

PRACTICAL 1

A) Write a program using Kotlin to implement control structures and loops.

If expression

```
fun main() {  
    val a = 2  
    val b = 3  
    var max = a  
    if (a < b) max = b  
    print(max)
```

Output:

3

```
// With else  
fun main() {  
    val a = 2  
    val b = 3  
    var max = a  
    if (a > b) {  
        max = a  
    }  
    else {  
        max = b  
    }  
}
```

Output:

3

```
// As expression
fun main() {
    val a = 2
    val b = 3
    var max = a

    max = if (a > b) a else b
}
```

Output:

3

```
// You can also use `else if` in expressions:
val maxLimit = 1
val maxOrLimit = if (maxLimit > a) maxLimit else if (a > b) a else b

println("max is $max")
println("maxOrLimit is $maxOrLimit")
}
```

Output:

max is 3
maxOrLimit is 3

When expressions and statements

```
fun main() {
    val x = 2
    when (x) {
        1 -> print("x == 1")
        2 -> print("x == 2")
        else -> print("x is neither 1 nor 2")
    }
    // x == 2
}
```

Output: x == 2

Combine when Conditions

```
fun main(args: Array<String>) {  
    val day = 2  
  
    when (day) {  
        1, 2, 3, 4, 5 -> println("Weekday")  
        else -> println("Weekend")  
    }  
}
```

Output:

Weekday

Expression in when Conditions

```
fun main(args: Array<String>) {  
    val x = 20  
    val y = 10  
    val z = 10  
  
    when (x) {  
        (y+z) -> print("y + z = x = $x")  
        else -> print("Condition is not satisfied")  
    }  
}
```

Output:

y + z = x = 20

Kotlin when with block of code

```
fun main(args: Array<String>) {  
    val day = 2  
  
    when (day) {  
        1 -> {  
            println("First day of the week")  
            println("Monday")  
        }  
        2 -> {  
            println("Second day of the week")  
            println("Tuesday")  
        }  
        3 -> {  
            println("Third day of the week")  
            println("Wednesday")  
        }  
        4 -> println("Thursday")  
        5 -> println("Friday")  
        6 -> println("Saturday")  
        7 -> println("Sunday")  
        else -> println("Invalid day.")  
    }  
}
```

Output:

Second day of the week
Tuesday

While loop

```
var i = 0
while (i < 5) {
  println(i)
  i++
}
```

Output:

```
0
1
2
3
4
```

Do while loop

```
var i = 0
do {
  println(i)
  i++
}
while (i < 5)
```

Output:

```
0
1
2
3
4
```

For Loop:

```
fun main(args: Array<String>) {  
    for (item in 1..5) {  
        println(item)  
    }  
}
```

Output:

```
1  
2  
3  
4  
5
```

Example 2:

```
fun main(args: Array<String>) {  
    for (item in 5 downTo 1 step 2) {  
        println(item)  
    }  
}
```

Output:

```
5  
3  
1
```

Example 3:

```
fun main() {  
    for (chars in 'a'..'g') {  
        println(chars)  
    }  
}
```

Output:

a
b
c
d
e
f
g

Iterate Through an Array

```
fun main(args: Array<String>) {  
    var fruits = arrayOf("Orange", "Apple", "Mango", "Banana")  
  
    for (item in fruits) {  
        println(item)  
    }  
}
```

Output:

Orange
Apple
Mango
Banana

Will iterate through the array using its index.

```
fun main(args: Array<String>) {  
    var fruits = arrayOf("Orange", "Apple", "Mango", "Banana")  
  
    for (index in fruits.indices) {  
        println(fruits[index])  
    }  
}
```

Output:

Orange

Apple

Mango

Banana

Break Statement

```
fun main(args: Array<String>) {  
    var i = 0;  
    while (i++ < 100) {  
        println(i)  
        if( i == 3 ){  
            break  
        }  
    }  
}
```

Output:

1

2

3

Continue

```
var i = 0
while (i < 10) {
    if (i == 4) {
        i++
        continue
    }
    println(i)
    i++
}
```

Output:

```
0
1
2
3
5
6
7
8
9
```

Labeled Break Statement

```
fun main(args: Array<String>) {
    outerLoop@ for (i in 1..3) {
        innerLoop@ for (j in 1..3) {
            println("i = $i and j = $j")
            if (i == 2){
                break@outerLoop
            }
        }
    }
}
```

Output:

```
i = 1 and j = 1
i = 1 and j = 2
i = 1 and j = 3
i = 2 and j = 1
```

Labeled Continue Statement

```
fun main() {  
    outerLoop@ for (i in 1..3) {  
        for (j in 1..3) {  
            if (i == 2 && j == 2) {  
                println("Skipping i: 2, j: 2")  
                continue@outerLoop  
            }  
            println("i: $i, j: $j")  
        }  
    }  
}
```

Output:

i: 1, j: 1

i: 1, j: 2

i: 1, j: 3

i: 2, j: 1

Skipping i: 2, j: 2

i: 3, j: 1

i: 3, j: 2

i: 3, j: 3

B) Write a program to implement object-oriented concepts in Kotlin.

// Encapsulation: Using private variables with public getter and setter methods
class Person(private var name: String, private var age: Int) {

// Getter for name
 fun getName(): String {
 return name
 }

// Setter for name
 fun setName(newName: String) {
 name = newName
 }

// Getter for age
 fun getAge(): Int {
 return age
 }

// Setter for age
 fun setAge(newAge: Int) {
 age = newAge
 }

// Display details
 fun displayDetails() {
 println("Name: \$name, Age: \$age")
 }
}

// Inheritance: Base class
open class Animal {
 open fun sound() {
 println("This animal makes a sound.")
 }
}

// Derived class
class Dog : Animal() {
 override fun sound() {
 println("\nInheritance Concept")
 println("The dog barks.")
 }
}

// Polymorphism: Overloading

```

class Calculator {
    fun add(a: Int, b: Int): Int {
        return a + b
    }

    fun add(a: Int, b: Int, c: Int): Int {
        return a + b + c
    }
}

// Abstraction: Abstract class with abstract and non-abstract methods
abstract class Shape {
    abstract fun area(): Double
    fun displayType() {
        println("This is a shape.")
    }
}

class Circle(private val radius: Double) : Shape() {
    override fun area(): Double {
        return Math.PI * radius * radius
    }
}

// Interface example
interface Vehicle {
    fun start()
    fun stop()
}

class Car : Vehicle {
    override fun start() {
        println("Car is starting.")
    }

    override fun stop() {
        println("Car is stopping.")
    }
}

fun main() {
    // Encapsulation example
    println("Encapsulation concept")
    val person = Person("John", 25)
}

```

```
person.displayDetails()
person.setName("Jane")
person.setAge(30)
println("Updated Details:")
person.displayDetails()
```

```
// Inheritance and Polymorphism example
val animal: Animal = Dog() // Polymorphism: Animal reference, Dog
object
    animal.sound()
```

```
// Polymorphism example (Overloading)
val calculator = Calculator()
println("\nPolymorphism Concept")
println("Sum of 2 numbers: ${calculator.add(5, 10)}")
println("Sum of 3 numbers: ${calculator.add(5, 10, 15)}")
```

```
// Abstraction example
val circle = Circle(7.0)
println("\nAbstraction Concept")
circle.displayType()
println("Area of the circle: ${circle.area()}")
```

```
// Interface example
val car: Vehicle = Car()
println("\nInterface Concept")
car.start()
car.stop()
}
```

Output:

Encapsulation concept

Name: John, Age: 25

Updated Details:

Name: Jane, Age: 30

Inheritance Concept

The dog barks.

Polymorphism Concept

Sum of 2 numbers: 15

Sum of 3 numbers: 30

Abstraction Concept

This is a shape.

Area of the circle: 153.93804002589985

Interface Concept

Car is starting.

Car is stopping.

Practical 2

Write an android application demonstrating response to event/user interaction for

- a. Checkbox
- b. Radio button
- c. Button
- d. Spinner

Overview: In this practical, we will create an Android app that displays a course form. The user will be able to fill out the form, and upon submission, a summary of their selections will be displayed.

We will cover the usage of checkboxes, radio buttons, spinners, and buttons, and also demonstrate how to use explicit intents in the app.

9:12 9:14

Programming Languages You Know

☐ Java

☐ Kotlin

☐ Python

☐ JavaScript

Interested in Applying for the Course?

☐ Yes ☐ No

Mode of Attending the Course

Online

Submit

Summary

Programming Languages You Know: Python

Interested in Applying: Yes

Mode of Attending the Course: Online

Back to Form

STEP 1: Create a New Android Project

1.1 Open Android Studio:

Launch Android Studio on your computer.

1.2 Create a New Project:

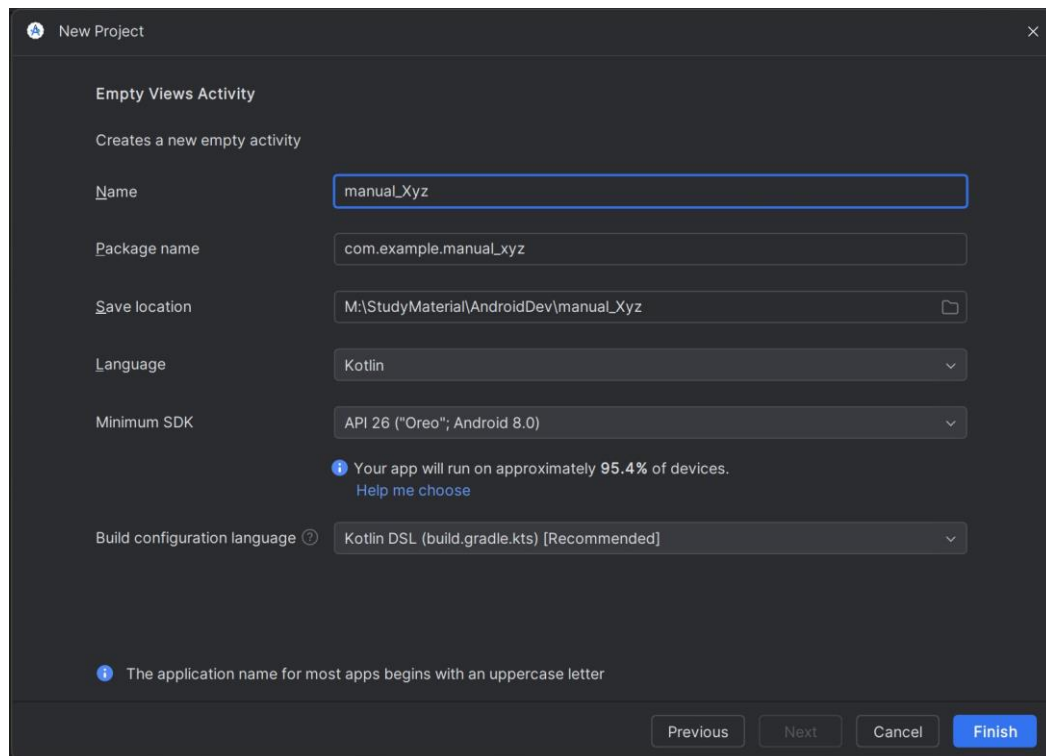
Click on "Start a new Android Studio project".

1.3 Choose a Project Template:

Select "Empty Activity" as the template (avoid choosing "Empty Views Activity").

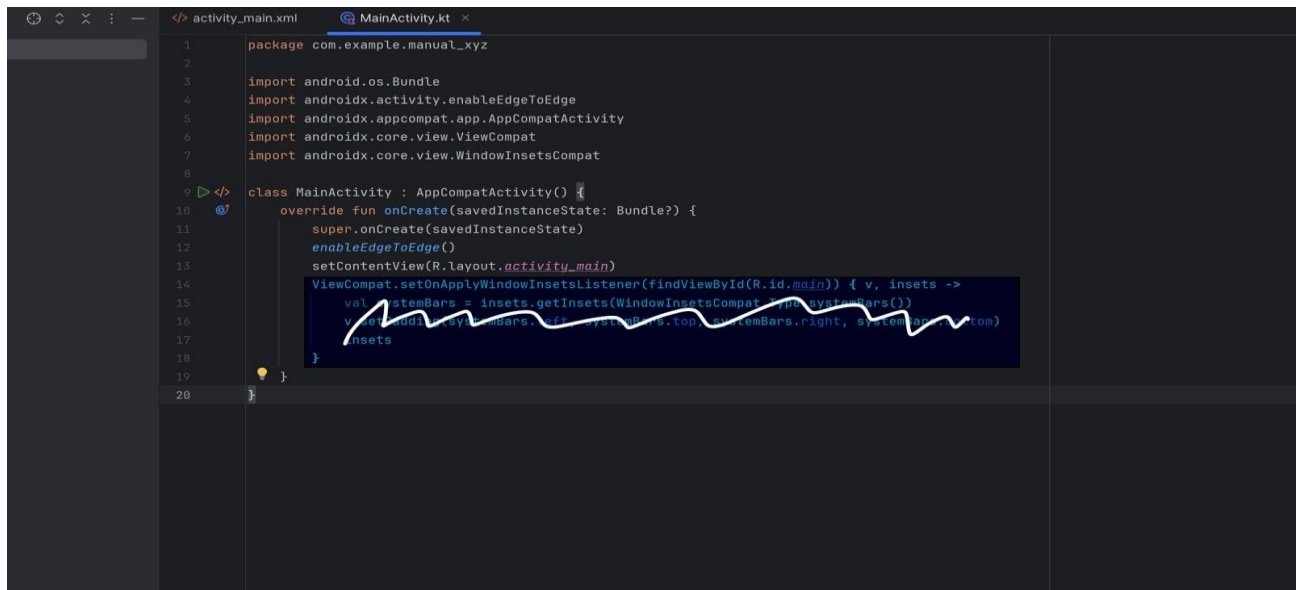
1.4 Configure Your Project:

- Enter a name for your project.
- Set the Save Location where your project will be stored.
- Choose Kotlin as the programming language.
- Set the Minimum API level based on your target devices (API 26 is a good choice).



1.5 Finish: Click "Finish" to create the project.

Note: If the loading bar is displayed, please do not take any action and wait patiently.

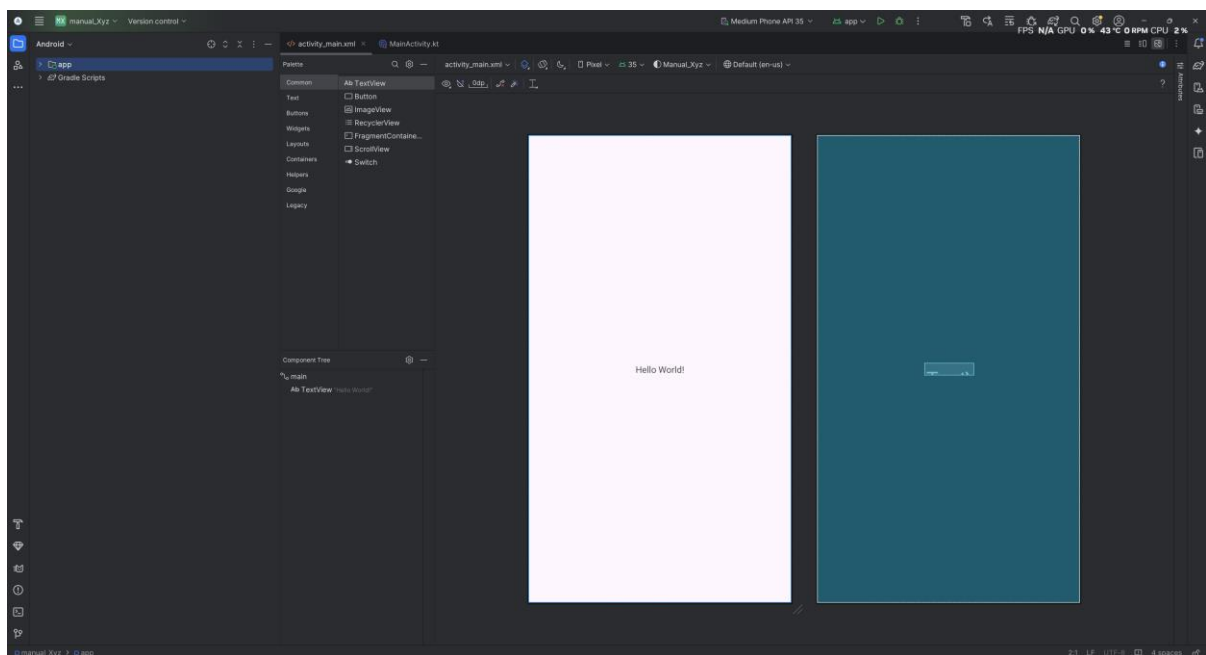


```
1 package com.example.manual_xyz
2
3 import android.os.Bundle
4 import androidx.activity.enableEdgeToEdge
5 import androidx.appcompat.app.AppCompatActivity
6 import androidx.core.view.ViewCompat
7 import androidx.core.view.WindowInsetsCompat
8
9 class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         enableEdgeToEdge()
13         setContentView(R.layout.activity_main)
14         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
15             val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
16             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
17         }
18     }
19 }
20 }
```

Make sure there are two closing curly braces at the end, and verify that the setContentView line is also present.

Step 3: activity_main.xml

1.1 Open the activity_main.xml file. By default, it should look like this.

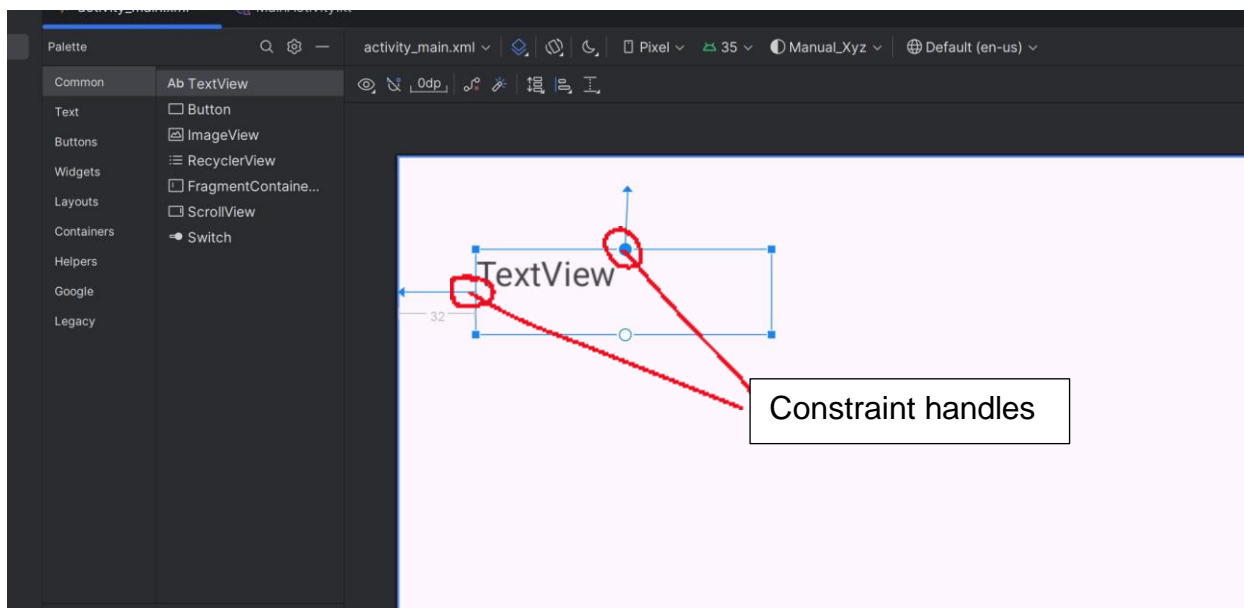


2.2 Remove the existing 'Hello World!' text and start adding the desired components from the Palette window.

- Add 2-3 checkboxes, 2 radio buttons, a spinner, and a submit button.
- Include TextViews between them for better organization.

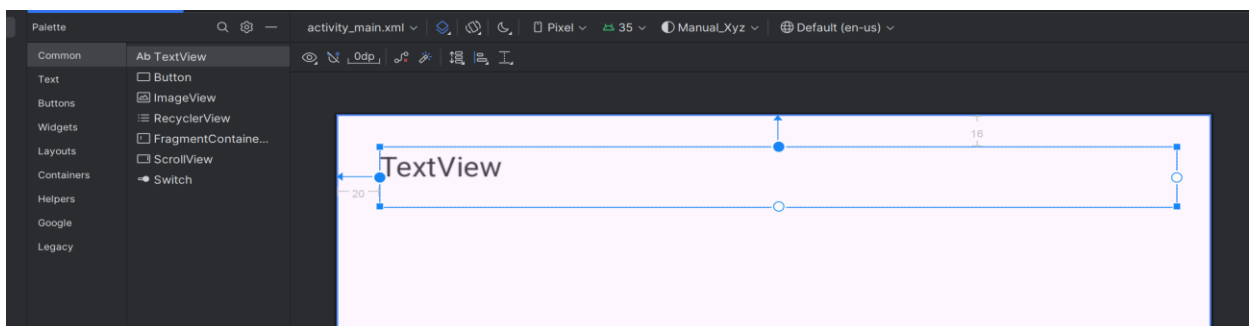
Note: Since we are working with ConstraintLayout, each component must be anchored to at least two points.

Otherwise, it will default to the top-left corner.



2.3 Let's start with checkboxes.

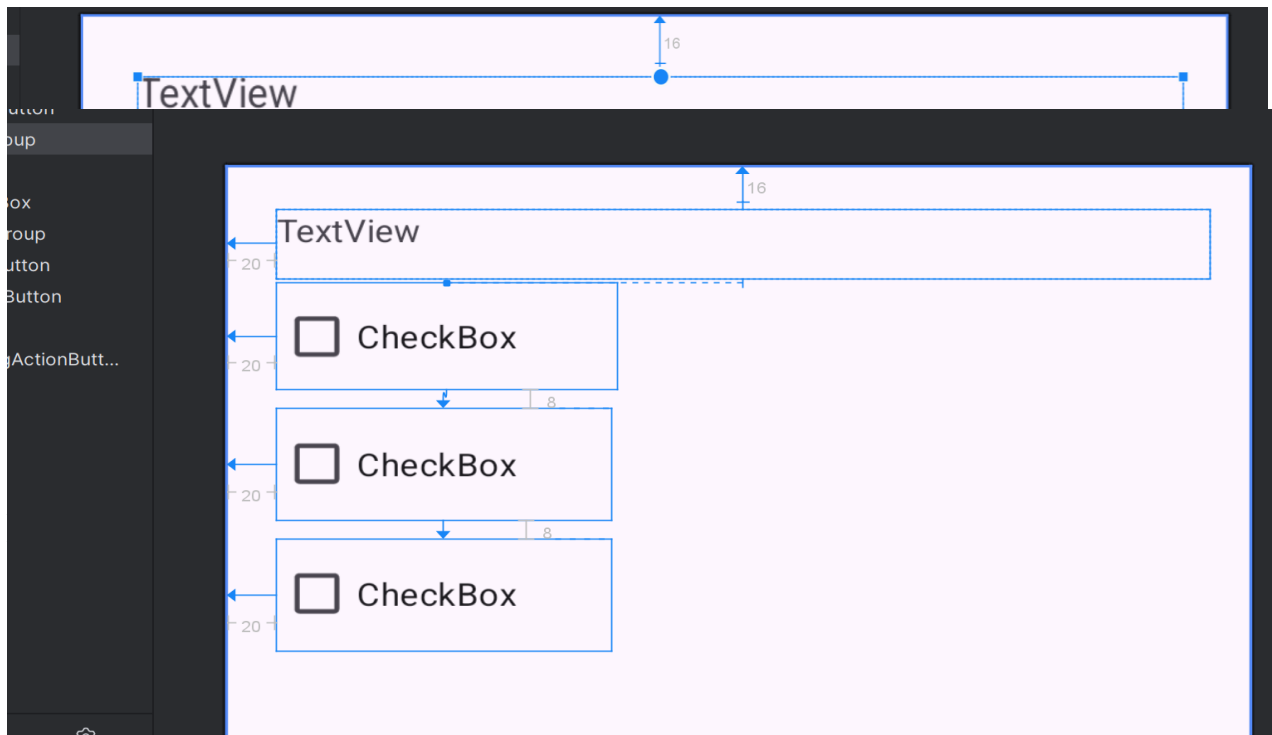
First, add a TextView, resize it, and adjust its constraint handles.



Now, add a checkbox by dragging it from Palette > Buttons. Resize it to make it slightly wider, ensuring it can accommodate longer words.



After adding and positioning the first checkbox, drag the baseline handle of the TextView to the Upper handle of the checkbox. Then, connect the side handles of the checkbox to the edges of the layout.

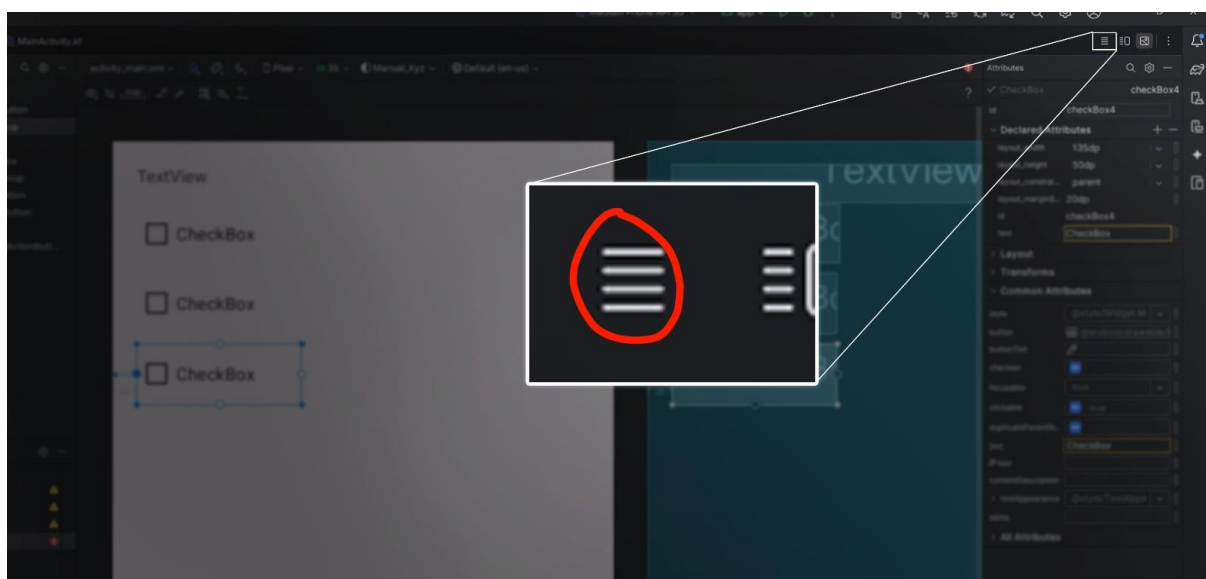


Repeat this process to add the remaining components, including the radio buttons, spinner, and submit button.

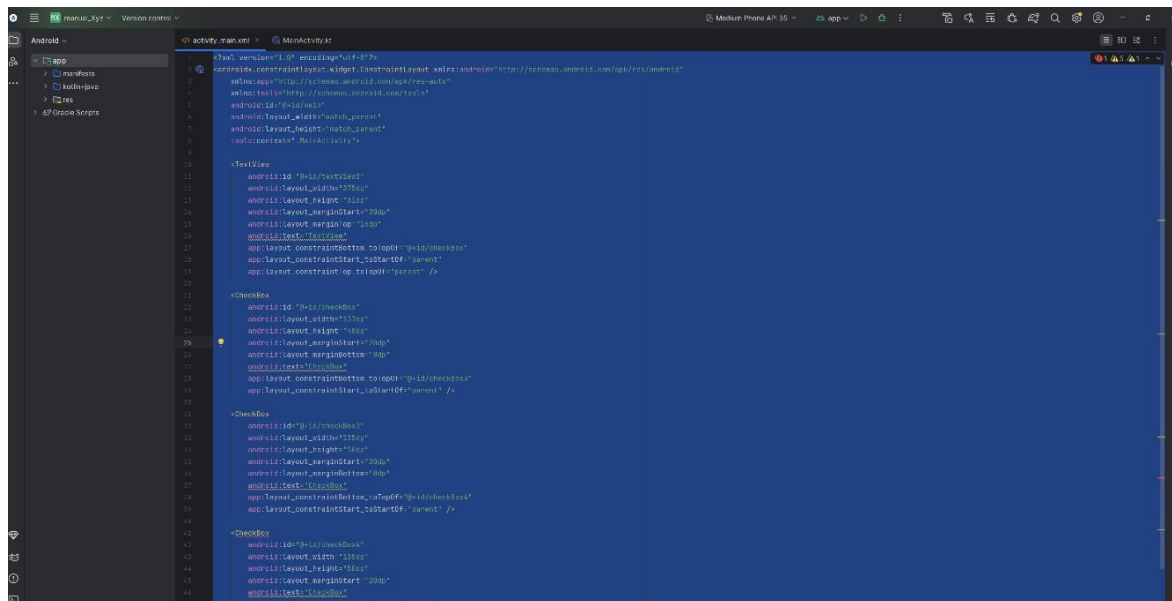
OR

If it's too complicated or not running properly on your computer, try this instead:

In activity_main.xml, open the code editor window by clicking the three lines icon, or use the shortcut **Alt + Shift + Right Arrow**.



Select all the code (**Shift + A**) and delete it.



Now, add the following code to the activity_main.xml file.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
```

```
<!-- CheckBoxes for Programming Languages -->
<TextView android:id="@+id/text_languages"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Programming Languages You Know"
    android:textSize="18sp"
    android:layout_marginBottom="8dp" />
```

```
<CheckBox android:id="@+id/checkbox_java"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Java" />
```

```
<CheckBox android:id="@+id/checkbox_kotlin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Kotlin" />
```

```
<CheckBox android:id="@+id/checkbox_python"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Python" />
```

```
<CheckBox android:id="@+id/checkbox_javascript"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

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

```
<!-- RadioGroup for Interested in Applying -->
```

```
<TextView android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Interested in Applying for the Course?"  
android:textSize="18sp"  
android:layout_marginTop="16dp" />
```

```
<RadioGroup android:id="@+id/radioGroup_interest"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:orientation="horizontal">
```

```
<RadioButton android:id="@+id/radioYes"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Yes" />
```

```
<RadioButton android:id="@+id/radioNo"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="No" />  
</RadioGroup>
```

```
<!-- Spinner for Mode of Attending -->
```

```
<TextView android:id="@+id/text_mode"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Mode of Attending the Course"  
android:textSize="18sp"  
android:layout_marginTop="16dp" />
```

```
<Spinner android:id="@+id/spinnerMode"  
android:layout_width="match_parent"  
android:layout_height="42dp" />
```

```
<!-- Submit Button -->
```

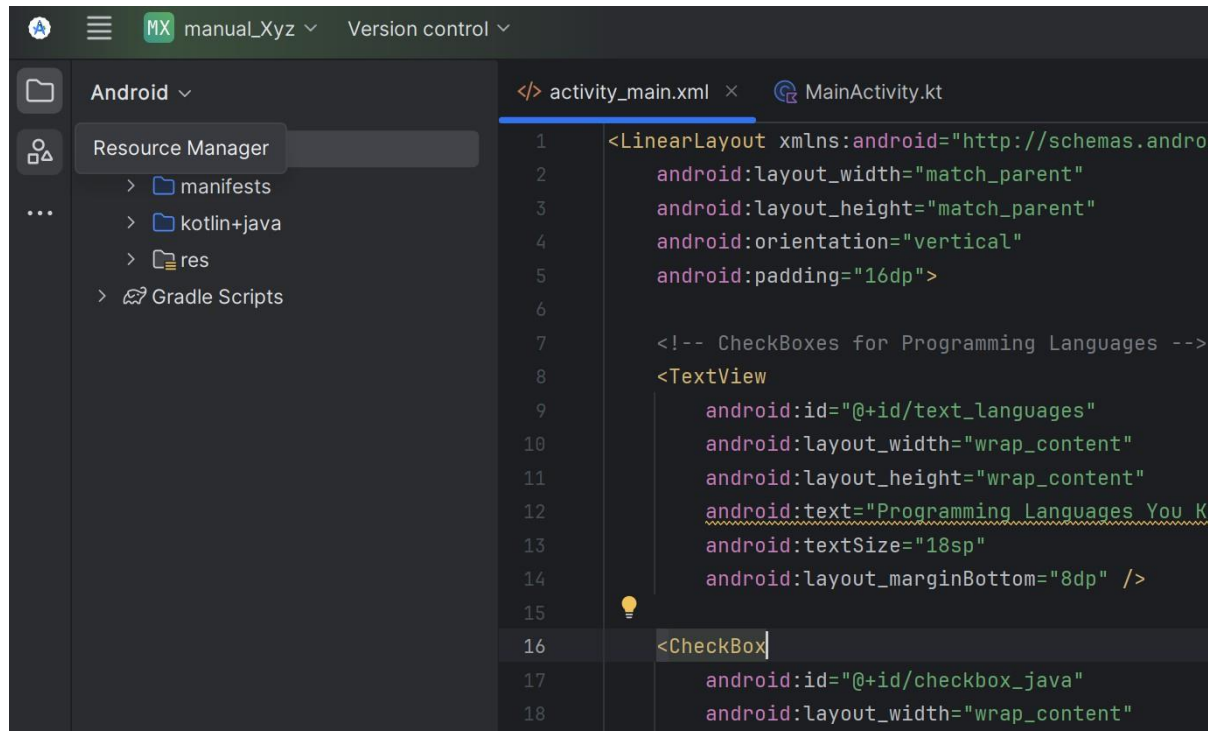
```
<Button android:id="@+id/submitButton"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Submit"  
android:layout_marginTop="16dp" />
```

```
</LinearLayout>
```

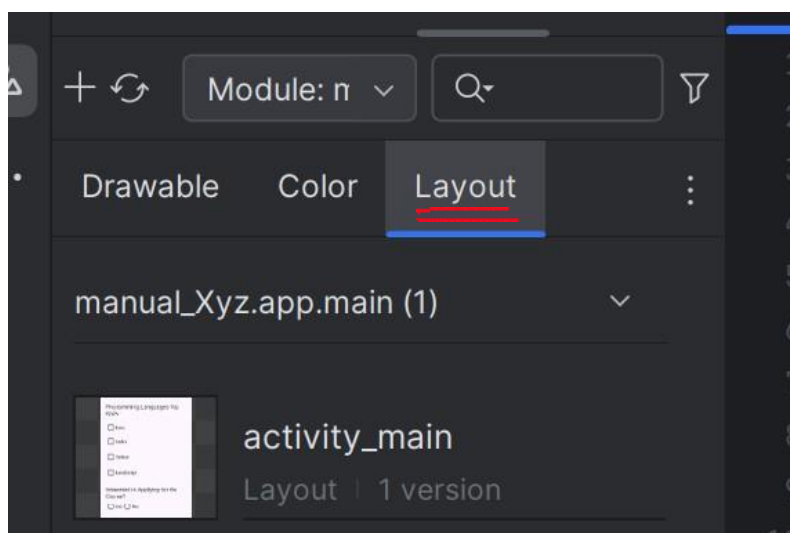
We're done with the activity_main.xml file. If you're using the first method, i.e., ConstraintLayout, update the attributes such as id and text to match the code provided above.

Step 4: activity_summary.xml

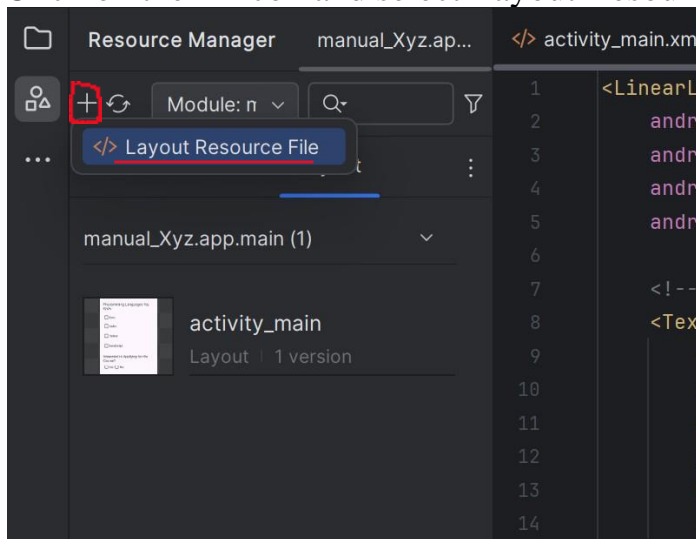
Since we will summarize our selections on the next page, let's create it now. Go to the Resource Manager (shapes icon on the left side).



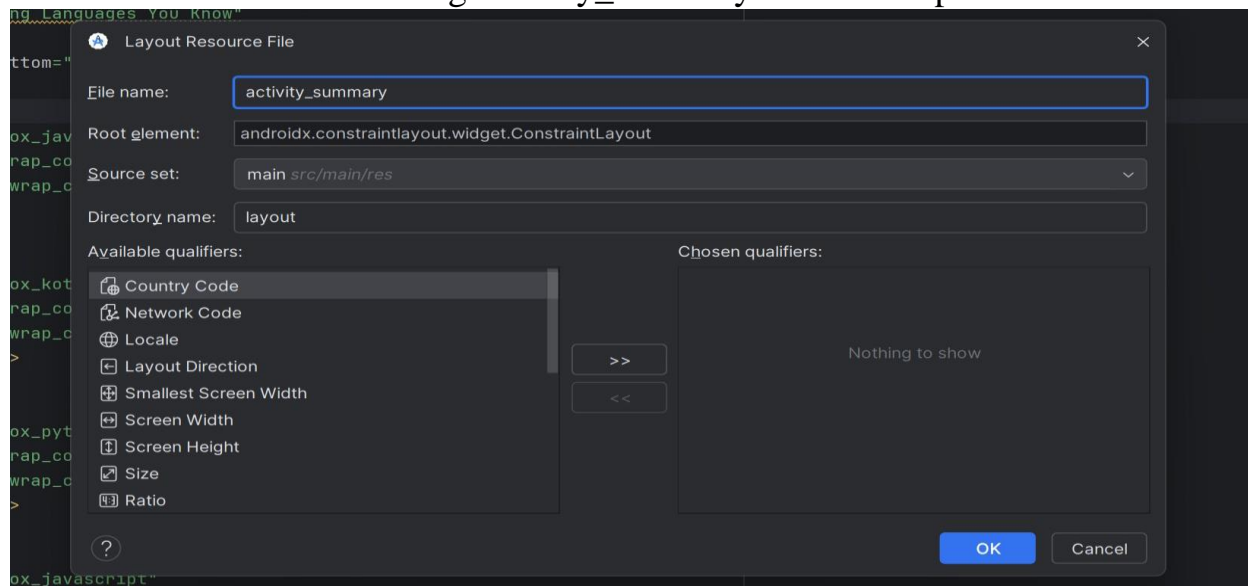
Click on the Layout tab.



Click on the '+' icon and select Layout Resource File.

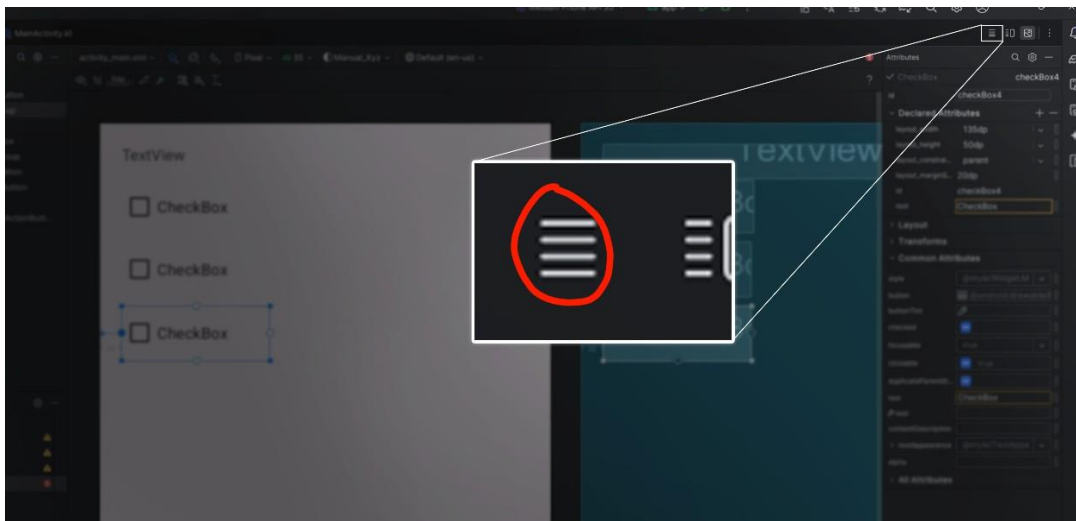


Enter the file name. I'm using 'activity_summary' as an example.



Note: If you choose a different name instead of `activity_summary`, make sure to update it consistently throughout all areas of the code.

In `activity_summary.xml`, go to the code window again.



Select all the code and delete it. Then, write the following code in the file.

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

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
```

```
    android:padding="16dp">
```

```
    <TextView android:id="@+id/text_summary"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="Summary"
```

```
        android:textSize="22sp"
```

```
        android:textStyle="bold" />
```

```
    <TextView android:id="@+id/text_details"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:text=""
```

```
        android:textSize="18sp"
```

```
        android:layout_marginTop="16dp" />
```

```
    <Button
```

```
        android:id="@+id/backButton"
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="Back to Form"
```

```
        android:layout_marginTop="16dp" />
    </LinearLayout>
```

Done with activity_summary.xml file.

Step 5: MainActivity.kt

Navigate to the MainActivity.kt file and paste the following code into it.

```
import android.os.Bundle
import android.widget.Spinner
import android.widget.Button
import android.widget.RadioGroup
import android.widget.CheckBox
import android.widget.ArrayAdapter
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private lateinit var spinner: Spinner
    private lateinit var radioGroup: RadioGroup
    private lateinit var submitButton: Button
    private lateinit var checkBoxJava: CheckBox
    private lateinit var checkBoxKotlin: CheckBox
    private lateinit var checkBoxPython: CheckBox
    private lateinit var checkBoxJavaScript: CheckBox
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Find views by their IDs
        spinner = findViewById(R.id.spinnerMode)
        radioGroup = findViewById(R.id.radioGroup_interest)
        submitButton = findViewById(R.id.submitButton)
        checkBoxJava = findViewById(R.id.checkbox_java)
        checkBoxKotlin = findViewById(R.id.checkbox_kotlin)
        checkBoxPython = findViewById(R.id.checkbox_python)
        checkBoxJavaScript = findViewById(R.id.checkbox_javascript)

        // Create an ArrayAdapter for the Spinner (Mode of Attending)
        val modes = arrayOf("Online", "Offline", "Hybrid")
        val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item,
modes)
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item)
        spinner.adapter = adapter
```

```

// Handle button click to submit the form
submitButton.setOnClickListener {
    val selectedLanguages = getSelectedLanguages()
    val interestedInCourse = getInterestedInCourse()
    val modeOfAttending = spinner.selectedItem.toString()
    // Create an Intent to pass data to the summary page
    val intent = Intent(this, SecondaryActivity::class.java)

    intent.putExtra("languages", selectedLanguages)
    intent.putExtra("interestedInCourse", interestedInCourse)
    intent.putExtra("modeOfAttending", modeOfAttending)
    startActivity(intent)
}
}

// Function to get selected programming languages
private fun getSelectedLanguages(): String {
    val languages = mutableListOf<String>()
    if (checkBoxJava.isChecked) languages.add("Java")
    if (checkBoxKotlin.isChecked) languages.add("Kotlin")
    if (checkBoxPython.isChecked) languages.add("Python")
    if (checkBoxJavaScript.isChecked)
        languages.add("JavaScript")
    return languages.toString(", ")
}

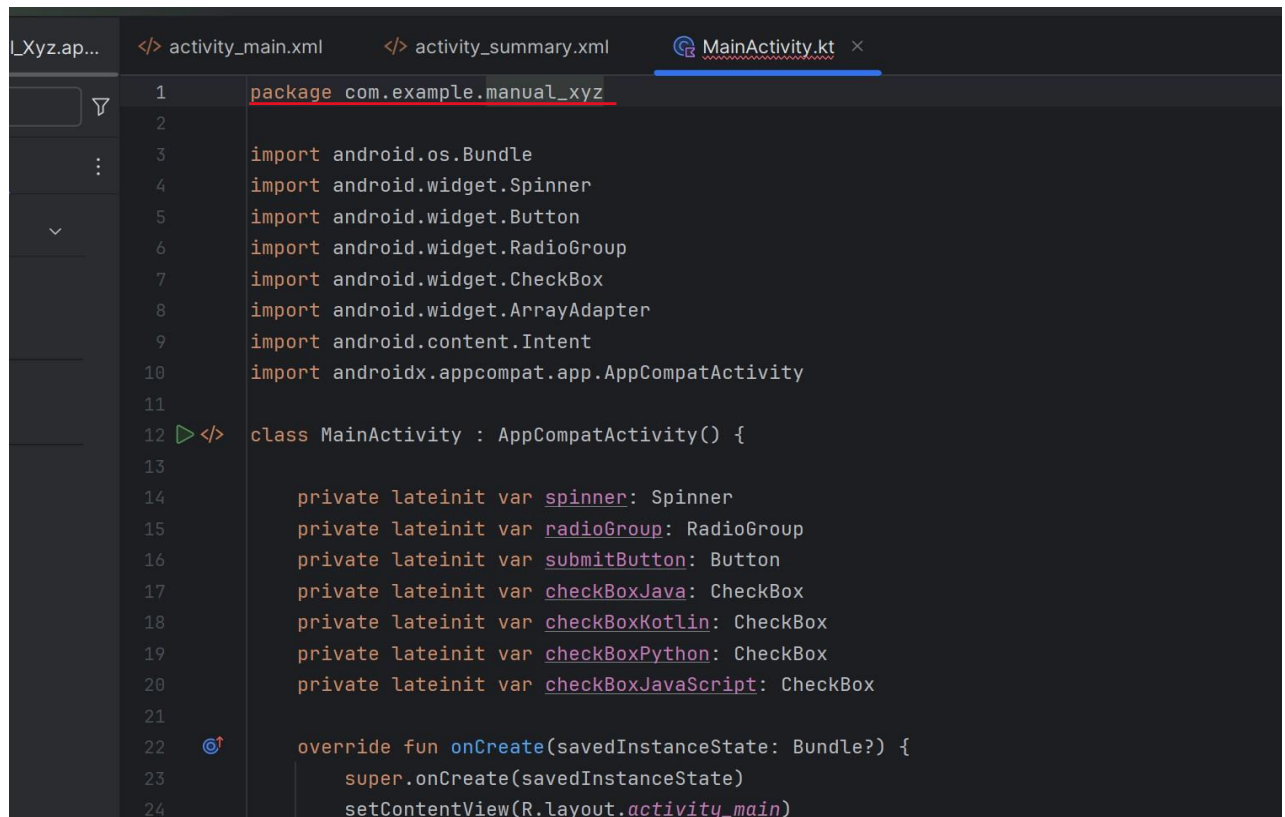
// Function to get if the user is interested in applying

private fun getInterestedInCourse(): String {
    return when (radioGroup.checkedRadioButtonId) {
        R.id.radioYes -> "Yes"
        R.id.radioNo -> "No"
        else -> "Not selected"
    }
}
}
}

```

Note: Make sure to add this line at the top of your code and replace `project_name` with the name you provided when creating the project (in my case, `manual_xyz`).

```
package come.example.project_name
```



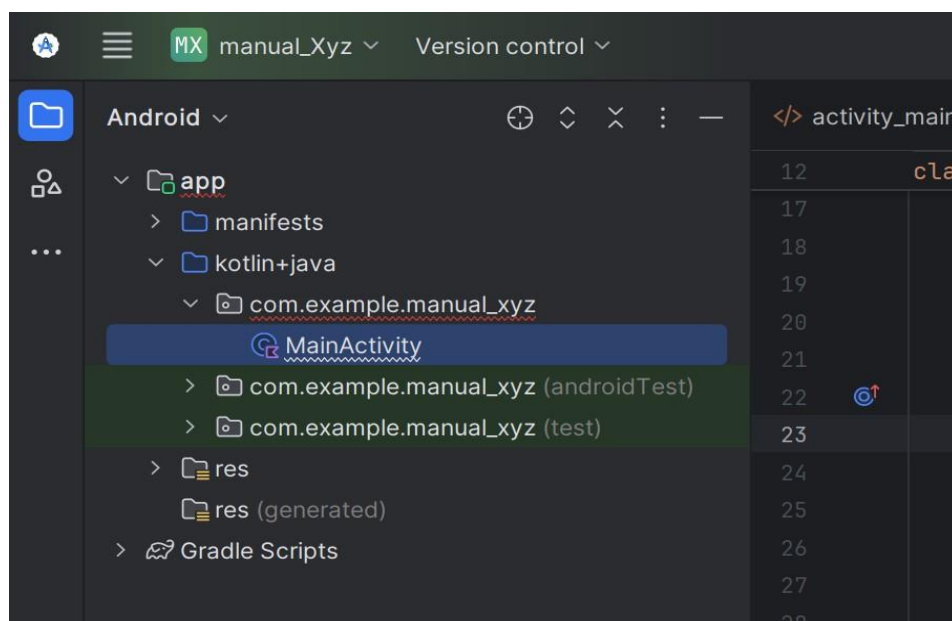
```
1 package com.example.manual_xyz
2
3 import android.os.Bundle
4 import android.widget.Spinner
5 import android.widget.Button
6 import android.widget.RadioGroup
7 import android.widget.CheckBox
8 import android.widget.AdapterView
9 import android.content.Intent
10 import androidx.appcompat.app.AppCompatActivity
11
12 class MainActivity : AppCompatActivity() {
13
14     private lateinit var spinner: Spinner
15     private lateinit var radioButton: RadioGroup
16     private lateinit var submitButton: Button
17     private lateinit var checkBoxJava: CheckBox
18     private lateinit var checkBoxKotlin: CheckBox
19     private lateinit var checkBoxPython: CheckBox
20     private lateinit var checkBoxJavaScript: CheckBox
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_main)
```

Step 6: SecondaryActivity.kt

Let's create a .kt file for the second page.

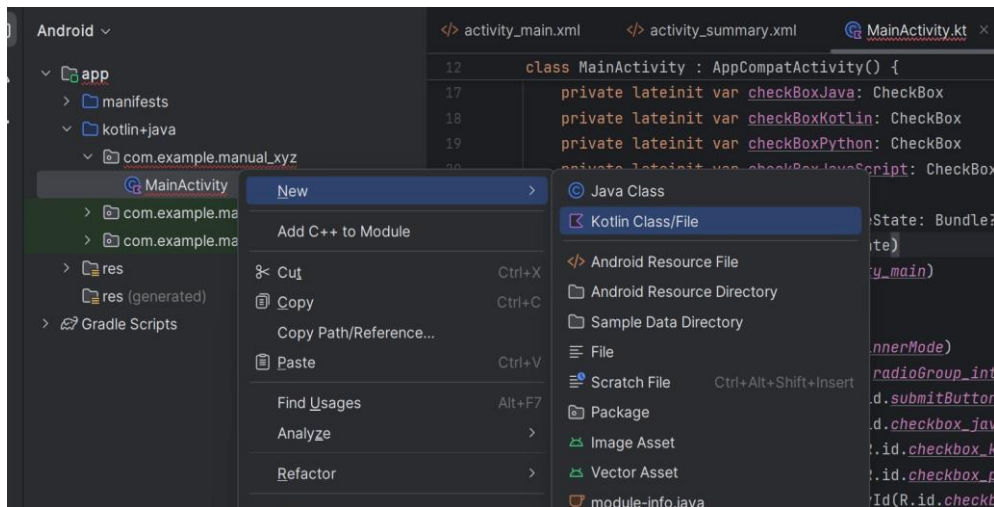
Go to the Project tab (on the sidebar or use the shortcut ALT + 1), then navigate to the location of the MainActivity.kt file:

app -> kotlin + Java -> com.example.project_name

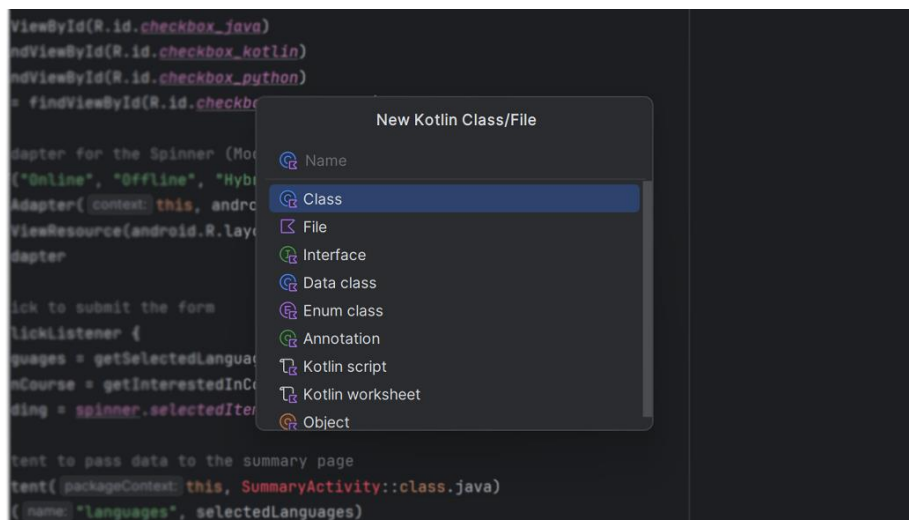


Now, click once on the MainActivity.kt file, right-click to open the context

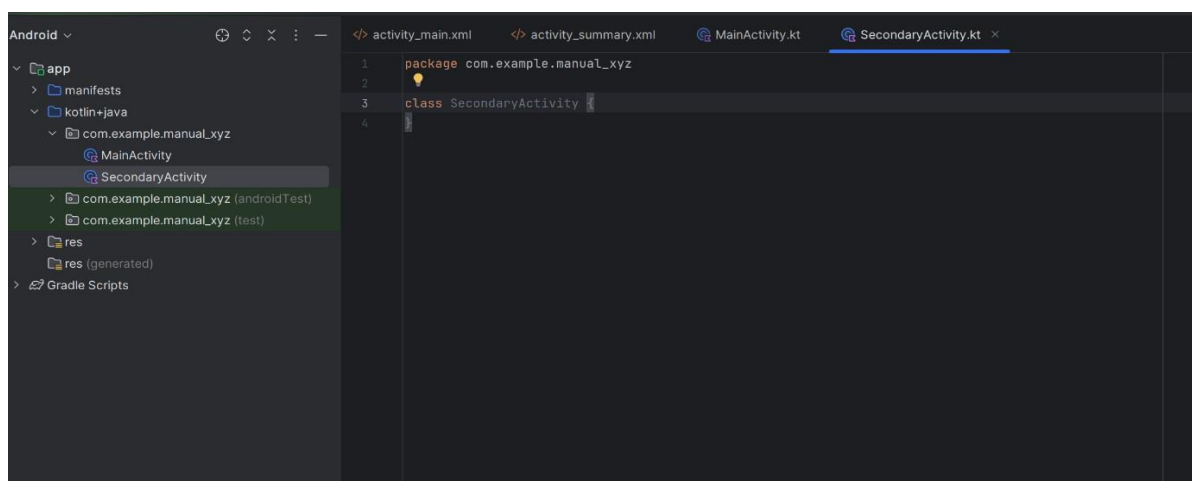
menu, select New, and then choose Kotlin Class/File.



Once this window pops up:



Click on the Name field, enter 'SecondaryActivity', and press Enter. Android Studio will create a .kt file and open it. Please wait for this process to complete.



Once the Secondary Activity window opens, delete all the existing code, except for the first line (package com.example.project_name), and replace it with the following code:

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.TextView
import android.widget.Button

class SecondaryActivity : AppCompatActivity() {

    private lateinit var textDetails: TextView
    private lateinit var backButton: Button

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

        textDetails = findViewById(R.id.text_details) backButton =
            findViewById(R.id.backButton)

        // Get the data from the Intent
        val selectedLanguages = intent.getStringExtra("languages")
        val interestedInCourse = intent.getStringExtra("interestedInCourse")
        val modeOfAttending = intent.getStringExtra("modeOfAttending")

        // Display the data in the TextView
        val summary = ""
        Programming Languages You Know:
        $selectedLanguages
        Interested in Applying: $interestedInCourse
        Mode of Attending the Course: $modeOfAttending """.trimIndent()

        textDetails.text = summary

        // Handle back button to go back to the form
        backButton.setOnClickListener {

            finish() // Finishes this activity and goes back to the previous one
        }
    }
}
```

```

1 package com.example.manual_xyz
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import android.widget.TextView
6 import android.widget.Button
7
8 class SummaryActivity : AppCompatActivity() {
9
10     private lateinit var textDetails: TextView
11     private lateinit var backButton: Button
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         setContentView(R.layout.activity_summary)
16
17         textDetails = findViewById(R.id.text_details)
18         backButton = findViewById(R.id.backButton)
19
20         // Get the data from the Intent
21         val selectedLanguages = intent.getStringExtra("languages")
22         val interestedInCourse = intent.getStringExtra("interestedInCourse")
23         val modeOfAttending = intent.getStringExtra("modeOfAttending")
24
25         // Display the data in the TextView
26         val summary = """
27             Programming Languages You Know: $selectedLanguages
28             Interested in Applying: $interestedInCourse
29             Mode of Attending the Course: $modeOfAttending
30             """
31             .trimIndent()
32
33         textDetails.text = summary
34
35         // Handle back button to go back to the form
36         backButton.setOnClickListener {
37             finish() // Finishes this activity and goes back to the previous one
38         }
39     }
40 }

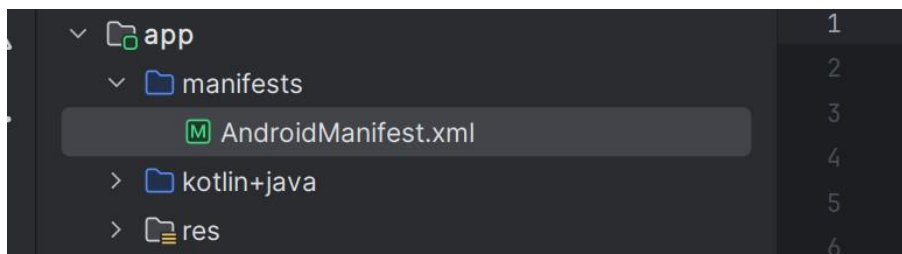
```

Step 7: Final Tweaks

We need to declare our SecondaryActivity in the AndroidManifest.xml file.

In the Project tab, navigate to the AndroidManifest.xml file

app -> manifests -> AndroidManifest.xml



Double-click the AndroidManifest.xml file to open it.

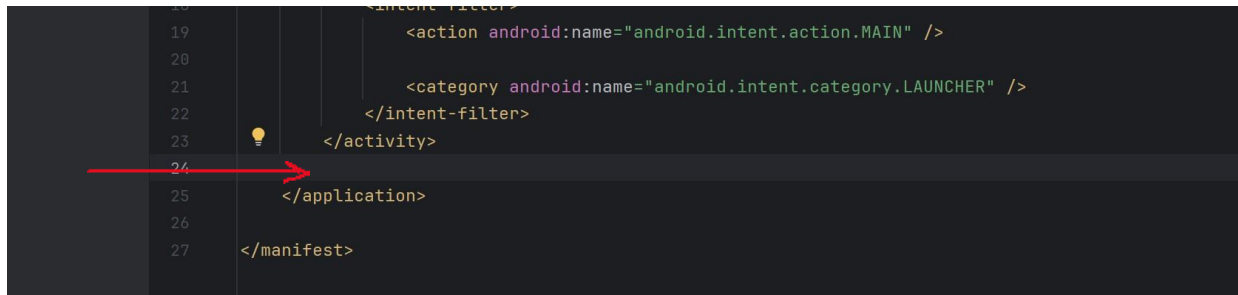
It should appear like this.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <application
6         android:allowBackup="true"
7         android:dataExtractionRules="@xml/data_extraction_rules"
8         android:fullBackupContent="@xml/backup_rules"
9         android:icon="@mipmap/ic_launcher"
10        android:label="@string/app_name"
11        android:roundIcon="@mipmap/ic_launcher_round"
12        android:supportRtl="true"
13        android:theme="@style/Theme.Manual_Xyz"
14        tools:targetApi="31">
15        <activity
16            android:name=".MainActivity"
17            android:exported="true">
18            <intent-filter>
19                <action android:name="android.intent.action.MAIN" />
20
21                <category android:name="android.intent.category.LAUNCHER" />
22            </intent-filter>
23        </activity>
24    </application>
25
26 </manifest>

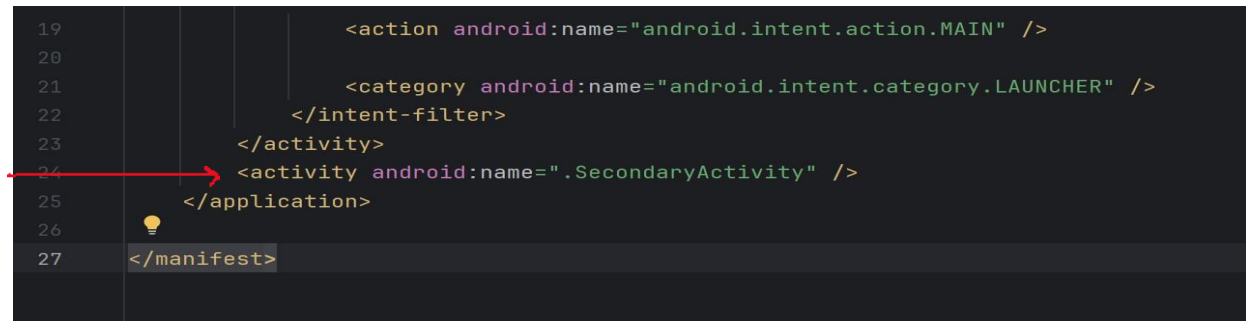
```

Now, let's declare SecondaryActivity.kt. Place your cursor before `</application>` and after `</activity>`.



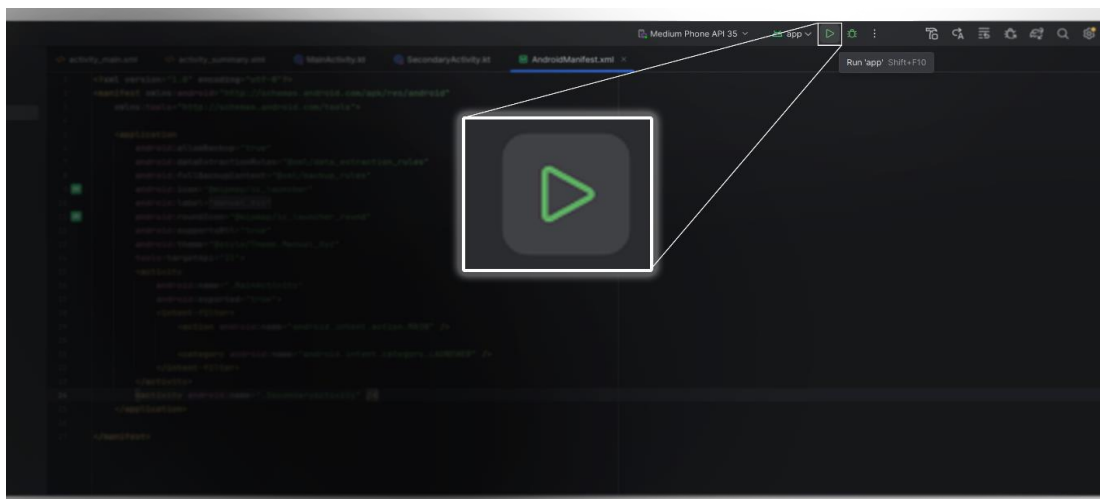
Add the following line there.

`<activity android:name=".SecondaryActivity" />`

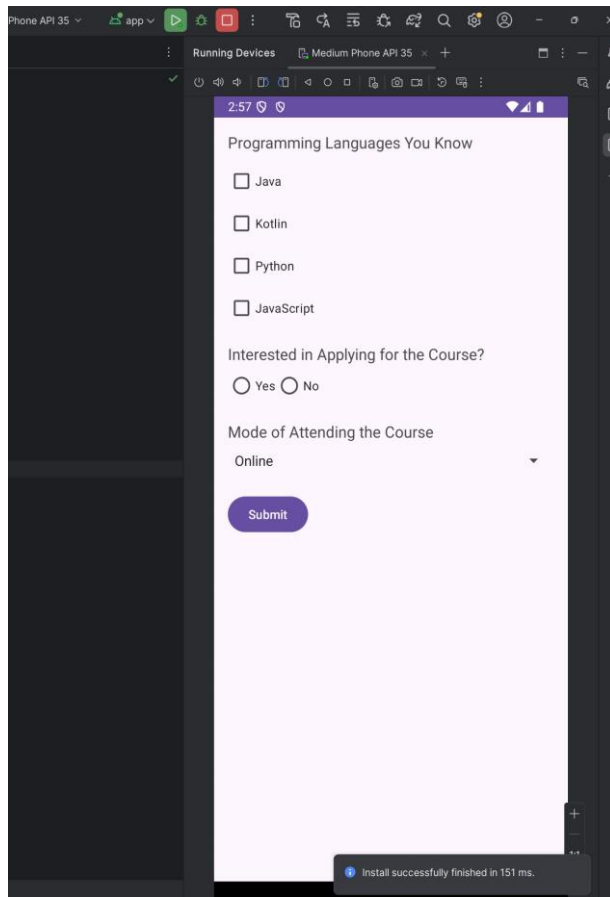


Step 8: Run the application

Click the play button at the top or use the shortcut 'SHIFT + F10'.



wait patiently



Done

Possible errors:

- 1. Cross-check all names to ensure consistency throughout the application.**
- 2. If an SDK error occurs, click the link and refactor accordingly.**

PRACTICAL 3

Create an application to create Image Flipper and Image Gallery. On click on the image display the information about the image.

STEP 1:-Working with the activity_main.xml file

Navigate to the app > res > layout > activity_main.xml and paste the following code to activity_main.xml file. Below is the code for the activity_main.xml file. Comments are added inside the code to understand the code in more detail.

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

    <!--on below line we are adding view pager -->

    <androidx.viewpager.widget.ViewPager

        android:id="@+id/idViewPager"

        android:layout_width="343dp"
        android:layout_height="272dp"
        android:layout_centerInParent="true"
        android:layout_gravity="center"
        android:layout_marginStart="10dp"
```

STEP 2:- Create a layout file for ImageView in View Pager

Navigate to the app > res > layout > Right-click on it > New > Layout Resource file and specify the name as image_slider_item. Paste the following code to the image_slider_item file. Comments are added in the code to understand the code in detail.

Code:-

```
<?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">
    <!--on below line we are creating an image view-->
    <ImageView
        android:id="@+id/idIVImage"
        android:layout_width="200dp"
```

STEP 3:-Create a new kotlin class for the adapter of our ViewPager

Navigate to the app > kotlin+java > your file name/your package name(here our file name is practical_no_4> Right-click on it > New > Java/Kotlin class and name it as ViewPagerAdapter . Delete all the lines in that file except the 1st line and paste the below code to it. Comments are added in the code to understand the code in detail.

```
package com.example.practical_no_4

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import
import android.widget.RelativeLayout
import androidx.viewpager.widget.PagerAdapter
import java.util.*
class ViewPagerAdapter(val context: Context, val imageList: List<Int>)
: PagerAdapter() { // on below line we are creating a method
    // as get count to return the size of the list.
    override fun getCount(): Int {
        return imageList.size
    }
}
```

```

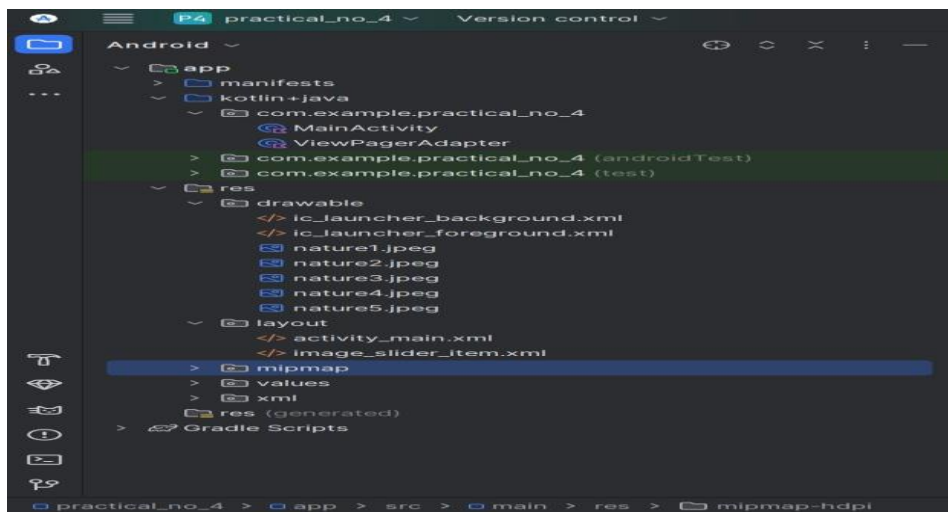
// on below line we are returning the object
override fun isViewFromObject(view: View, `object`: Any): Boolean {
    return view === `object` as RelativeLayout
}
// on below line we are initializing
// our item and inflating our layout file
override fun instantiateItem(container: ViewGroup, position: Int): Any
{
    // on below line we are initializing
    // our layout inflater. val
    mLayoutInflater =
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
    // on below line we are inflating our custom
    // layout file which we have created.
    val itemView: View =
        mLayoutInflater.inflate(R.layout.image_slider_item, container, false)
    // on below line we are initializing
    // our image view with the id. val imageView: ImageView =
    itemView.findViewById<View>(R.id.idIVImage) as ImageView
    // on below line we are setting
    // image resource for image view.
    imageView.setImageResource(imageList.get(position))
    // on the below line we are adding this
    // item view to the container.
    Objects.requireNonNull(container).addView(itemView)
    // on below line we are simply
    // returning our item view. return itemView
}
// on below line we are creating a destroy item method. override fun
destroyItem(container: ViewGroup, position: Int,
`object`: Any) { // on below line we are removing view
    container.removeView(`object` as RelativeLayout)
}
}

```

STEP 4:- Add images to the drawable folder

Select the images which you want to add copy them Navigate to app > res > drawable and right-click on it. Simply paste it and add all the images to the drawable folder.

(for us it looks like this when we add all the images to a drawable folder.)



STEP 5:- Working with the MainActivity.kt file

Go to the MainActivity.kt file and paste the following code. Below is the code for the MainActivity.kt file. Delete all the lines except the first line of this file and paste the code. Comments are added inside the code to understand the code in more detail.

Code:-

```
package com.example.practical_no_4

import android.os.Bundle
import
androidx.appcompat.app.AppCompatActivity
import androidx.viewpager.widget.ViewPager
class MainActivity : AppCompatActivity() {
    // on below line we are creating variable for view pager,
    // viewpager adapter and the image list.
    lateinit var viewPager: ViewPager
    lateinit var viewPagerAdapter:
    ViewPagerAdapter lateinit var imageList:
    List<Int>
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // initializing variables
        // of below line with their id.
        viewPager = findViewById(R.id.idViewPager)
        // on below line we are initializing
        // our image list and adding data to it. imageList
        = ArrayList<Int>()
        imageList = imageList + R.drawable.nature1 //your image
        name imageList = imageList + R.drawable.nature2 //your
```

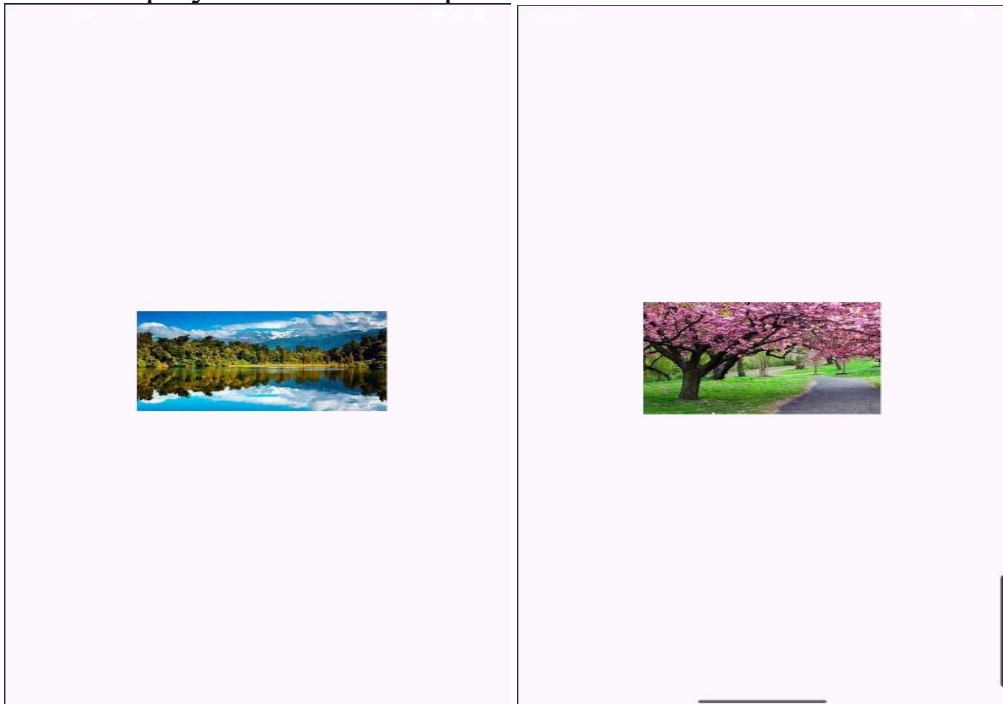
```

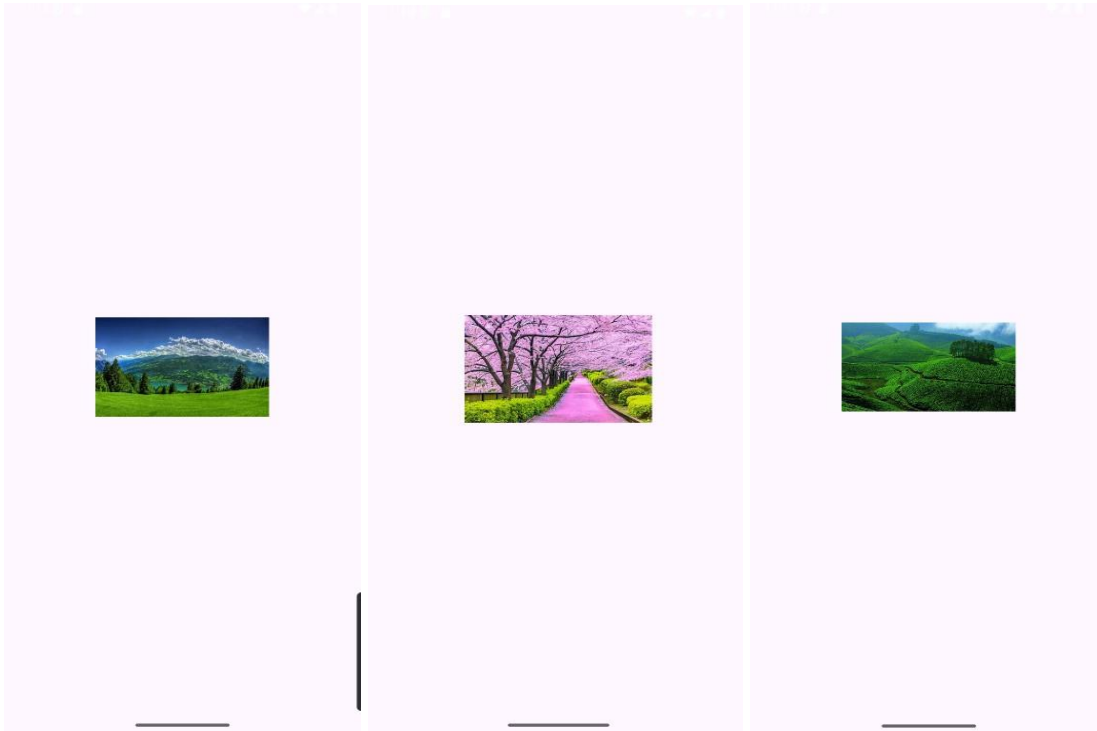
    image name imageUrl = imageUrl + R.drawable.nature3
    //your image name imageUrl = imageUrl +
    R.drawable.nature4 //your image name imageUrl =
    imageUrl + R.drawable.nature5 //your image name
    // on below line we are initializing our view
    // pager adapter and adding image list to it.
    viewPagerAdapter = ViewPagerAdapter(this@MainActivity,
    imageUrl)
    // on below line we are setting
    // adapter to our view pager.
    viewPager.adapter = viewPagerAdapter

```

STEP 6:- Run the application.

Click the play button at the top or use the shortcut 'SHIFT + F10'.





Just click on the screen and your image changes.

Practical 4

Create an Android application to demonstrate implicit and explicit intents

STEP 1:-In this step first we will create a new Android project in Android studio.

STEP2:-Working with the activity_main.xml file.

Now let us design the UI of the activity_main file. In this file, we need to first design two buttons each for explicit intent and implicit intent.

Navigate to app > res > layout > activity_main.xml and paste the following code to activity_main.xml file.

Code:-

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

    <Button
        android:id="@+id/bt_explicit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:layout_gravity="center_horizontal"
        android:text="EXPLICIT INTENT"
        android:onClick="callSecondActivity" />

    <Button
        android:id="@+id/bt_implicit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_gravity="center_horizontal"
        android:text="IMPLICIT INTENT" />

</LinearLayout>
```

</LinearLayout>

STEP 3:- Working with the MainActivity.kt file.

In the **MainActivity.kt** file paste the following code.

```
package com.example.practical_5
```

```
import android.content.Intent
```

```
import android.os.Bundle
```



```

import android.view.View
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import android.net.Uri

class MainActivity : AppCompatActivity() {
    var bt_explicit: Button? = null
    var bt_implicit: Button? = null

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

        bt_explicit = findViewById<Button>(R.id.bt_explicit)
        bt_implicit = findViewById<Button>(R.id.bt_implicit)

        bt_implicit?.setOnClickListener(View.OnClickListener { val implicit_intent
        = Intent(
        Intent.ACTION_VIEW,
        Uri.parse("geo:37.7749,-122.4194")
        )
        startActivity(implicit_intent)
        Toast.makeText(
        this@MainActivity,
        "Clicked Implicit Intent Button",
        Toast.LENGTH_SHORT ).show()
        })
        }

        fun callSecondActivity(view: View?) {
            val explicit_intent = Intent(
            this@MainActivity,
            Explicit_intent::class.java )
            startActivity(explicit_intent)
            Toast.makeText(this@MainActivity, "Clicked Explicit Intent Button",
            Toast.LENGTH_SHORT).show()
            }
        }
    }

```

STEP 4:- Design the UI of the second layout file.

Now design the UI of another activity where the user will navigate after they click on the bt_explicit button.

Navigate to app > res > layout > create a new activity and name it activity_explicit_intent and paste the following code.

Code:-

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

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".Explicit_intent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Explicit Intent "
        android:textSize="20sp"/>

    <Button
        android:id="@+id/bt_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Call First Activity"
        android:onClick="callFirstActivity"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"/>

</LinearLayout>

```

STEP 5:-Create a new kotlin class.

Navigate to the app > kotlin+java > your file name/your package name(here our file name is practical_no_4> Right-click on it > New > Java/Kotlin class and name it as Explicit_intent . Delete all the lines in that file except the 1st line and paste the below code to it.

Code:-

```
package com.example.practical_5
```

```

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class Explicit_intent : AppCompatActivity() {
    var bt_back: Button? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activiy_explicit_intent)
    }
}

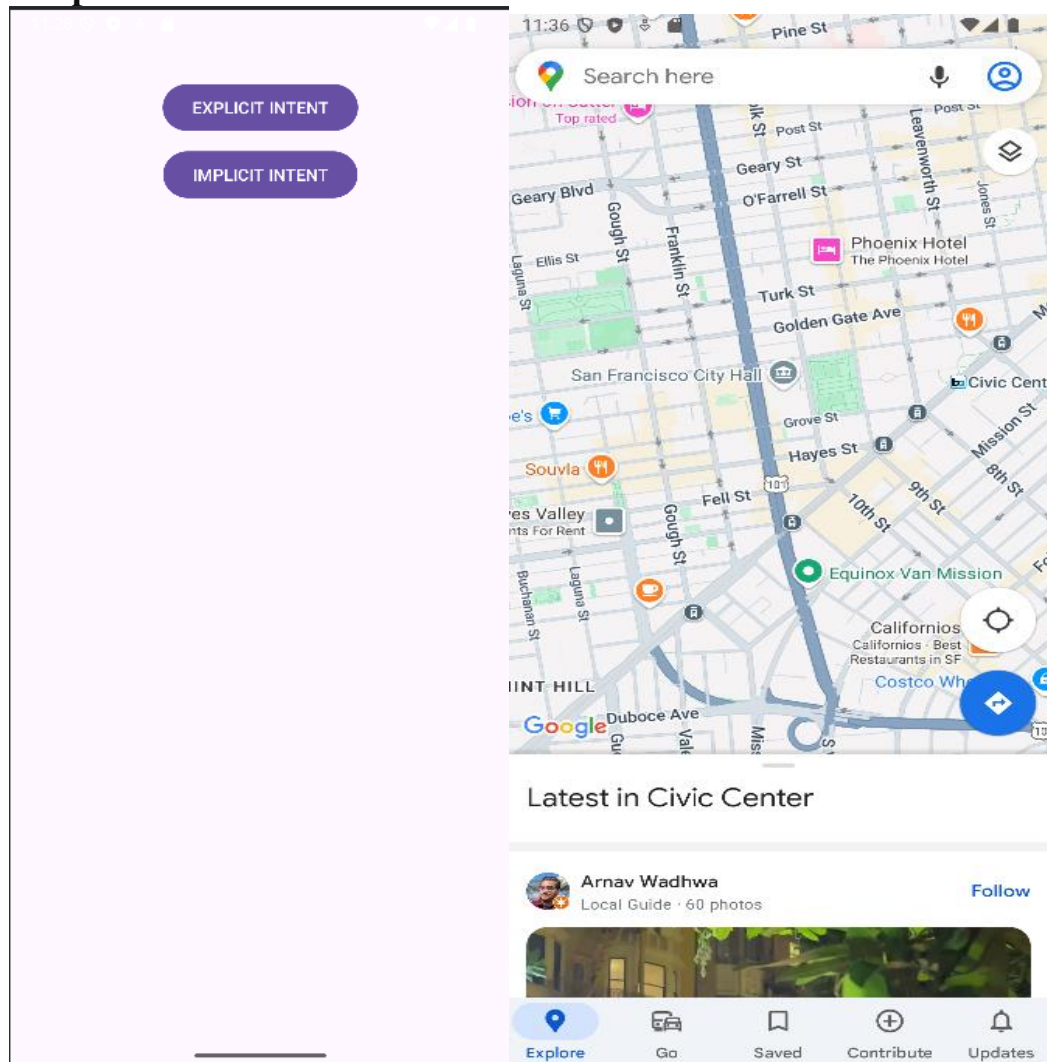
```

```

bt_back = findViewById<Button>(R.id.bt_back)
}
fun callFirstActivity(view: View?) {
    val i = Intent(this, MainActivity::class.java)
    startActivity(i)
    Toast.makeText(this, "We are moved to First Activity",
    Toast.LENGTH_SHORT)
    .show()
}
}

```

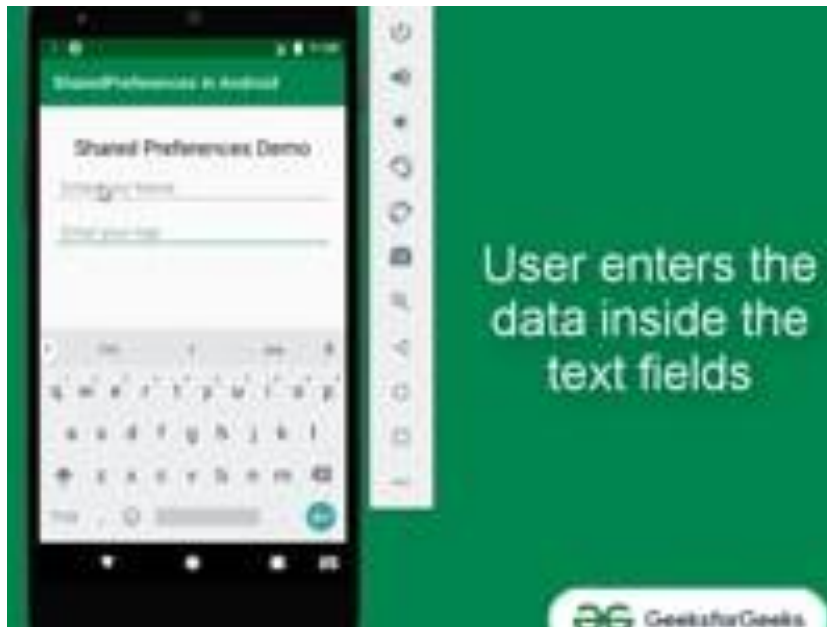
Output:



Practical 5

Create an application to demonstrate shared preferences

We are going to explore the *implementation of SharedPreferences* in Android. Our goal is to create an Android app that stores user-entered data from a textbox and retains it even after the app is removed from the background.



Before proceeding with the steps, please read the following points:

1. This guide **does not cover everything from scratch.**
2. It assumes you have some familiarity with Android Studio's UI.
3. **Avoid blindly copying and pasting**—ensure your project name and required libraries are correctly set.
4. This manual covers only a basic structure—nothing complex or fancy.
5. **Experiment with UI elements**, attributes, and other settings to enhance your understanding.
6. If you encounter any errors, try debugging them yourself first.
7. If an Android SDK error occurs, simply click on the link and refactor it accordingly.

Step 1: Project Setup

1.1 Create a New Project:

Click on "Start a new Android Studio project".

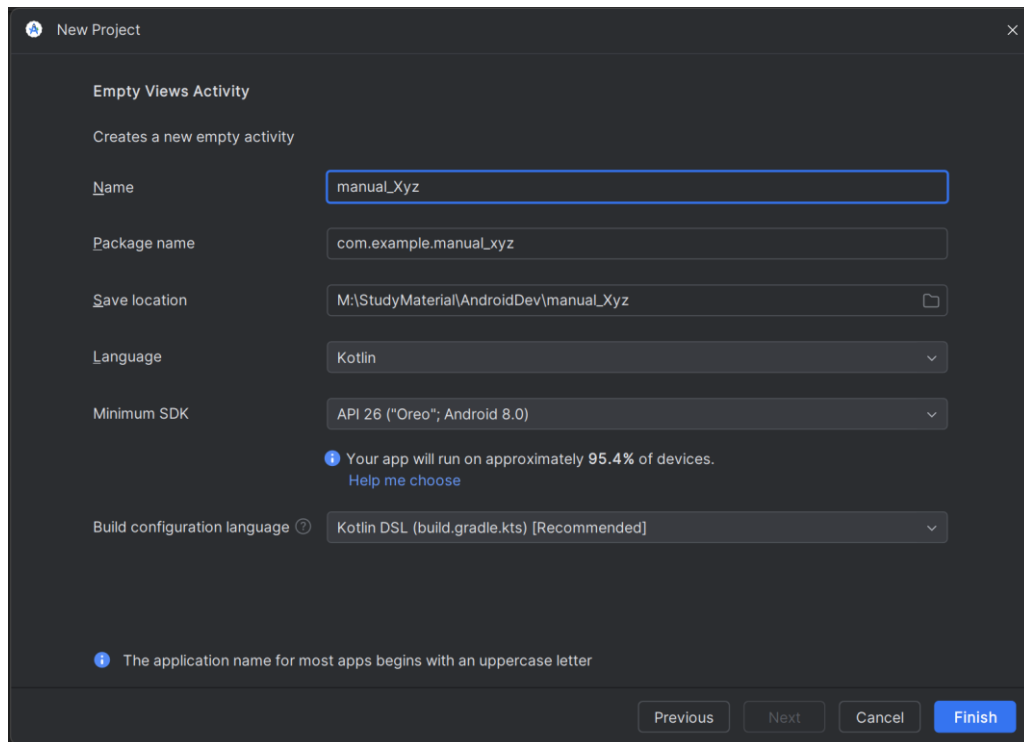
1.2 Choose a Project Template:

Select "Empty Activity" as the template (**avoid choosing "Empty Views Activity"**).

1.3 Configure Your Project:

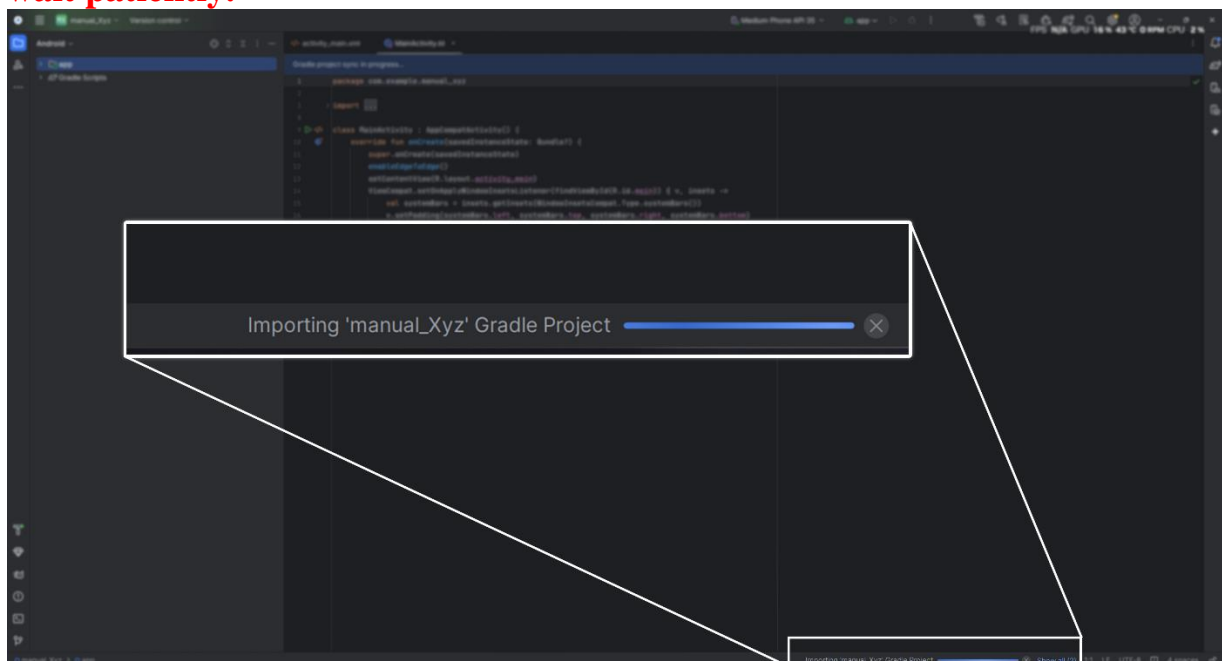
- Enter a name for your project.

- Set the Save Location where your project will be stored.
- **Choose Kotlin** as the programming language.
- Set the Minimum API level based on your target devices (**API 26** is a good choice).



1.4 Finish: Click "Finish" to create the project.

Note: If the loading bar is displayed, please do not take any action and wait patiently.

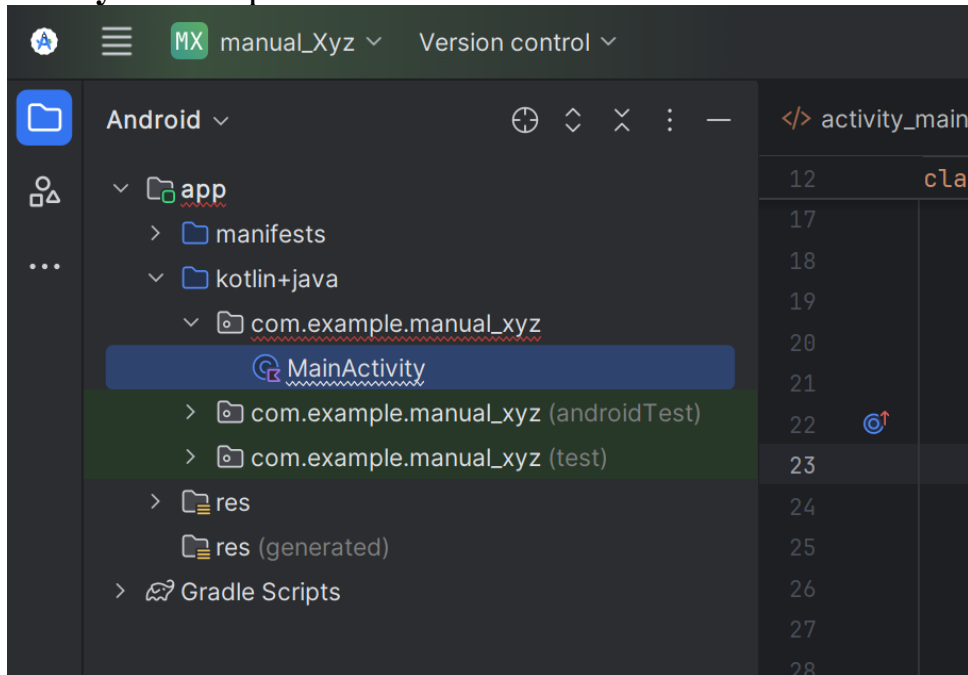


Step 2: MainActivity.kt

2.1: Opening MainActivity.kt

By default, your project should open with MainActivity.kt.

If not, navigate to **app > java + kotlin > com.example.ProjectName > MainActivity.kt** and open the file.



2.2: Writing Kotlin Code in MainActivity.kt

Delete all the existing code in MainActivity.kt, except for the first line:

```
package com.example.ProjectName
```

Note: This line should remain unchanged, reflecting your project's package name.

Now, copy and paste the following code directly below the package name line.

Code:

```
import android.os.Bundle
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private lateinit var name: EditText
    private lateinit var age: EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        name = findViewById(R.id.edit1)
        age = findViewById(R.id.edit2)
    }
}
```

// Fetch the stored data in onResume() Because this is what will be called when the app opens again

```

override fun onResume() {
    super.onResume()
    // Fetching the stored data from the SharedPreferences
    val sh = getSharedPreferences("MySharedPref", MODE_PRIVATE)
    val s1 = sh.getString("name", "")
    val a = sh.getInt("age", 0)

    // Setting the fetched data in the EditTexts
    name.setText(s1)
    age.setText(a.toString())
}

// Store the data in the SharedPreferences in the onPause() method

// When the user closes the application onPause() will be called and data will
be stored
override fun onPause() {
    super.onPause()
    // Creating a shared pref object with a file name "MySharedPref" in
private mode
    val sharedPreferences = getSharedPreferences("MySharedPref",
MODE_PRIVATE)
    val myEdit = sharedPreferences.edit()

    // write all the data entered by the user in SharedPreferences and apply
myEdit.putString("name", name.text.toString())
myEdit.putInt("age", age.text.toString().toInt())
myEdit.apply()
}
}

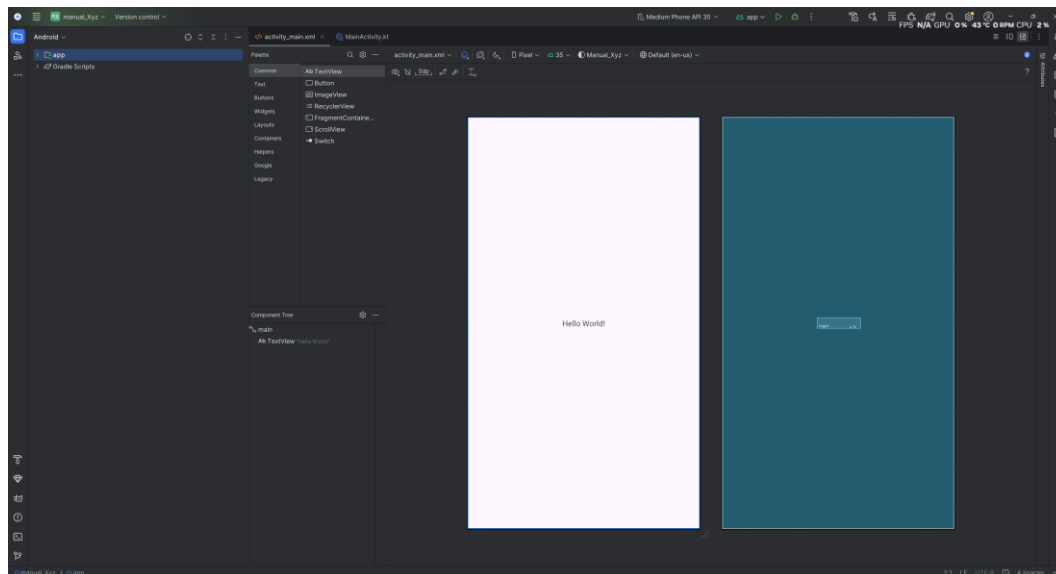
```

Note: Since we haven't created any UI elements in the XML file yet, you may see two errors for edit1 and edit2 IDs. You can ignore them for now.

Step 3: activity_main.xml

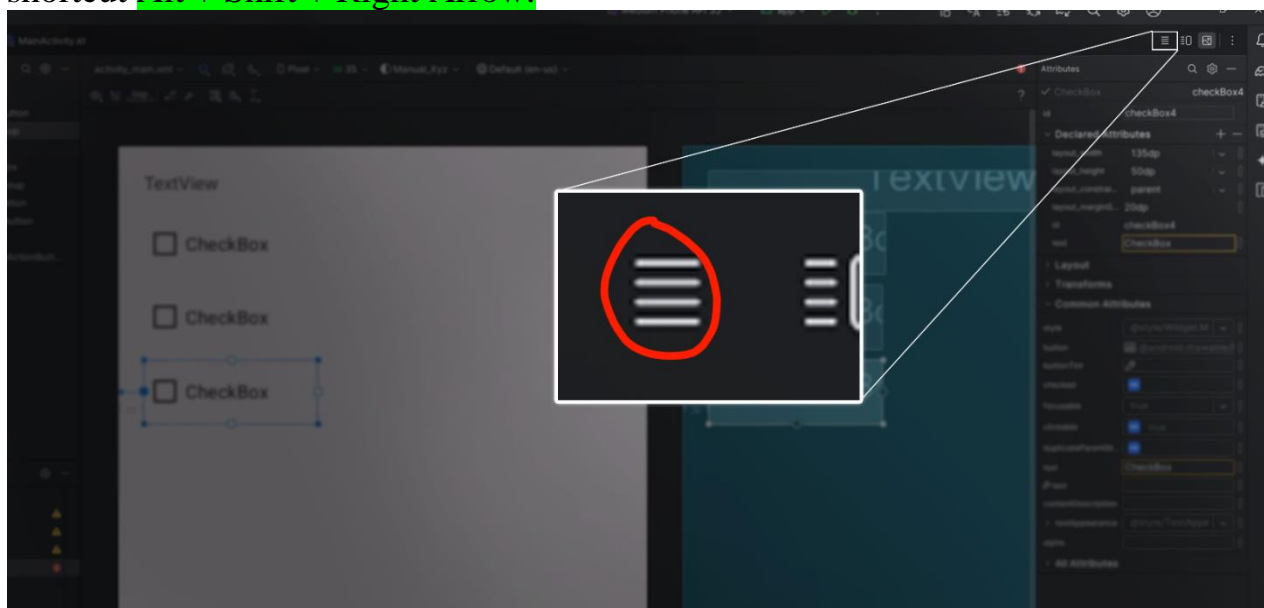
3.1: Opening activity_main.xml file to code

navigate to **app > res > layout > activity_main.xml** and open the file.



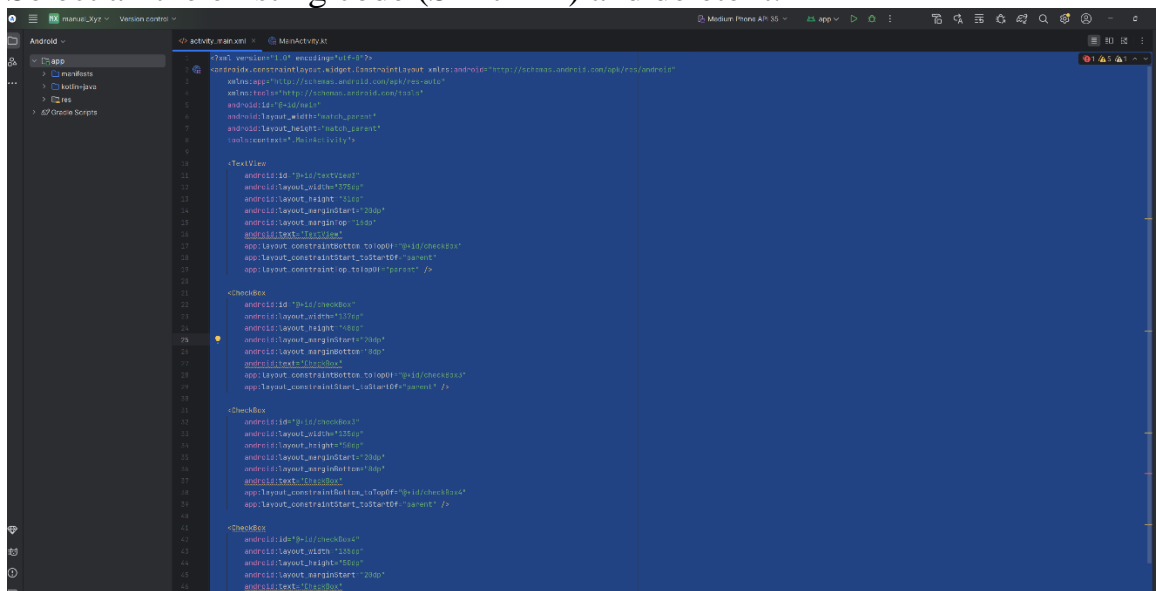
By default, it should look like this.

open the code editor window by clicking the three lines icon, or use the shortcut **Alt + Shift + Right Arrow**.



3.2: Writing Code in activity_main.xml

Select all the existing code (Shift + A) and delete it.



Next, **copy and paste** the following code into the activity_main.xml file.

Code:

```
<?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"
tools:ignore="HardcodedText">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="Shared Preferences Demo"
        android:textColor="@android:color/black"
        android:textSize="24sp" />

    <!--EditText to take the data from the user and save the data in
    SharedPreferences-->
    <EditText
        android:id="@+id/edit1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textview"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="16dp"
        android:hint="Enter your Name"
        android:padding="10dp" />

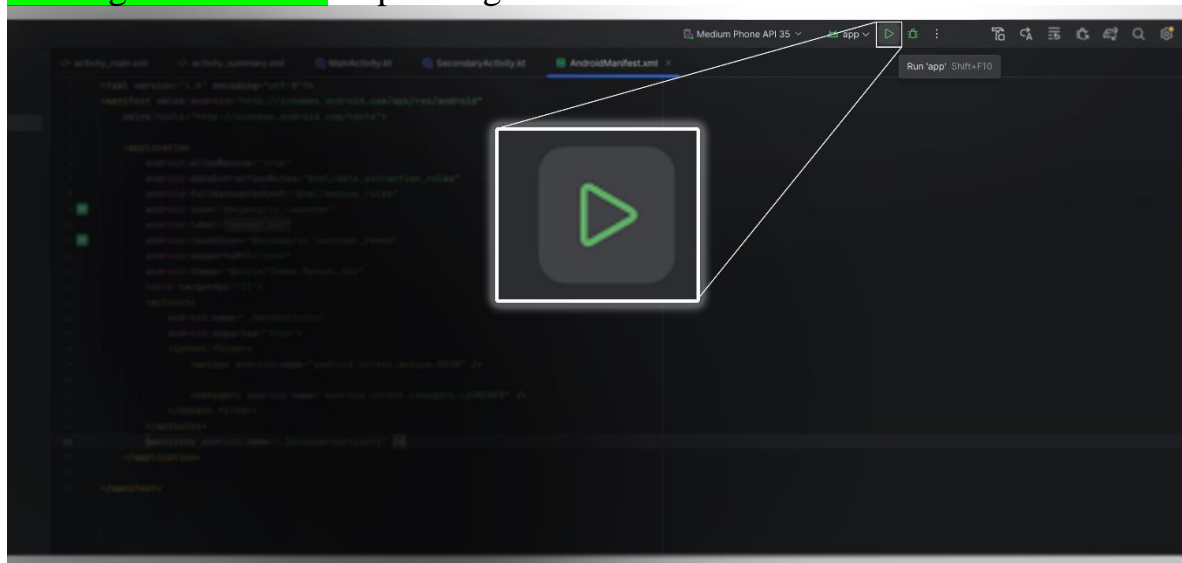
    <!--EditText to take the data from the user and save the data in
    SharedPreferences-->
    <EditText
        android:id="@+id/edit2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edit1"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="16dp"
        android:hint="Enter your Age"
```

```
android:inputType="number"
android:padding="10dp" />
</RelativeLayout>
```

Step 4: Running the Application

4.1: Save and Run

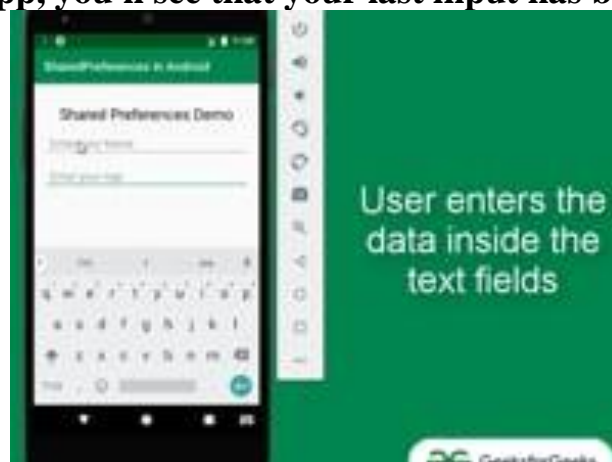
Save both files (.xml and .kt) using Ctrl + S, then run the application by clicking the Run icon or pressing Shift + F10.



Note: Wait patiently while Android Studio builds the app in the virtual machine. If an SDK error occurs, click the provided link, refactor accordingly, and then re-run the application.

DONE

Once completed, enter some text and age in the input fields (EditText), then close the application—even remove it from the background. When you reopen the app, you'll see that your last input has been saved.



Practical 6

Create an Android application to display canvas and allow the user to draw on it.

MainActivity.kt

```
package com.example.canva

import android.annotation.SuppressLint
import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color import
android.graphics.Paint import
android.os.Build
import androidx.appcompat.app.AppCompatActivity import
android.os.Bundle
import android.view.MotionEvent
import android.view.View
import android.widget.ImageView
import androidx.annotation.RequiresApi

class MainActivity : AppCompatActivity(), View.OnTouchListener {

    // Declaring ImageView, Bitmap, Canvas, Paint,
    // Down Coordinates and Up Coordinates
    private lateinit var mImageView: ImageView
    private lateinit var bitmap: Bitmap
    private lateinit var canvas: Canvas
    private lateinit var paint: Paint private
    var downX = 0f
    private var downY = 0f
    private var upX = 0f
    private var upY = 0f

    @RequiresApi(Build.VERSION_CODES.R)
    @SuppressLint("ClickableViewAccessibility")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```

// Initializing the ImageView
mImageView = findViewById(R.id.image_view_1)
// Getting the current window dimensions
val currentDisplay = windowManager.currentWindowMetrics
val dw = currentDisplay.bounds.width()
val dh = currentDisplay.bounds.height()

// Creating a bitmap with fetched dimensions
bitmap = Bitmap.createBitmap(dw, dh, Bitmap.Config.ARGB_8888)

// Storing the canvas on the bitmap
canvas = Canvas(bitmap)

// Initializing Paint to determine
// stroke attributes like color and size
paint = Paint()
paint.color = Color.RED
paint.strokeWidth = 10F

// Setting the bitmap on ImageView
mImageView.setImageBitmap(bitmap)

// Setting onTouchListener on the ImageView
mImageView.setOnTouchListener(this)
}

// When Touch is detected on the ImageView,
// Initial and final coordinates are recorded
// and a line is drawn between them.
// ImageView is updated
@SuppressLint("ClickableViewAccessibility")
override fun onTouch(v: View?, event: MotionEvent?): Boolean {
    when (event!!.action) {
        MotionEvent.ACTION_DOWN ->
        {
            downX = event.x
            downY = event.y
        }

        MotionEvent.ACTION_MOVE -> {
            upX = event.x
            upY = event.y
        }
    }
}

```

```

        canvas.drawLine(downX, downY, upX, upY, paint)
        downX = upX // Update start point for continuous drawing
        downY = upY
        mImageView.invalidate()
    }

    MotionEvent.ACTION_UP -> {
        upX = event.x
        upY = event.y
        canvas.drawLine(downX, downY, upX, upY, paint)
        mImageView.invalidate()
    }
}
return true // Ensure event is consumed
}
}

```

activity_main.xml

```

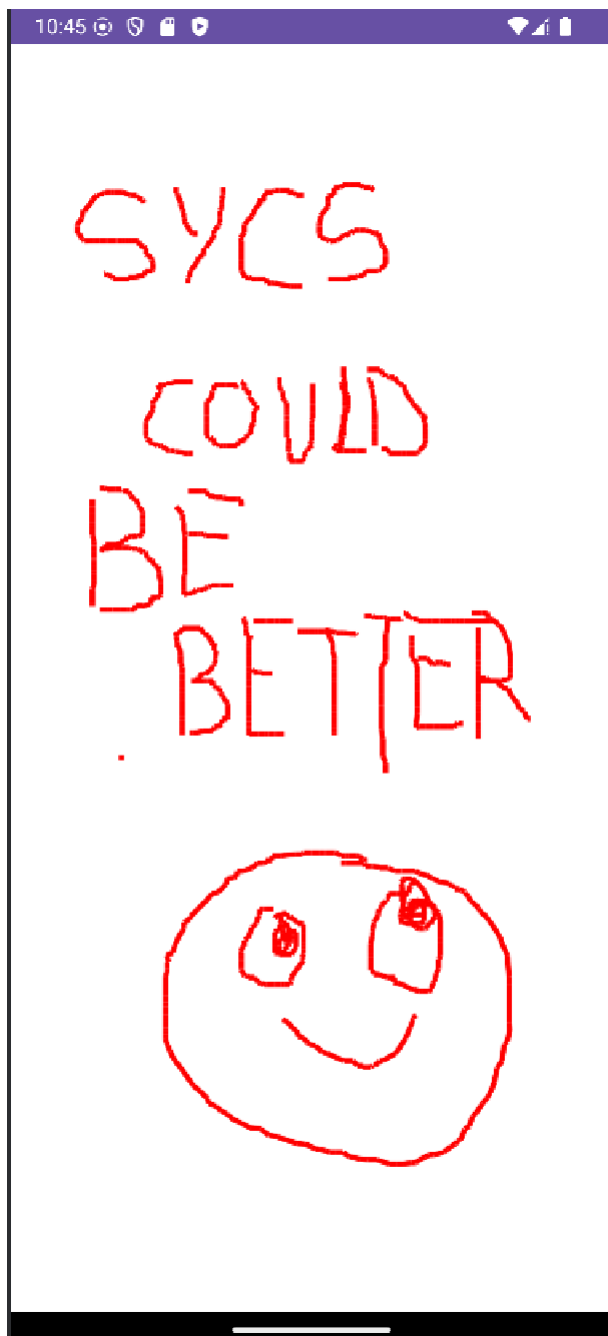
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <ImageView
        android:id="@+id/image_view_1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/white"
        tools:ignore="ContentDescription" />

</RelativeLayout>

```

Output:



Practical 7

Create an Android application to use a camera and capture image/video and display them on the screen.

activity_main.xml file

```
<?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">

    <!-- add Camera Button to open the Camera -->
    <Button
        android:id="@+id/camera_button"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_marginStart="150dp"
        android:text="Camera" />

    <!-- add ImageView to display the captured image -->
    <ImageView
        android:id="@+id/click_image"
        android:layout_width="350dp"
        android:layout_height="450dp"
        android:layout_marginStart="30dp"
        android:layout_marginTop="70dp"
        android:layout_marginBottom="10dp" />
</RelativeLayout>
```

Mainactivity.kt file

```
package com.example.practical_7

import android.content.Intent
import android.graphics.Bitmap
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.Button
import android.widget.ImageView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    // Define the button and imageview type variable
    private lateinit var cameraOpenId: Button
```

```

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

    // By ID we can get each component which id is assigned in XML file get
    Buttons and imageview.
    cameraOpenId = findViewById(R.id.camera_button)
    clickImageId = findViewById(R.id.click_image)

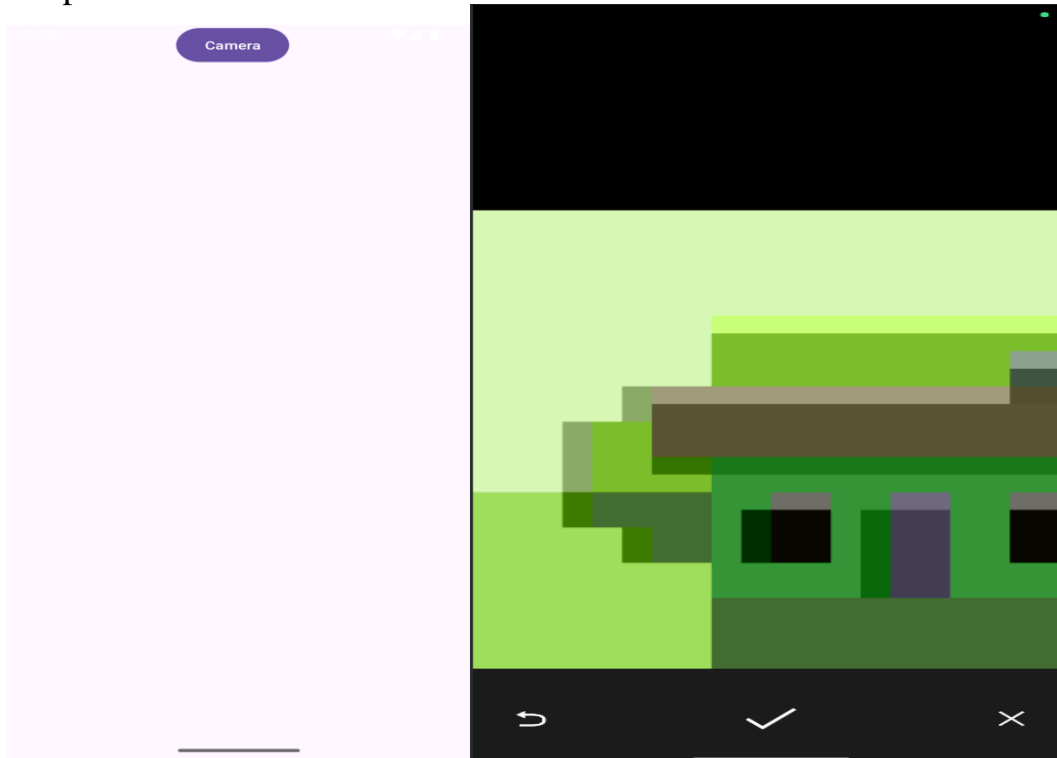
    // Camera_open button is for open the camera and add the
    setOnClickListener in this button
    cameraOpenId.setOnClickListener(View.OnClickListener { v: View? ->
        // Create the camera_intent ACTION_IMAGE_CAPTURE it will open
        the camera for capture the image
        val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        // Start the activity with camera_intent, and request pic id
        startActivityForResult(cameraIntent, pic_id)
    })
}

// This method will help to retrieve the image
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?)
{
    super.onActivityResult(requestCode, resultCode, data)
    // Match the request 'pic id with requestCode
    if (requestCode == pic_id) {
        // BitMap is data structure of image file which store the image in memory
        val photo = data!!.extras!!["data"] as Bitmap?
        // Set the image in imageview for display
        clickImageId.setImageBitmap(photo)
    }
}

companion object {
    // Define the pic id
    private const val pic_id = 123
}
}

```


Output:

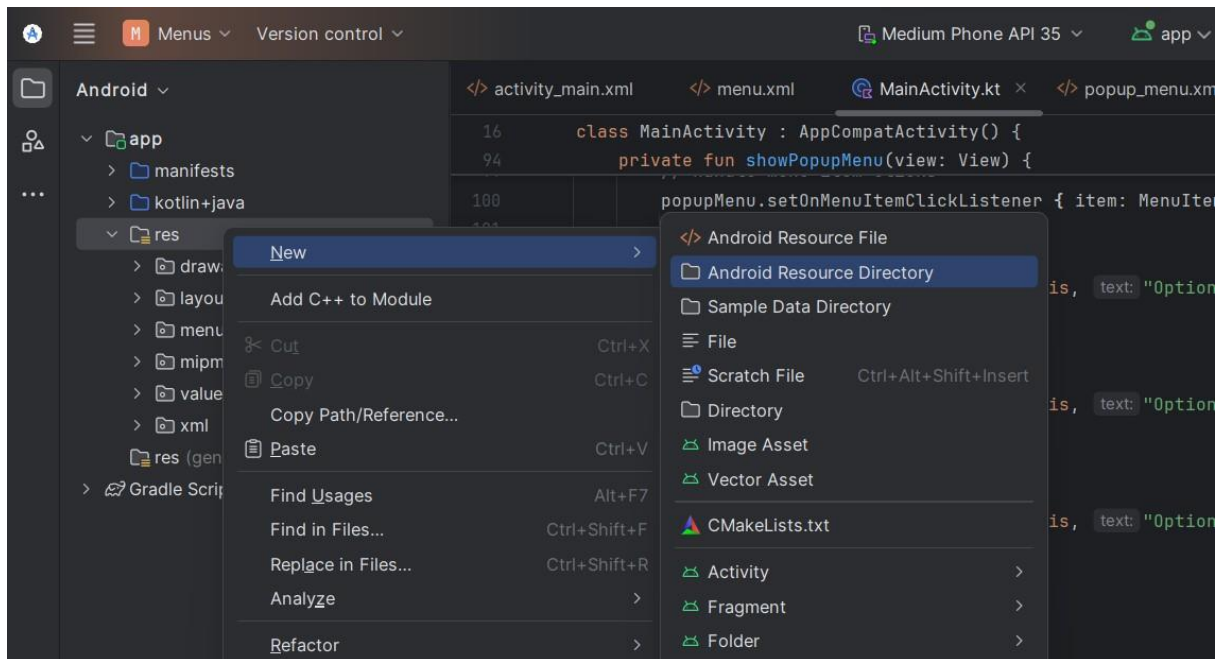


Practical 8

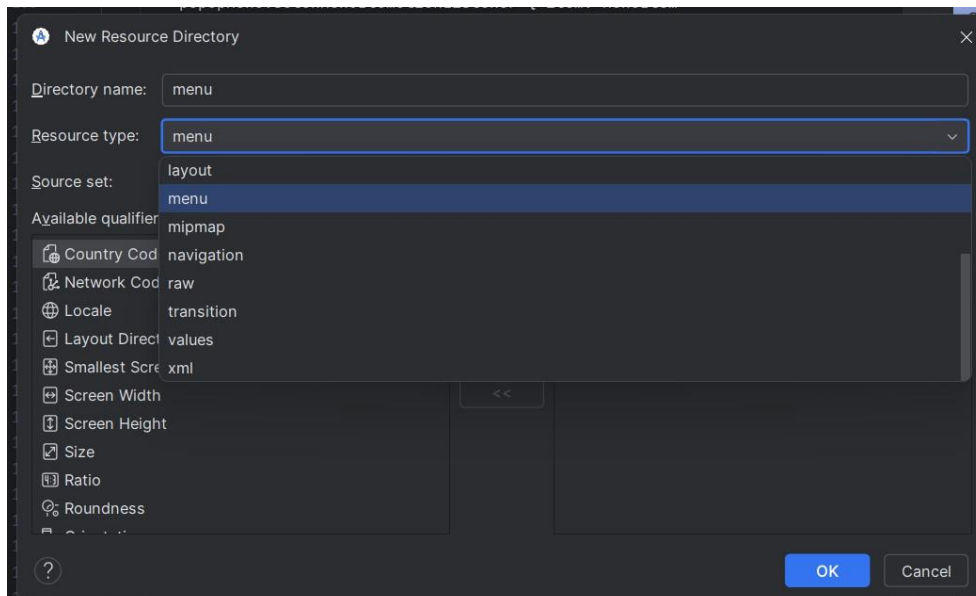
Create an Android application to demonstrate the different types of menus.

- a. Pop-up Menu
- b. Context Menu
- c. Option Menu

Step 1: Create a new directory (Android Resource Directory) in the res directory.

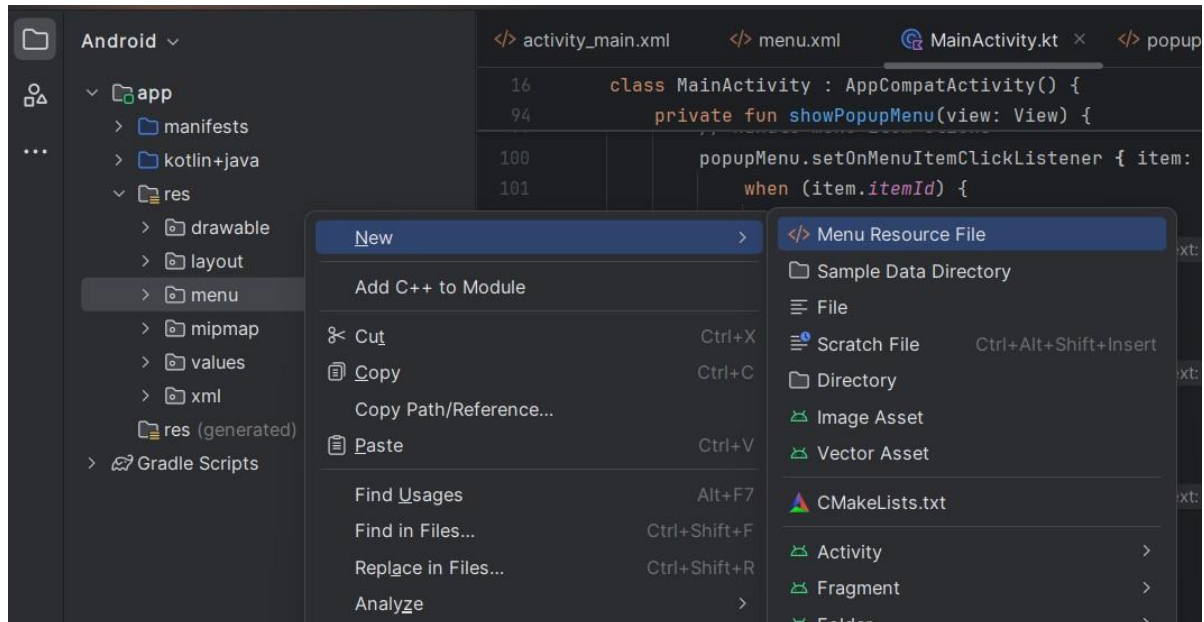


Step 2: Change Resource type to menu and Directory name to menu



Step 3: Create 3 Menu Resource Files

- 1.menu.xml
- 2.popup_menu.xml
- 3.context_menu.xml



Step 4: Enter all the codes in the respective files.

MainActivity.kt

```
package com.example.menus
```

```
import android.annotation.SuppressLint
```

```
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
```

```
import android.view.ContextMenu
```

```
import android.view.Menu
```

```
import android.view.MenuInflater
```

```
import android.view.MenuItem
```

```
import android.view.View
```

```
import android.widget.TextView
```

```
import android.widget.Toast
```

```
import androidx.appcompat.widget.PopupMenu
```

```
import androidx.appcompat.widget.Toolbar
```

```
class MainActivity : AppCompatActivity() {
```

```

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

    val toolbar = findViewById<Toolbar>(R.id.toolbar)
    setSupportActionBar(toolbar) supportActionBar?.title =
    "Options Menu"

    // Find TextView and register it for the context menu val
    textView: TextView = findViewById(R.id.textView)
    registerForContextMenu(textView)

    //Find TextView
    val atextView: TextView = findViewById(R.id.textViews)

    // Set click listener to show pop-up menu
    atextView.setOnClickListener { view ->
        showPopupMenu(view)
    }
}

override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.menu, menu) return
    true
}

override fun onOptionsItemSelected(item: MenuItem): Boolean { return when
(item.itemId) {
    R.id.about -> {
        Toast.makeText(this, "About Selected", Toast.LENGTH_SHORT).show()
        true
    }
    R.id.settings -> {
        Toast.makeText(this, "Settings Selected",
Toast.LENGTH_SHORT).show()
        true
    }
    R.id.exit -> {

```

```

        Toast.makeText(this, "Exit Selected", Toast.LENGTH_SHORT).show() true
    }
    else -> super.onOptionsItemSelected(item)
}
}

// Create the Context Menu
override fun onCreateContextMenu( menu:
    ContextMenu?,
    v: View?,
    menuInfo: ContextMenu.ContextMenuInfo?
) {
    super.onCreateContextMenu(menu, v, menuInfo)
    menuInflater.inflate(R.menu.context_menu, menu)
}

// Handle the Context Menu Item Click
override fun onContextItemSelected(item: MenuItem): Boolean { return when
    (item.itemId) {
        R.id.option1 -> {
            Toast.makeText(this, "Option 1 Selected",
Toast.LENGTH_SHORT).show()
            true
        }
        R.id.option2 -> {
            Toast.makeText(this, "Option 2 Selected",
Toast.LENGTH_SHORT).show()
            true
        }
        R.id.option3 -> {
            Toast.makeText(this, "Option 3 Selected",
Toast.LENGTH_SHORT).show()
            true
        }
        else -> super.onContextItemSelected(item)
    }
}

// Function to show Pop-up Menu

```

```

private fun showPopupMenu(view: View) {
    val popupMenu = PopupMenu(this, view)
    val inflater: MenuInflater = popupMenu.menuInflater
    inflater.inflate(R.menu.popup_menu, popupMenu.menu)

    // Handle menu item clicks
    popupMenu.setOnMenuItemClickListener { item: MenuItem ->
        when (item.itemId) { R.id.option1 ->
            {
                Toast.makeText(this, "Option 1 Selected",
Toast.LENGTH_SHORT).show()
                true
            }
            R.id.option2 -> {
                Toast.makeText(this, "Option 2 Selected",
Toast.LENGTH_SHORT).show()
                true
            }
            R.id.option3 -> {
                Toast.makeText(this, "Option 3 Selected",
Toast.LENGTH_SHORT).show()
                true
            }
            else -> false
        }
    }
    popupMenu.show()
}
}

```

activity_main.xml

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

<androidx.appcompat.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
android:background="@color/material_dynamic_primary70"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
tools:ignore="MissingConstraints" />

<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@android:color/holo_blue_light"
android:gravity="center"
android:padding="16dp"
android:text="Long Press Me for Context Menu"
android:textColor="@android:color/white"
android:textSize="18sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.495"
app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.33" />
```

```
<TextView
    android:id="@+id/textViews"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@android:color/holo_blue_light"
    android:gravity="center"
    android:padding="16dp" android:text="Tap
    Me for Pop-up Menu"
    android:textColor="@android:color/white"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.642" />
```

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

context_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/option1"
        android:title="Option 1" />
    <item
        android:id="@+id/option2"
        android:title="Option 2" />
    <item
        android:id="@+id/option3"
        android:title="Option 3" />
</menu>
```


popup_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/option1"
        android:title="Option 1" />
    <item
        android:id="@+id/option2"
        android:title="Option 2" />
    <item
        android:id="@+id/option3"
        android:title="Option 3" />
</menu>
```

menu.xml

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

    <item
        android:id="@+id/overflowMenu"
        android:icon="@drawable/ic_3_dots"
        android:title="Options"
        app:showAsAction="always">
        <menu>
            <item
                android:id="@+id/settings"
                android:icon="@drawable/ic_settings"
                android:title="SETTINGS"
                app:showAsAction="never" />
            <item
                android:id="@+id/about"
                android:icon="@drawable/ic_about"
                android:title="ABOUT"
                app:showAsAction="never" />
            <item
```

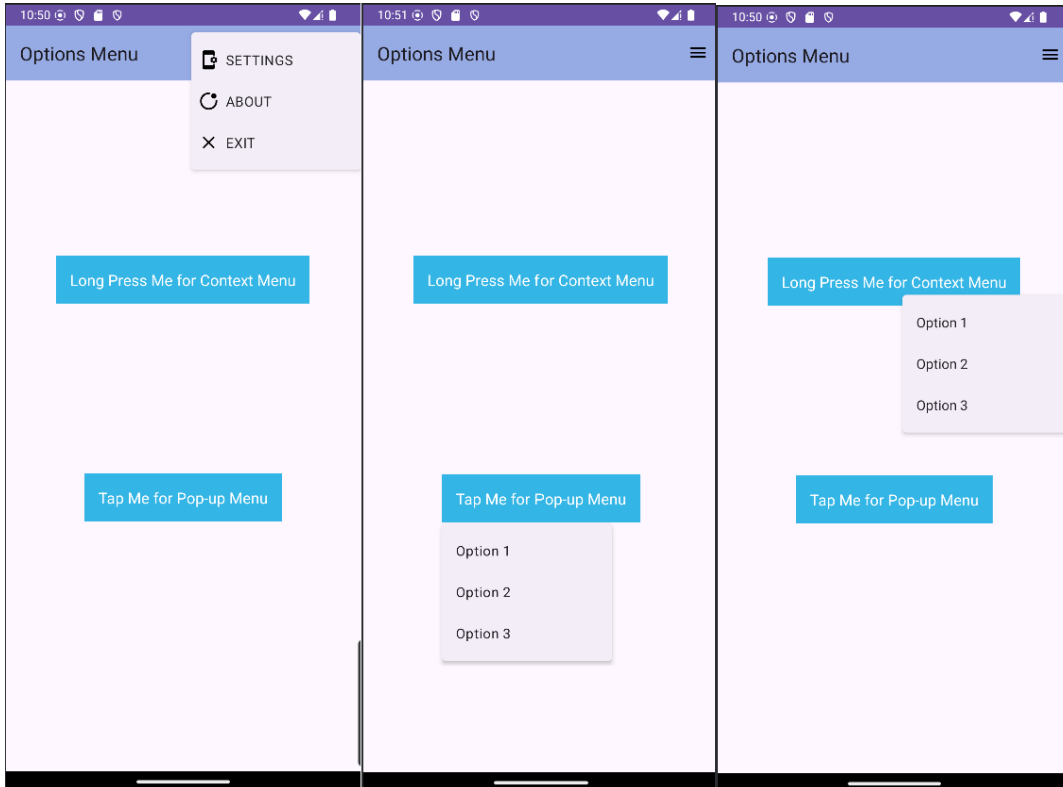
```
android:id="@+id/exit"  
android:icon="@drawable/ic_exit"  
android:title="EXIT"  
app:showAsAction="never" />
```

```
</menu>
```

```
</item>
```

```
</menu>
```

Output:

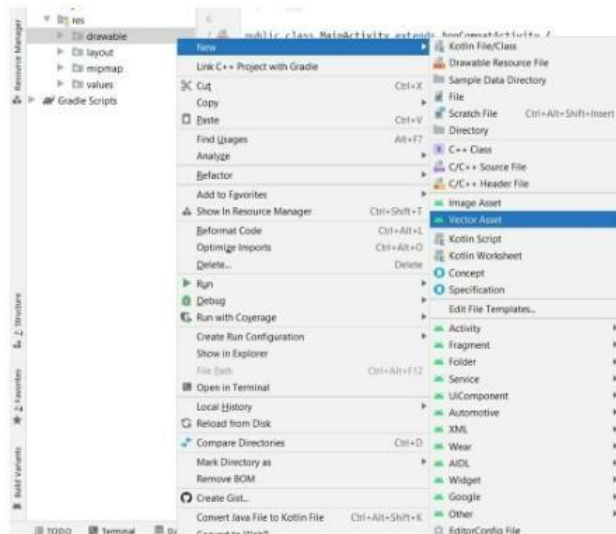


Note: The “@drawable/ic_3_dots”, “@drawable/ic_settings”, “@drawable/ic_about”, “@drawable/ic_exit” are vectors used for Options menu.

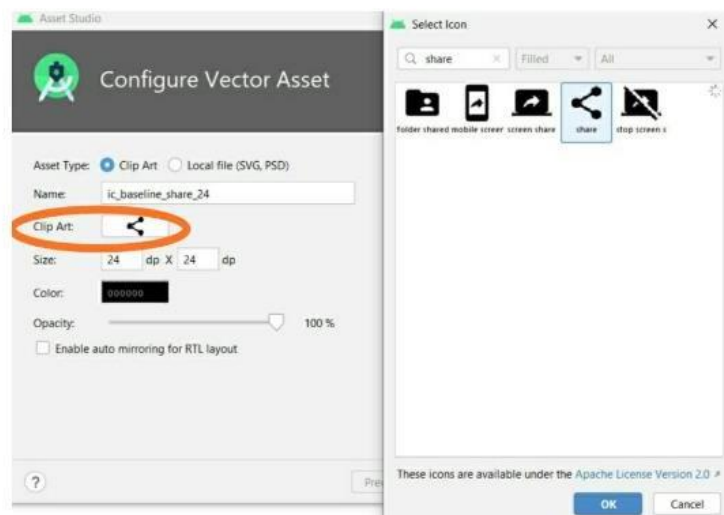
```
<item
    android:id="@+id/overflowMenu"
    android:icon="@drawable/ic_3_dots"
    android:title="Options"
    app:showAsAction="always">
    <menu>
        <item
            android:id="@+id/settings"
            android:icon="@drawable/ic_settings"
            android:title="SETTINGS"
            app:showAsAction="never" />
        <item
            android:id="@+id/about"
            android:icon="@drawable/ic_about"
            android:title="ABOUT"
            app:showAsAction="never" />
        <item
            android:id="@+id/exit"
            android:icon="@drawable/ic_exit"
            android:title="EXIT"
            app:showAsAction="never" />
    </menu>
</item>
</menu>
```

Follow the steps below to define vectors in your project:

Step 1: Right-click on drawable > New > Vector Asset

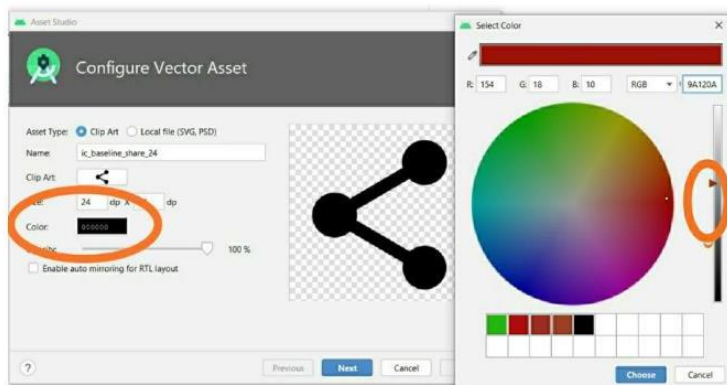


Step 2: Click on Clip Art and Search for the Icons and click ok button

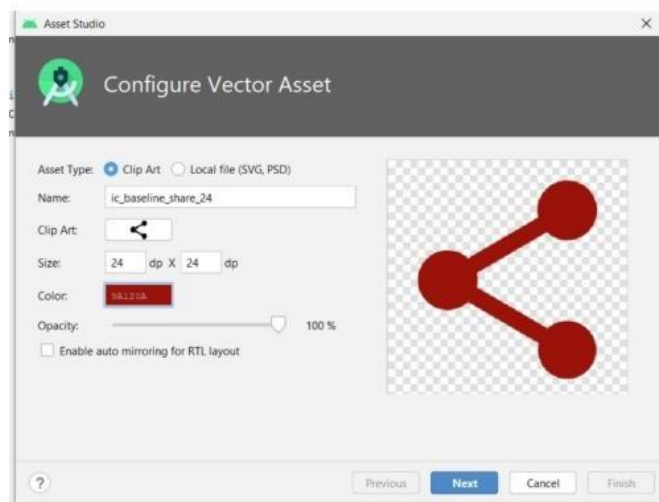


Step 3: Change the color of the icon

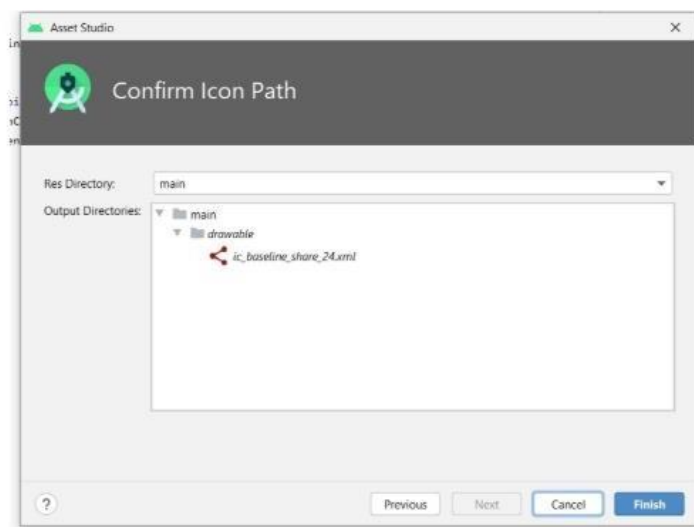
Icon color can be changed either by directly adding color code or by adjusting the color using brightness and then click on **choose** button.



Step 4: Click Next



Step 5: Now Click on Finish Button



Step 6: Icon is created in the drawable folder as shown in the image

