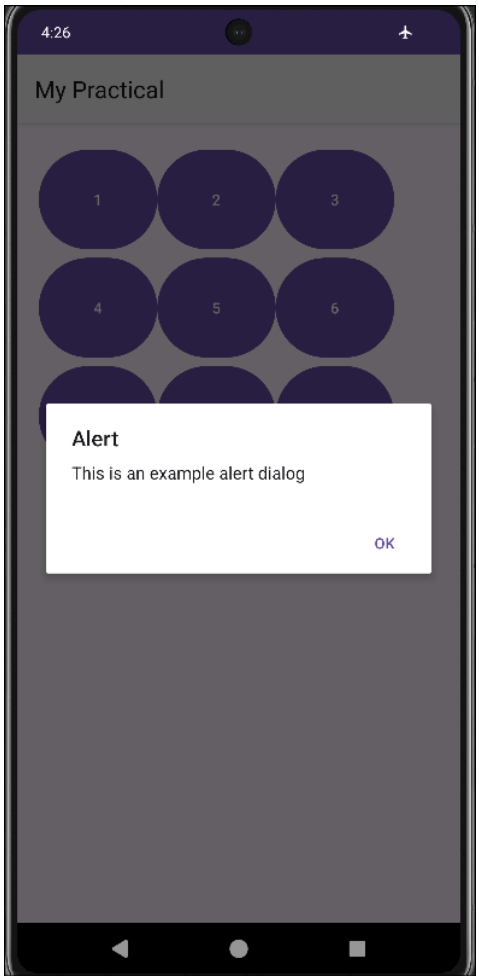
PRACTICAL 6

Programming menus, dialog, dialog fragments

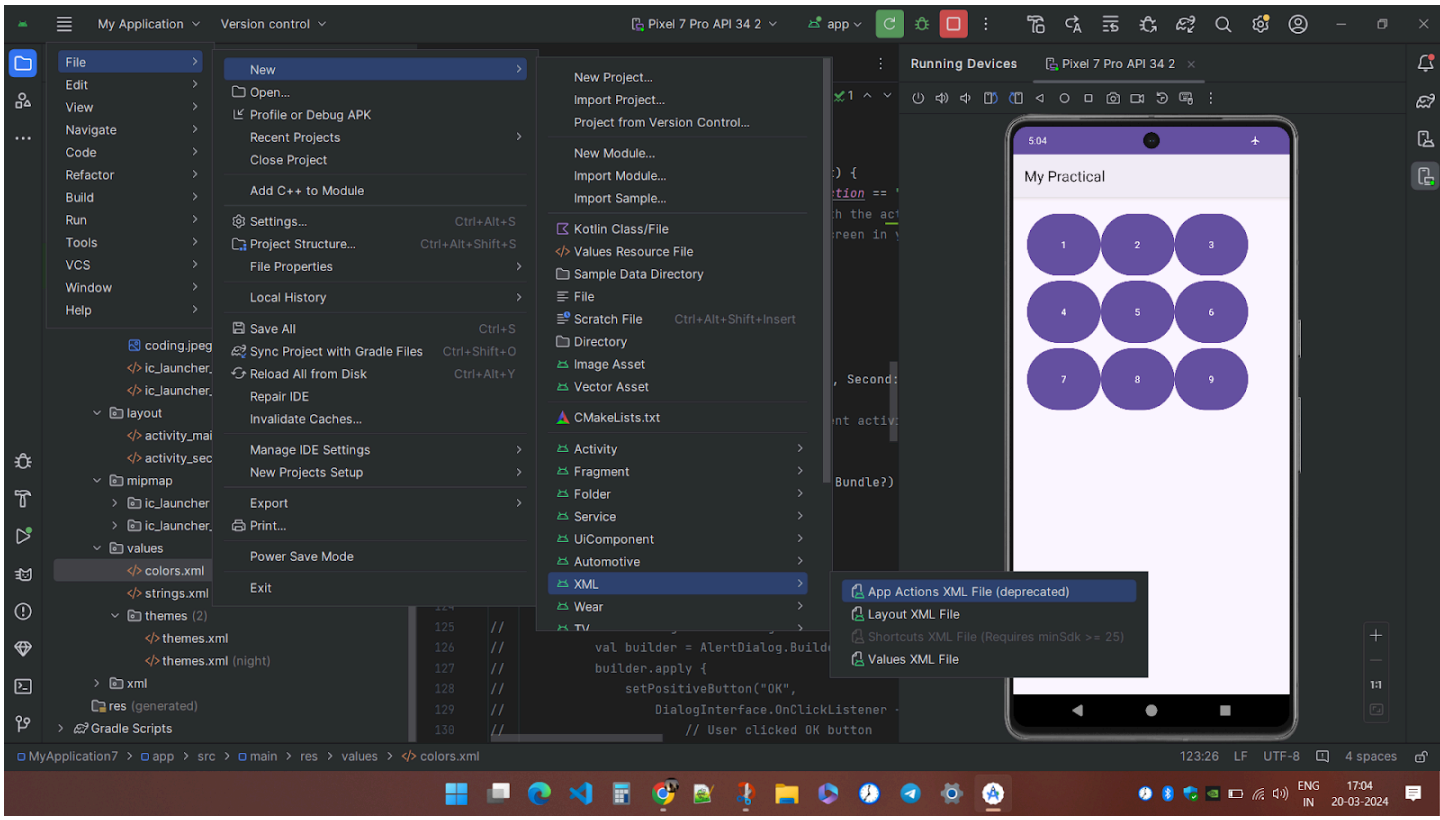
Alert Dialog

```
package com.example.myapplication

import android.content.Context
import android.os.Bundle
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
// Method to show the alert dialog
private fun showAlert(context:Context) {
   // Instantiate an AlertDialog.Builder with its constructor
   val builder = AlertDialog.Builder(context)

   // Chain together various setter methods to set the dialog characteristics
   builder.setTitle("Alert")
       .setMessage("This is an example alert dialog")
       .setPositiveButton("OK") { dialog, id ->
           // User clicked OK button
           dialog.dismiss() // Dismiss the dialog
       }

   // Get the AlertDialog from create()
   val dialog = builder.create()

   // Show the alert dialog
   dialog.show()
}
override fun onCreate(savedInstanceState: Bundle?) {
   super.onCreate(savedInstanceState)
   setContentView(R.layout.activity_main)

   showAlert(this)
}
}
```

output:


4:26
My Practical
1
2
3
4
5
6
Alert
This is an example alert dialog
OK

Menu:



menu.xml:

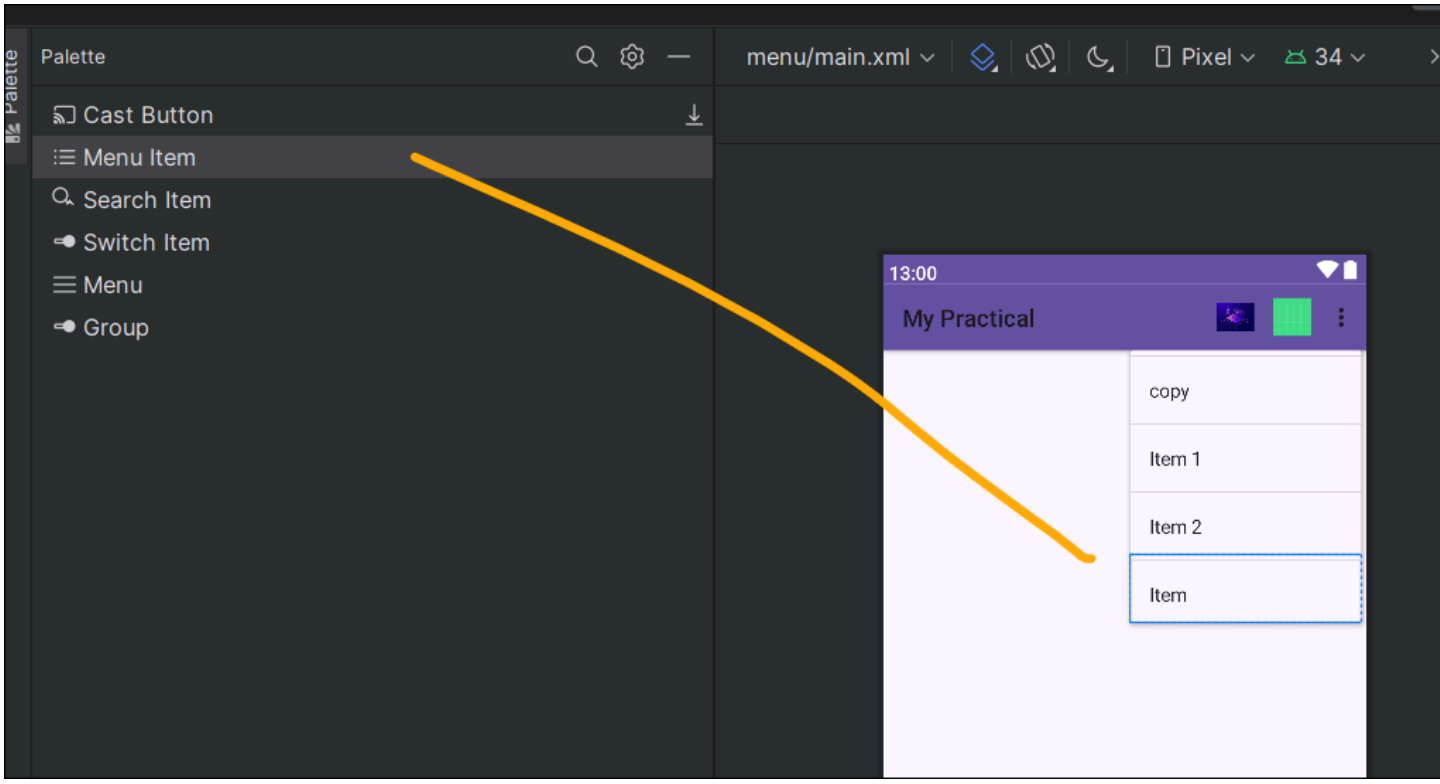
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- action button for search -->
    <item android:title="search"
        android:id="@+id/search_button"
        android:orderInCategory="100"
        app:showAsAction="ifRoom"
        android:icon="@drawable/coding"/>

    <!-- action button for refresh -->
    <item android:title="refresh"
        android:id="@+id/refreh"
        android:orderInCategory="100"
        app:showAsAction="ifRoom"
        android:icon="@drawable/ic_launcher_background"/>

    <!-- action button for copy -->
    <item android:title="copy"
        android:id="@+id/Copy"
        android:orderInCategory="100"
        app:showAsAction="ifRoom"
        android:icon="@drawable/ic_launcher_foreground"/>
    <item
        android:orderInCategory="101"
```

```
        app:showAsAction="ifRoom"

        android:title="Item 1 " />
    <item
        android:orderInCategory="102"
        android:title="Item 2" />
</menu>
```

Here in the above code the main settings are bold. `showAsAction` , id are the 2 main things to have a look upon.

The id will be used to set the action in the Mainactivity.kt

You can drag and drop a menu item also. Then edit it in the code view of it.



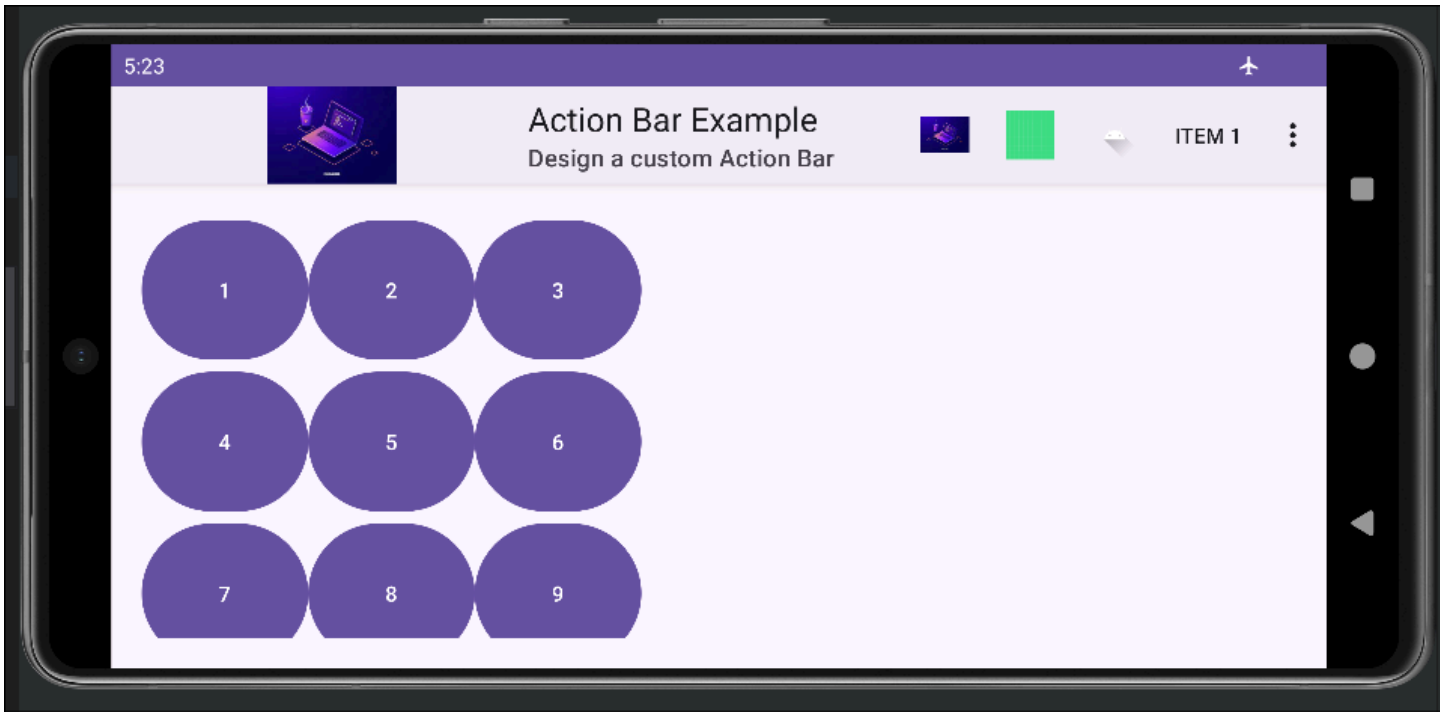
MainActivity.kt:

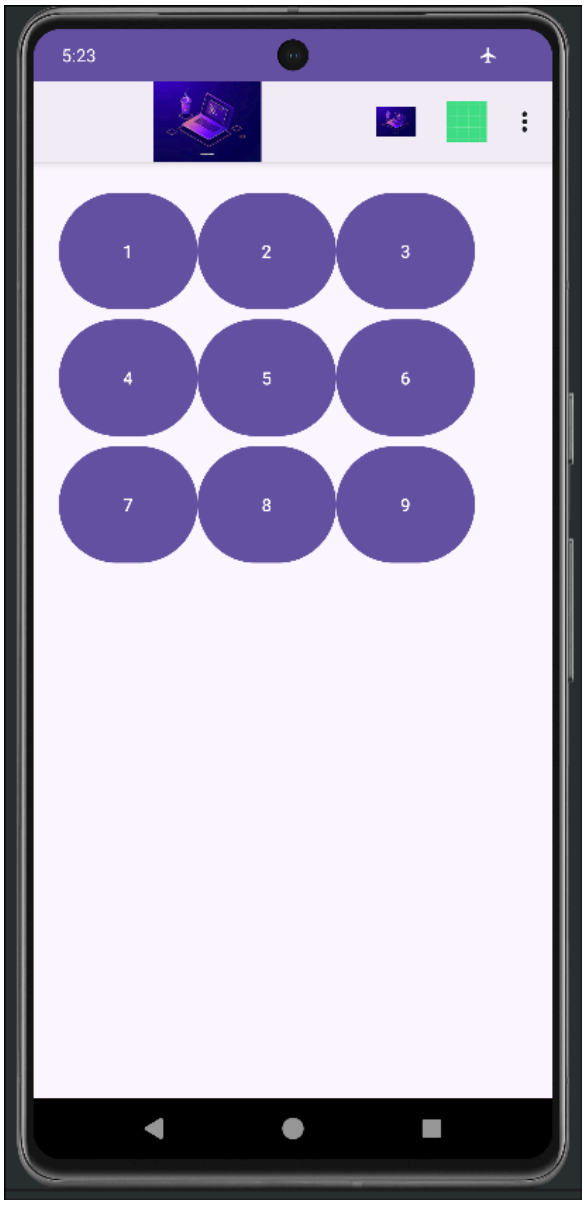
```
package com.example.myapplication

import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity


class MainActivity : AppCompatActivity() {
```

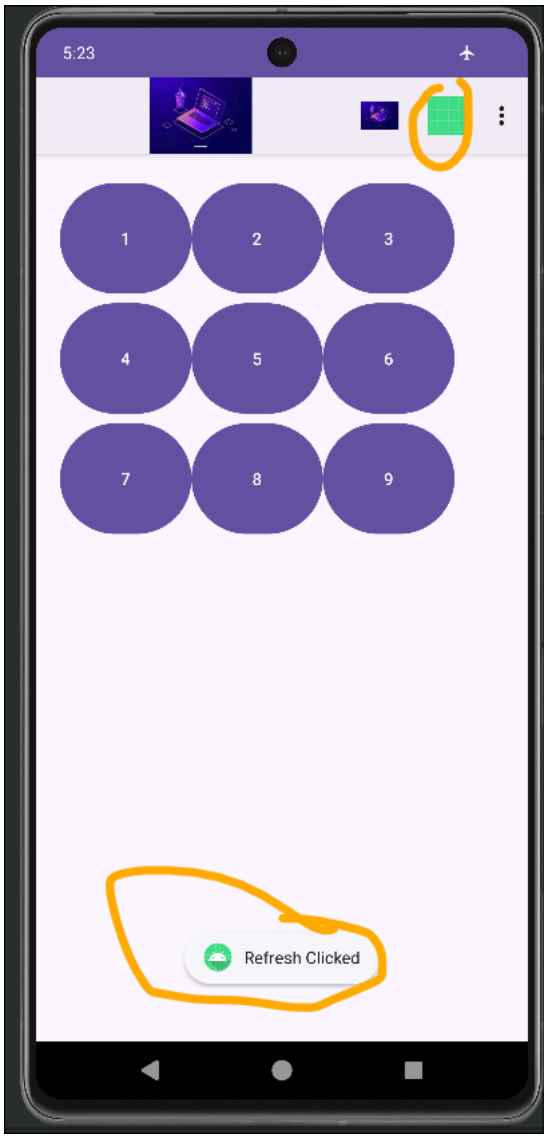
```
override fun onCreate(savedInstanceState: Bundle?) {
   super.onCreate(savedInstanceState)
   setContentView(R.layout.activity_main)

   // calling this activity's function to
   // use ActionBar utility methods
   val actionBar = supportActionBar

   // providing title for the ActionBar
   actionBar!!.title = "Action Bar Example"

   // providing subtitle for the ActionBar
   actionBar.subtitle = "Design a custom Action Bar"

   // adding icon in the ActionBar
   actionBar.setIcon(R.drawable.coding)

   // methods to display the icon in the ActionBar
   actionBar.setDisplayUseLogoEnabled(true)
   actionBar.setDisplayShowHomeEnabled(true)
}

// method to inflate the options menu when
// the user opens the menu for the first time
override fun onCreateOptionsMenu(menu: Menu): Boolean {
   menuInflater.inflate(R.menu.main, menu)
   return super.onCreateOptionsMenu(menu)
}

// methods to control the operations that will
// happen when user clicks on the action buttons
override fun onOptionsItemSelected(item: MenuItem): Boolean {
   when (item.itemId) {
       R.id.search_button -> Toast.makeText(this, "Search Clicked",
Toast.LENGTH_SHORT).show()
       R.id.refreh -> Toast.makeText(this, "Refresh Clicked",
Toast.LENGTH_SHORT).show()
       R.id.Copy -> Toast.makeText(this, "Copy Clicked", Toast.LENGTH_SHORT).show()
   }
   return super.onOptionsItemSelected(item)
}

}
```

Output:

Landscape mode


5:23
Action Bar Example
Design a custom Action Bar
ITEM 1
1
2
3
4
5
6
7
8
9


5:23
1
2
3
4
5
6
7
8
9


5:23
1
2
3
4
5
6
7
8
9
Refresh Clicked

PRACTICAL 7

Programs on Intents, Events Listeners

Note: Refer Table layout code for Events Listeners and for Intent GUI code

PRACTICAL 8

Programs on Services, notification and broadcast receivers

Same as practical 1

PRACTICAL 9

Database Programming with SQLite

activity_main.xml:

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

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="SQLite Tutorial - User Management"
android:textSize="20dp"
android:padding="10dp" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
<EditText
android:id="@+id/edittext_userid"
android:hint="User ID"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<EditText
android:id="@+id/edittext_name"
android:hint="User Name"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<EditText
android:id="@+id/edittext_age"
android:hint="User Age"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">
<Button
android:id="@+id/button_add_user"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:onClick="addUser"
```

```
android:text="Add" />

<Button
android:id="@+id/button_delete_user"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:onClick="deleteUser"
android:text="Delete" />

<Button
android:id="@+id/button_show_all"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:onClick="showAllUsers"
android:text="Show All" />
</LinearLayout>
<TextView
android:id="@+id/textview_result"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<LinearLayout
android:id="@+id/ll_entries"
android:padding="15dp"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content"></LinearLayout>
</LinearLayout>
```

UserModel.kt:

```
package com.tutorialkart.sqlitetutorial

class UserModel(val userid: String, val name: String, val age: String)
```

DBContract.kt

```
package com.tutorialkart.sqlitetutorial

import android.provider.BaseColumns

object DBContract {

  /* Inner class that defines the table contents */
  class UserEntry : BaseColumns {
    companion object {
      val TABLE_NAME = "users"
      val COLUMN_USER_ID = "userid"
      val COLUMN_NAME = "name"
      val COLUMN_AGE = "age"
    }
  }
}
```

UserDBHelper.kt:

```
package com.tutorialkart.sqlitetutorial

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteConstraintException
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteException
import android.database.sqlite.SQLiteOpenHelper

import java.util.ArrayList

class UsersDBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(SQL_CREATE_ENTRIES)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        // This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
        db.execSQL(SQL_DELETE_ENTRIES)
        onCreate(db)
    }

    override fun onDowngrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        onUpgrade(db, oldVersion, newVersion)
    }

    @Throws(SQLiteConstraintException::class)
    fun insertUser(user: UserModel): Boolean {
        // Gets the data repository in write mode
        val db = writableDatabase

        // Create a new map of values, where column names are the keys
        val values = ContentValues()
        values.put(DBContract.UserEntry.COLUMN_USER_ID, user.userid)
        values.put(DBContract.UserEntry.COLUMN_NAME, user.name)
        values.put(DBContract.UserEntry.COLUMN_AGE, user.age)

        // Insert the new row, returning the primary key value of the new row
        val newRowId = db.insert(DBContract.UserEntry.TABLE_NAME, null, values)

        return true
    }

    @Throws(SQLiteConstraintException::class)
    fun deleteUser(userid: String): Boolean {
        // Gets the data repository in write mode
        val db = writableDatabase
        // Define 'where' part of the query.
        val selection = DBContract.UserEntry.COLUMN_USER_ID + " LIKE ?"
        // Specify arguments in placeholder order.
        val selectionArgs = arrayOf(userid)
        // Issue SQL statement.
        db.delete(DBContract.UserEntry.TABLE_NAME, selection, selectionArgs)
```

```
        return true
    }

    fun readUser(userid: String): ArrayList<UserModel> {
        val users = ArrayList<UserModel>()
        val db = writableDatabase
        var cursor: Cursor? = null
        try {
            cursor = db.rawQuery("select * from " + DBContract.UserEntry.TABLE_NAME + " WHERE " +
DBContract.UserEntry.COLUMN_USER_ID + "='" + userid + "'", null)
        } catch (e: SQLiteException) {
            // if table not yet present, create it
            db.execSQL(SQL_CREATE_ENTRIES)
            return ArrayList()
        }

        var name: String
        var age: String
        if (cursor!!.moveToFirst()) {
            while (cursor.isAfterLast == false) {
                name = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_NAME))
                age = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_AGE))

                users.add(UserModel(userid, name, age))
                cursor.moveToNext()
            }
        }
        return users
    }

    fun readAllUsers(): ArrayList<UserModel> {
        val users = ArrayList<UserModel>()
        val db = writableDatabase
        var cursor: Cursor? = null
        try {
            cursor = db.rawQuery("select * from " + DBContract.UserEntry.TABLE_NAME, null)
        } catch (e: SQLiteException) {
            db.execSQL(SQL_CREATE_ENTRIES)
            return ArrayList()
        }

        var userid: String
        var name: String
        var age: String
        if (cursor!!.moveToFirst()) {
            while (cursor.isAfterLast == false) {
                userid = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_USER_ID))
                name = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_NAME))
                age = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_AGE))

                users.add(UserModel(userid, name, age))
                cursor.moveToNext()
            }
        }
        return users
    }

    companion object {
        // If you change the database schema, you must increment the database version.
        val DATABASE_VERSION = 1
        val DATABASE_NAME = "FeedReader.db"
```

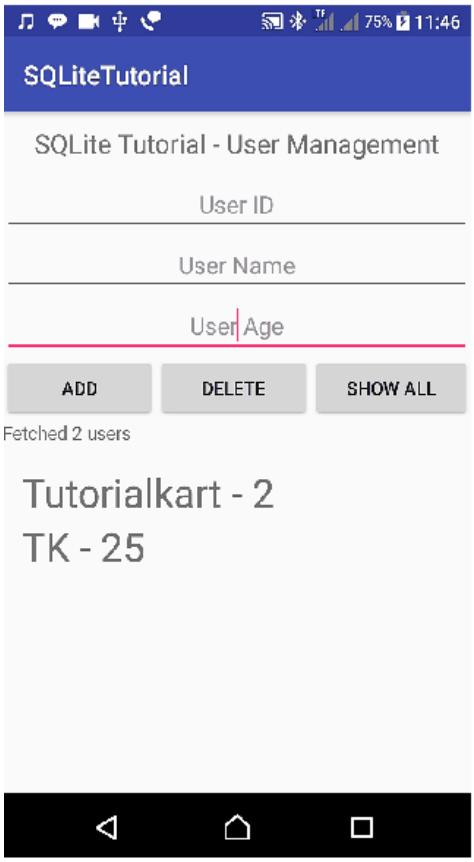
```
    private val SQL_CREATE_ENTRIES =
        "CREATE TABLE " + DBContract.UserEntry.TABLE_NAME + " (" +
                DBContract.UserEntry.COLUMN_USER_ID + " TEXT PRIMARY KEY," +
                DBContract.UserEntry.COLUMN_NAME + " TEXT," +
                DBContract.UserEntry.COLUMN_AGE + " TEXT)"

    private val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS " + DBContract.UserEntry.TABLE_NAME
}

}
```

MainActivity.kt:

```
package com.tutorialkart.sqlitetutorial

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.TextView
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    lateinit var usersDBHelper : UsersDBHelper

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

        usersDBHelper = UsersDBHelper(this)
    }

    fun addUser(v:View){
        var userid = this.edittext_userid.text.toString()
        var name = this.edittext_name.text.toString()
        var age = this.edittext_age.text.toString()
        var result = usersDBHelper.insertUser(UserModel(userid = userid,name = name,age = age))
        //clear all edittext s
        this.edittext_age.setText("")
        this.edittext_name.setText("")
        this.edittext_userid.setText("")
        this.textview_result.text = "Added user : "+result
        this.ll_entries.removeAllViews()
    }

    fun deleteUser(v:View){
        var userid = this.edittext_userid.text.toString()
        val result = usersDBHelper.deleteUser(userid)
        this.textview_result.text = "Deleted user : "+result
        this.ll_entries.removeAllViews()
    }

    fun showAllUsers(v:View){
        var users = usersDBHelper.readAllUsers()
        this.ll_entries.removeAllViews()
        users.forEach {
            var tv_user = TextView(this)
            tv_user.textSize = 30F
```

```
            tv_user.text = it.name.toString() + " - " + it.age.toString()
            this.ll_entries.addView(tv_user)
        }
        this.textview_result.text = "Fetched " + users.size + " users"
    }
}
```

output:



PRACTICAL 10

Programming Media API and Telephone API

MainActivity.kt

```
package com.example.myapplication

import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.net.Uri
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat

class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
   super.onCreate(savedInstanceState)
   setContentView(R.layout.activity_main)
   val button = findViewById<Button>(R.id.button)
   val edittext = findViewById<EditText>(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
        Log.d("d", "Starting")
        phone_intent.data = Uri.parse("tel:$phone_number")

        // start Intent
        if(ActivityCompat.checkSelfPermission(this,
Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED){
            Log.d("d", "Permission not given")
        }else
        {
            startActivity(phone_intent)
        }
    })
}
```

activity_main.xml

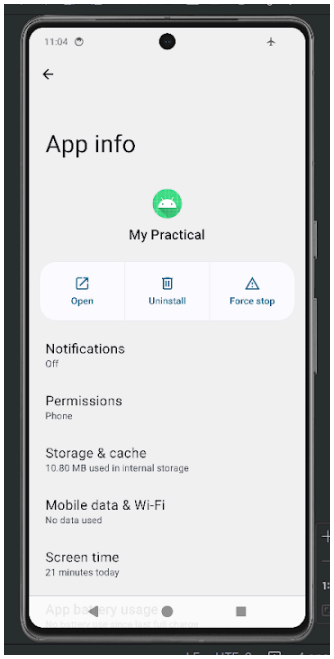
```
<?xml version="1.0" encoding="utf-8"?>


<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!-- Edit text for phone number -->
    <EditText
        android:id="@+id/editText"
```

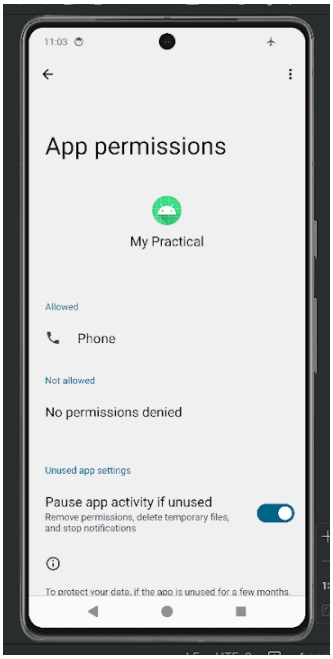
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:hint="10 digit mobile number"
        android:inputType="text" />

    <!-- 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>
```

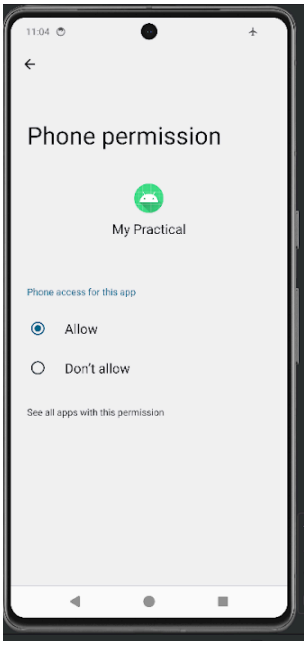
Android Manifest.xml

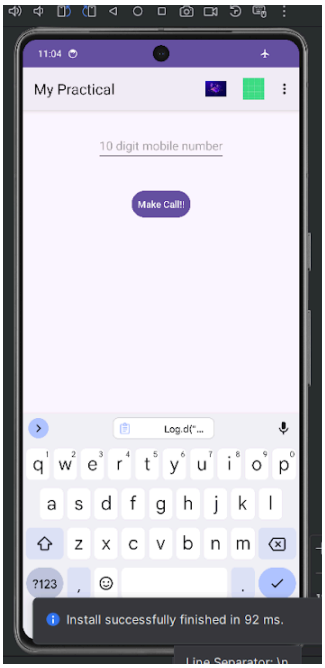
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
<uses-permission android:name="android.permission.CALL_PHONE" />



<application
…… .. … .. … ….
</application>
```

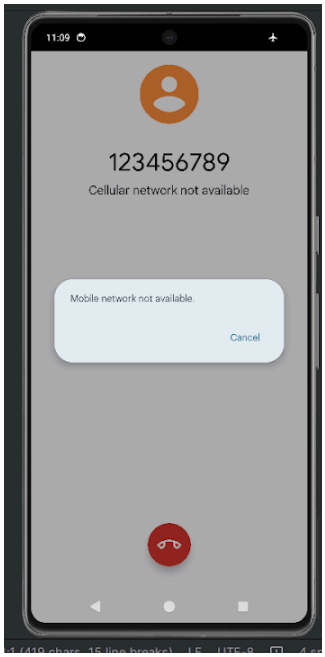
```
<uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
</manifest>
```











PRACTICAL 11

Programming Security and permissions

1. **Create a new project in Android studio**

Then open the acivity_main.xml then add a button

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <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="Get Permissions" />
</RelativeLayout>
```

2. **An app must publicize the permissions it requires by including <uses-permission> tags in the app manifest.**

```
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
```

AndroidManifest.xml | MainActivity.kt | ManagePermissions.kt

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

    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CALENDAR"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="My Practical"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
```

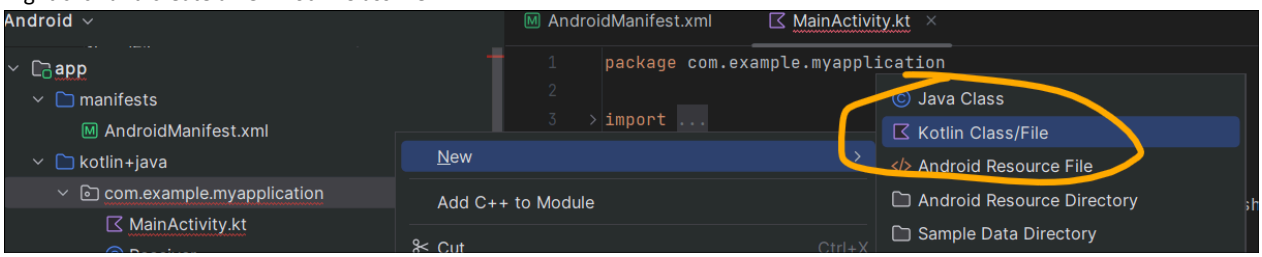
3. MainActivity.kt

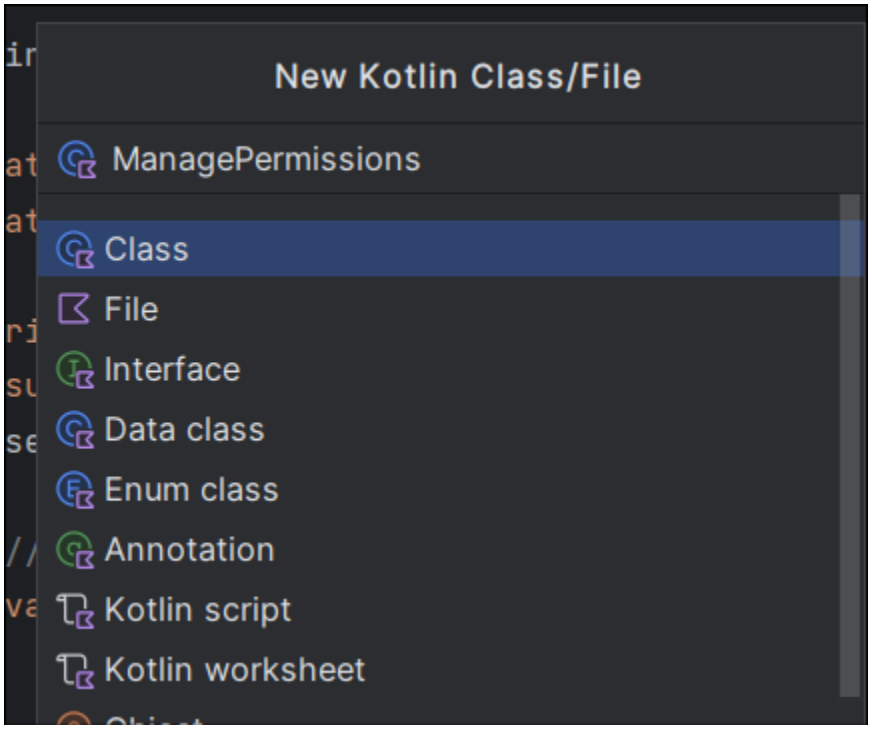
```kotlin
package com.example.myapplication

import android.Manifest
import android.content.Context
import android.os.Build
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

// Extension function to show toast message
fun Context.toast(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}


class MainActivity : AppCompatActivity() {
```

```
    private val PermissionsRequestCode = 123
    private lateinit var managePermissions: ManagePermissions

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

        // Initialize a list of required permissions to request runtime
        val list = listOf<String>(
            Manifest.permission.CAMERA,
            Manifest.permission.READ_CALENDAR,
            Manifest.permission.READ_CONTACTS,
            Manifest.permission.SEND_SMS,
            Manifest.permission.READ_MEDIA_IMAGES,


            )

        // Initialize a new instance of ManagePermissions class
        managePermissions = ManagePermissions(this, list,
PermissionsRequestCode)
        val button = findViewById<Button>(R.id.button)
        // Button to check permissions states
        button.setOnClickListener {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
                managePermissions.checkPermissions()
        }
    }


    // Receive the permissions request result
    override fun onRequestPermissionsResult(
        requestCode: Int, permissions: Array<String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        when (requestCode) {
            PermissionsRequestCode -> {
                val isPermissionsGranted = managePermissions
                    .processPermissionsResult(requestCode, permissions,
grantResults)

                if (isPermissionsGranted) {
                    // Do the task now
                    toast("Permissions granted.")
                } else {
                    toast("Permissions denied.")
                }
                return
            }
        }
    }
}
```

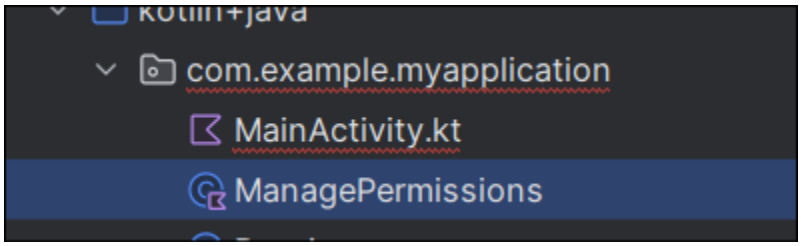
4. Create a New Kotlin Class

Right click and create a new Kotlin Class file





Class file is generated

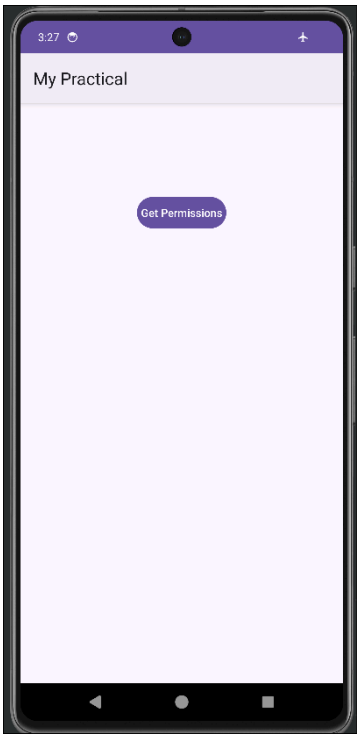


5. Write the following code in the Class File

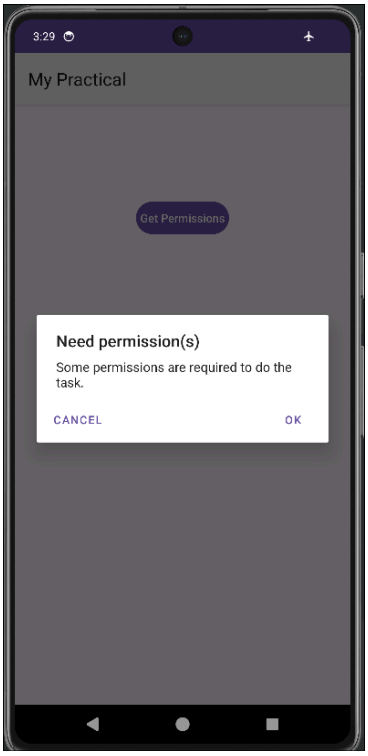
```
package com.example.myapplication


import android.app.Activity
import android.app.AlertDialog
import android.content.pm.PackageManager
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat



class ManagePermissions(val activity: Activity,val list: List<String>,val
code:Int) {

    // Check permissions at runtime
    fun checkPermissions() {
        if (isPermissionsGranted() != PackageManager.PERMISSION_GRANTED) {
            showAlert()
        } else {
            activity.toast("Permissions already granted.")
        }
    }


    // Check permissions status
    private fun isPermissionsGranted(): Int {
        // PERMISSION_GRANTED : Constant Value: 0
        // PERMISSION_DENIED : Constant Value: -1
        var counter = 0;
        for (permission in list) {
            counter += ContextCompat.checkSelfPermission(activity,
permission)
        }
        return counter
    }


    // Find the first denied permission
    private fun deniedPermission(): String {
        for (permission in list) {
```

```
            if (ContextCompat.checkSelfPermission(activity, permission)
                == PackageManager.PERMISSION_DENIED) return permission
        }
        return ""
    }


    // Show alert dialog to request permissions
    private fun showAlert() {
        val builder = AlertDialog.Builder(activity)
        builder.setTitle("Need permission(s)")
        builder.setMessage("Some permissions are required to do the
task.")
        builder.setPositiveButton("OK", { dialog, which ->
requestPermissions() })
        builder.setNeutralButton("Cancel", null)
        val dialog = builder.create()
        dialog.show()
    }


    // Request the permissions at run time
    private fun requestPermissions() {
        val permission = deniedPermission()
        if (ActivityCompat.shouldShowRequestPermissionRationale(activity,
permission)) {
            // Show an explanation asynchronously
            activity.toast("Should show an explanation.")
        } else {
            ActivityCompat.requestPermissions(activity,
list.toTypedArray(), code)
        }
    }


    // Process permissions result
    fun processPermissionsResult(requestCode: Int, permissions:
Array<String>,
                                 grantResults: IntArray): Boolean {
        var result = 0
        if (grantResults.isNotEmpty()) {
            for (item in grantResults) {
                result += item
            }
        }
        if (result == PackageManager.PERMISSION_GRANTED) return true
        return false
    }
}
```

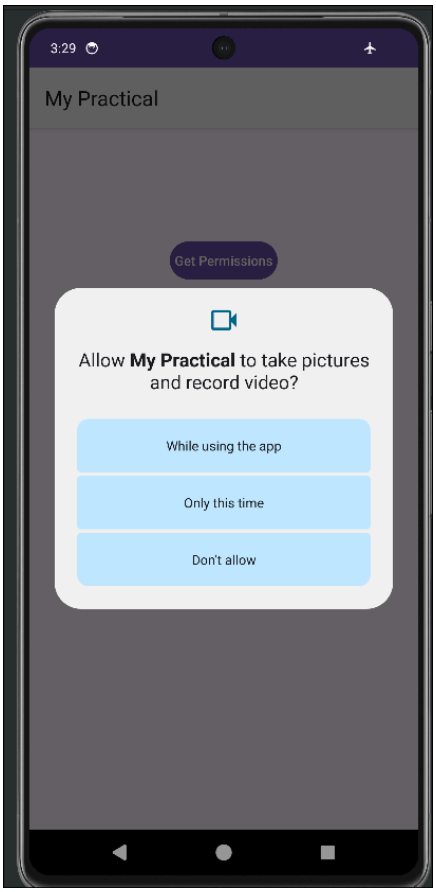
Output:



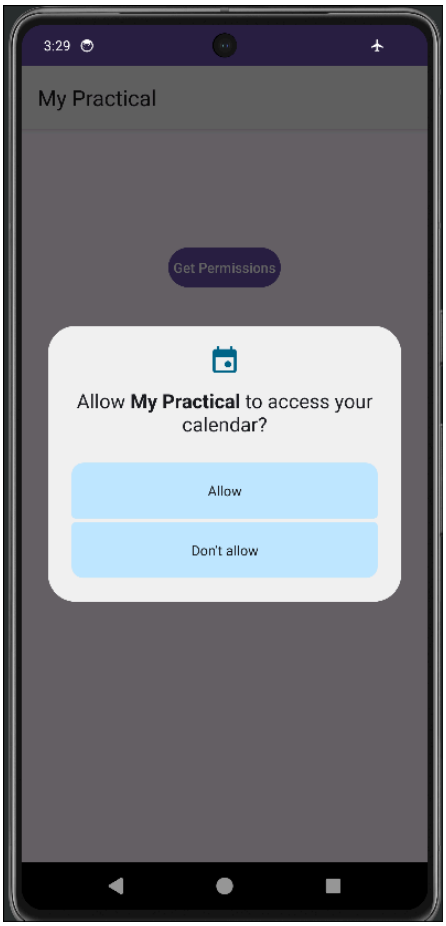
Then Click on the get permissions then see the popups coming



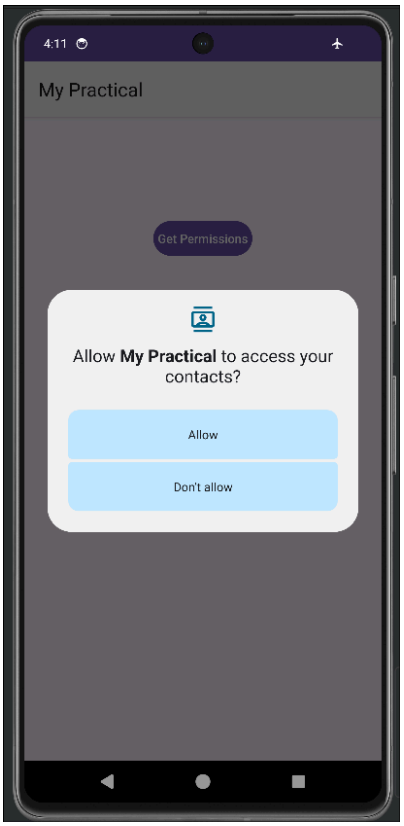
Then click oK

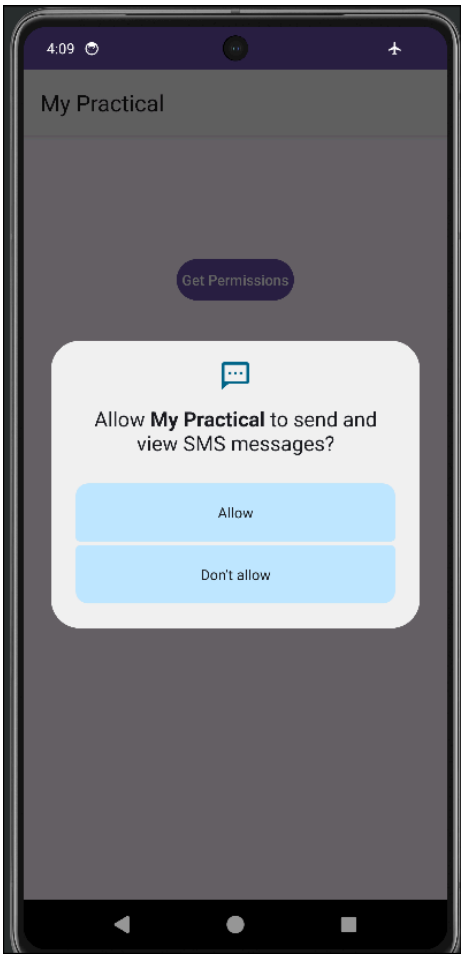


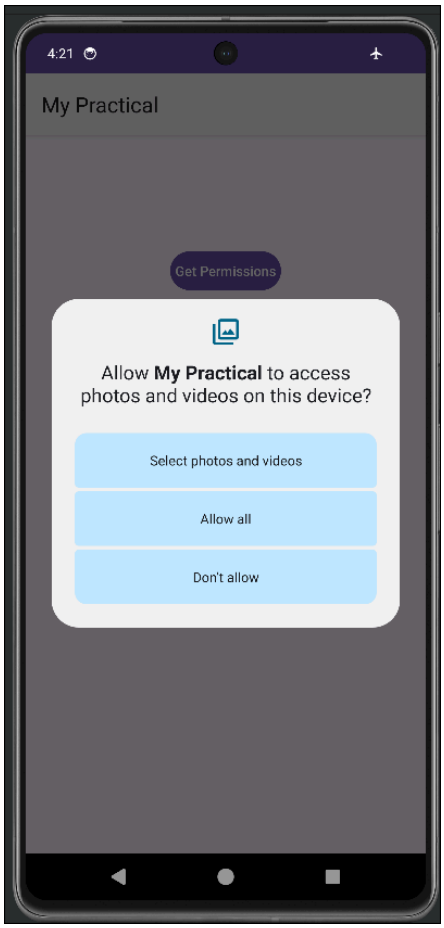
Click While using this app

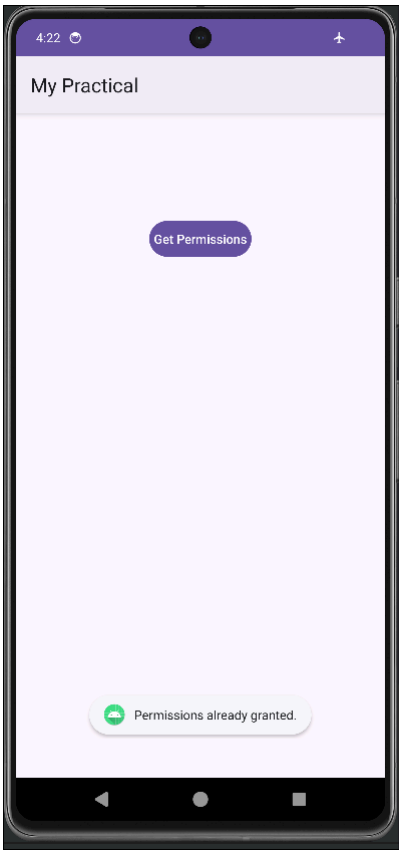

3:29
My Practical
Get Permissions
Allow My Practical to access your calendar?
Allow
Don't allow

Click Allow


4:11
My Practical
Get Permissions
Allow My Practical to access your contacts?
Allow
Don't allow


4:09
My Practical
Get Permissions
Allow **My Practical** to send and view SMS messages?
Allow
Don't allow


4:21
My Practical
Get Permissions
Allow **My Practical** to access photos and videos on this device?
Select photos and videos
Allow all
Don't allow


4:22
My Practical
Get Permissions
Permissions already granted.