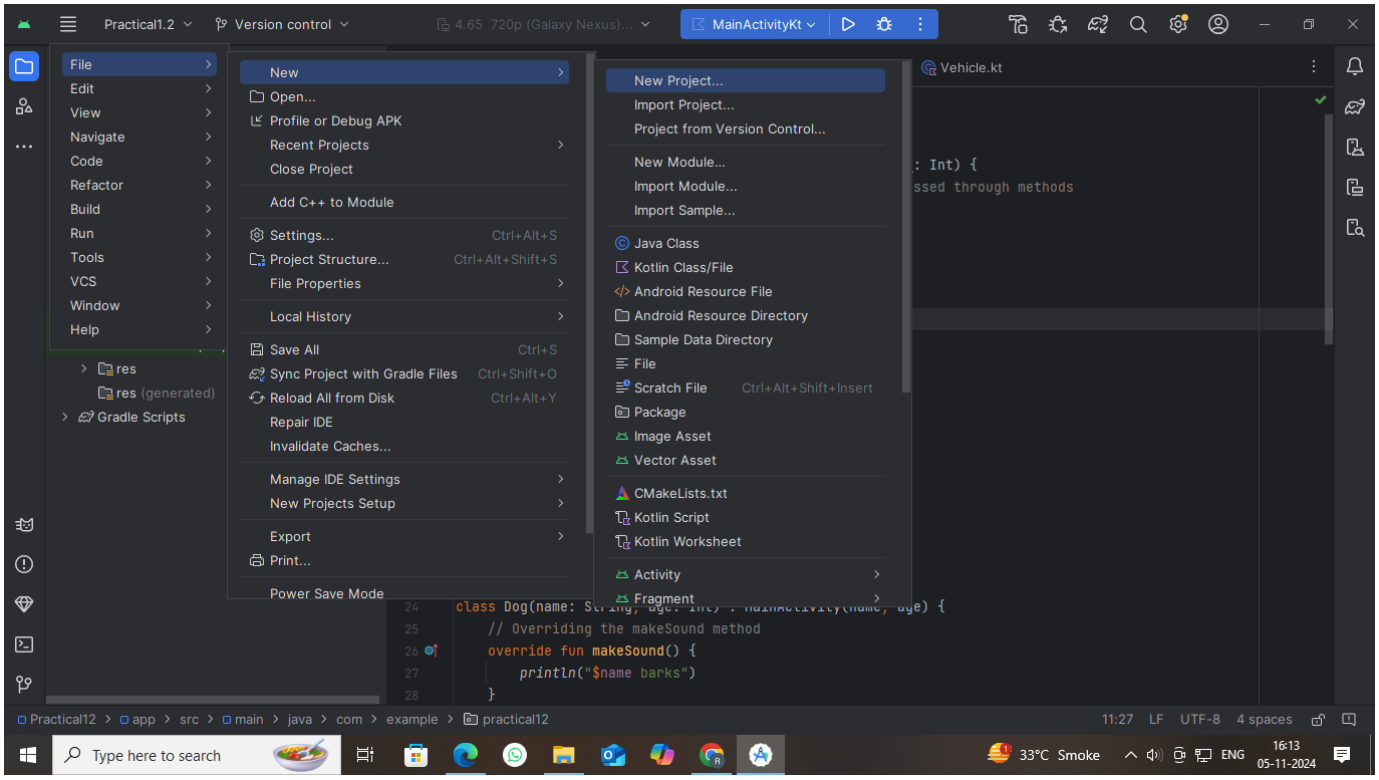
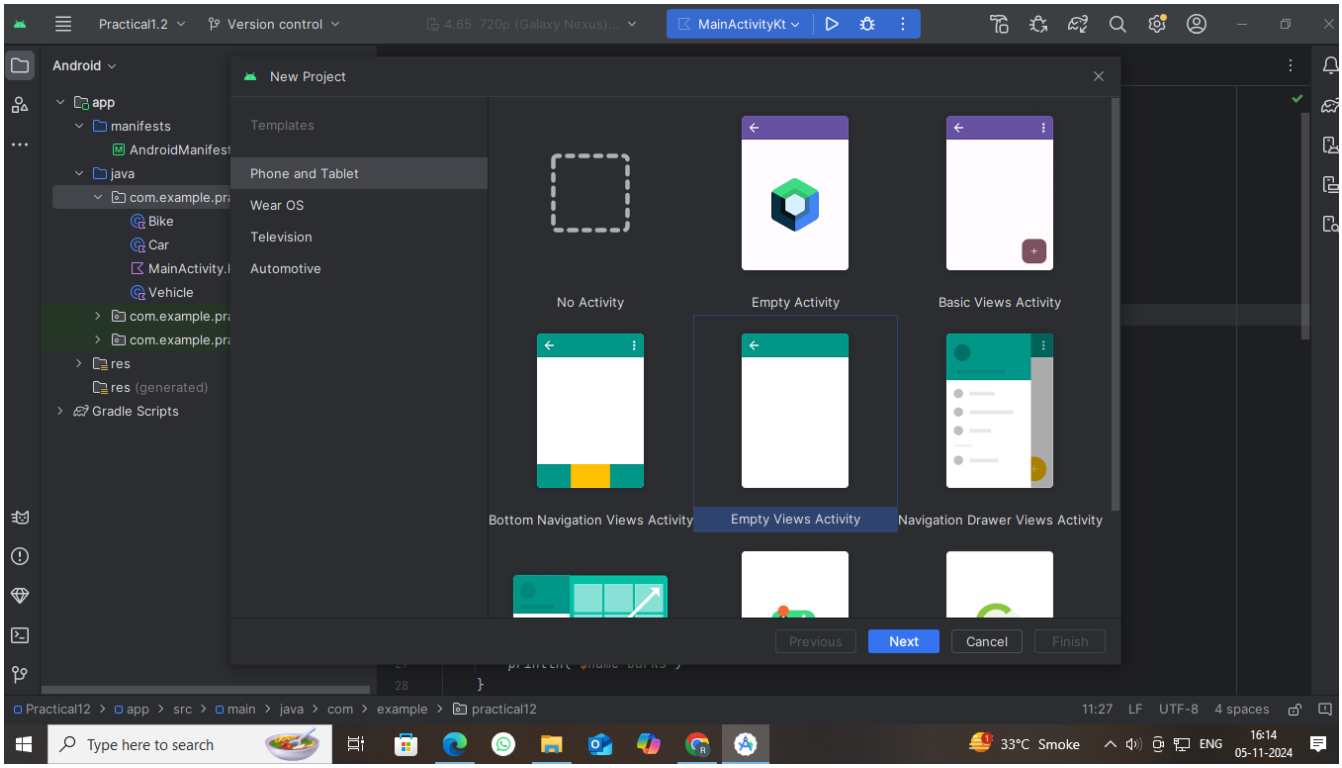
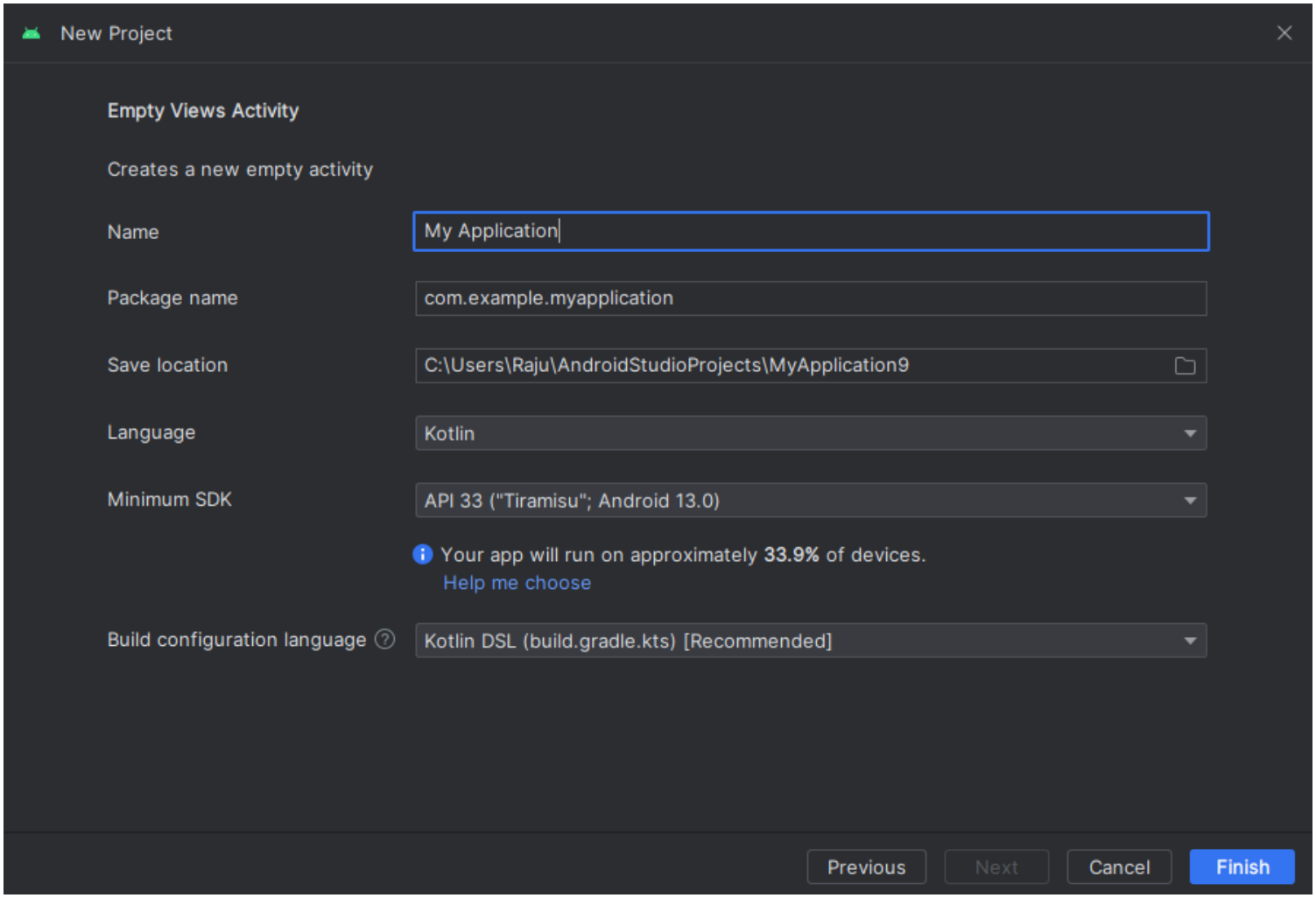
Android Application Development(Practical)

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




Practical1.2
Version control
MainActivityKt
Android
New Project
Templates
Phone and Tablet
Wear OS
Television
Automotive
No Activity
Empty Activity
Basic Views Activity
Bottom Navigation Views Activity
Empty Views Activity
Navigation Drawer Views Activity
Previous
Next
Cancel
Finish
app
manifests
java
com.example.pra
Bike
Car
Vehicle
res
Gradle Scripts
Practical12 > app > src > main > java > com > example > practical12
11:27 LF UTF-8 4 spaces
Type here to search
33°C Smoke
ENG
16:14
05-11-2024



This code is written in Kotlin, and it demonstrates the use of various control structures, including if-else statements, when statements, and different types of loops (for and while).

1. If-Else Statement

The code starts by checking if a variable number is positive, negative, or zero using an if-else statement:

```
val number = 10

if (number > 0) {

    println("$number is a positive number.")
```

```
} else if (number < 0) {
    println("$number is a negative number.")
} else {
    println("$number is zero.")
}
```

- The if statement checks if number is greater than 0, in which case it prints "10 is a positive number."
- The else if statement checks if number is less than 0, printing if it's negative.
- The else statement covers the remaining case, where number is zero.

so the output will be:

10 is a positive number.

2. When Statement

Next, the code uses a when statement, similar to a switch in other languages, to assign a day name based on the value of dayOfWeek:

```
val dayOfWeek = 3
val dayName = when (dayOfWeek) {
    1 -> "Monday"
    2 -> "Tuesday"
    3 -> "Wednesday"
    4 -> "Thursday"
    5 -> "Friday"
```

```
    6 -> "Saturday"
    7 -> "Sunday"
    else -> "Invalid day"
}
println("Day $dayOfWeek is $dayName.")
```

- when checks each condition for dayOfWeek and assigns the corresponding day name to dayName.
- If dayOfWeek doesn't match any condition (i.e., it's not between 1 and 7), it assigns "Invalid day".

Here, dayOfWeek is 3, so dayName becomes "Wednesday", and the output is:

Day 3 is Wednesday.

3. For Loop

The code then demonstrates a for loop that iterates from 1 to 5:

```
println("Numbers from 1 to 5:")
for (i in 1..5) {
    println(i)
}
```

- This loop prints numbers from 1 to 5, with i taking each value in the range 1..5.
- The output is:

Numbers from 1 to 5:

1

2

3

4

5

4. For Loop with Step

Here’s another for loop, this time with a step of 2:

```
println("Even numbers from 1 to 10:")

for (i in 1..10 step 2) {

    println(i)

}
```

- The step keyword allows skipping numbers in the range.
- In this case, it iterates from 1 to 10, printing every second number.

The output is:

Even numbers from 1 to 10:

1

3

5

7

9

5. While Loop

Finally, a while loop is used to create a countdown:

```
var count = 5

println("Countdown:")

while (count > 0) {

    println(count)

    count--

}

println("Lift off!")
```

- while continues looping as long as count is greater than 0.
- count-- decreases the value of count by 1 each iteration.
- When count reaches 0, the loop ends, and "Lift off!" is printed.

```
Countdown:

5

4

3

2

1
```

Lift off!

Conclusion:-

This code demonstrates the basics of control structures in Kotlin, covering conditional statements and loop types.

Code

```
package com.example.myapplication

    fun main() {
        // Control Structures

        // If-Else Statement
        val number = 10
        if (number > 0) {
            println("$number is a positive number.")
        } else if (number < 0) {
            println("$number is a negative number.")
        } else {
            println("$number is zero.")
        }

        // When Statement
        val dayOfWeek = 3
        val dayName = when (dayOfWeek) {
            1 -> "Monday"
            2 -> "Tuesday"
            3 -> "Wednesday"
            4 -> "Thursday"
            5 -> "Friday"
            6 -> "Saturday"
            7 -> "Sunday"
            else -> "Invalid day"
```

```
    }
    println("Day $dayOfWeek is $dayName.")

    // For Loop
    println("Numbers from 1 to 5:")
    for (i in 1..5) {
        println(i)
    }

    // For Loop with step
    println("Even numbers from 1 to 10:")
    for (i in 1..10 step 2) {
        println(i)
    }

    // While Loop
    var count = 5
    println("Countdown:")
    while (count > 0) {
        println(count)
        count--
    }
    println("Lift off!")
}
```

OUTPUT

```
10 is a positive number.

Day 3 is Wednesday.

Numbers from 1 to 5:

1

2
```

3

4

5

Even numbers from 1 to 10:

1

3

5

7

9

Countdown:

5

4

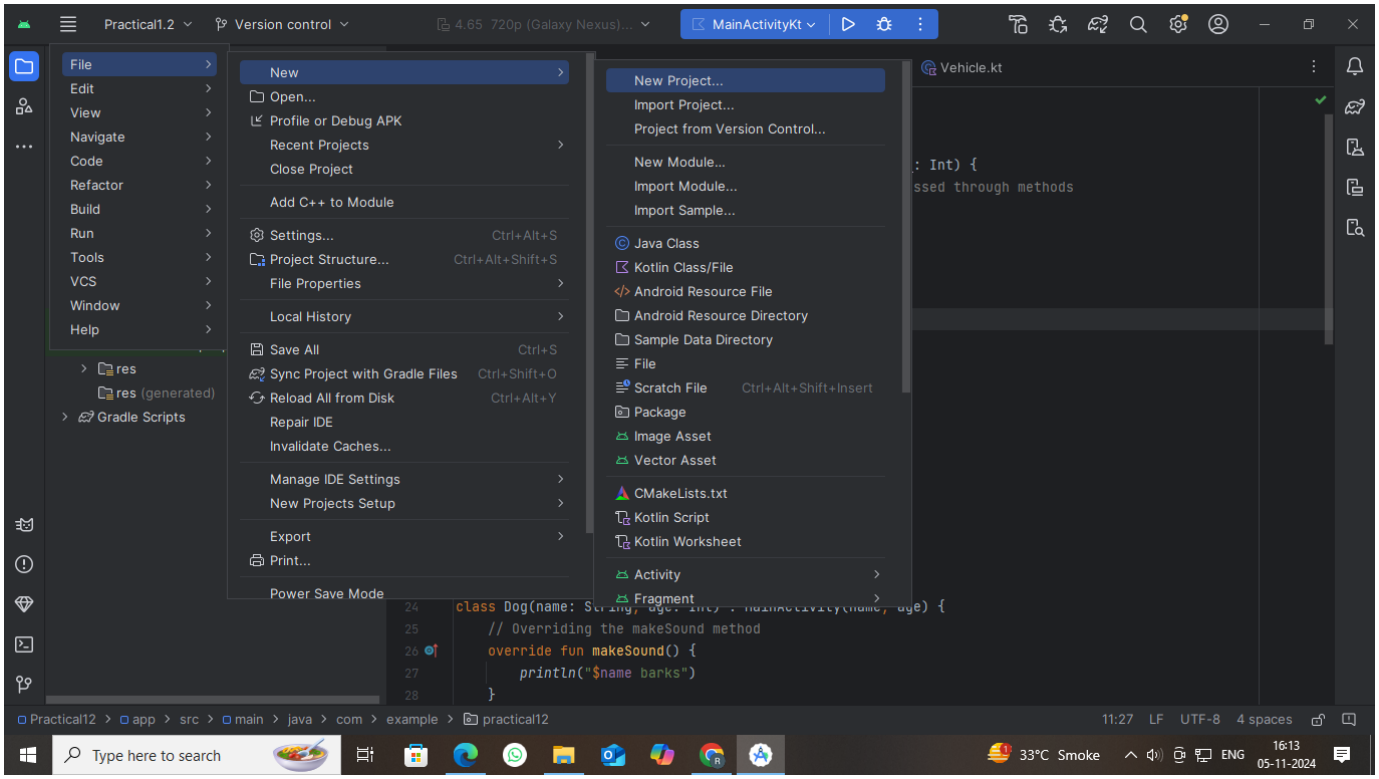
3

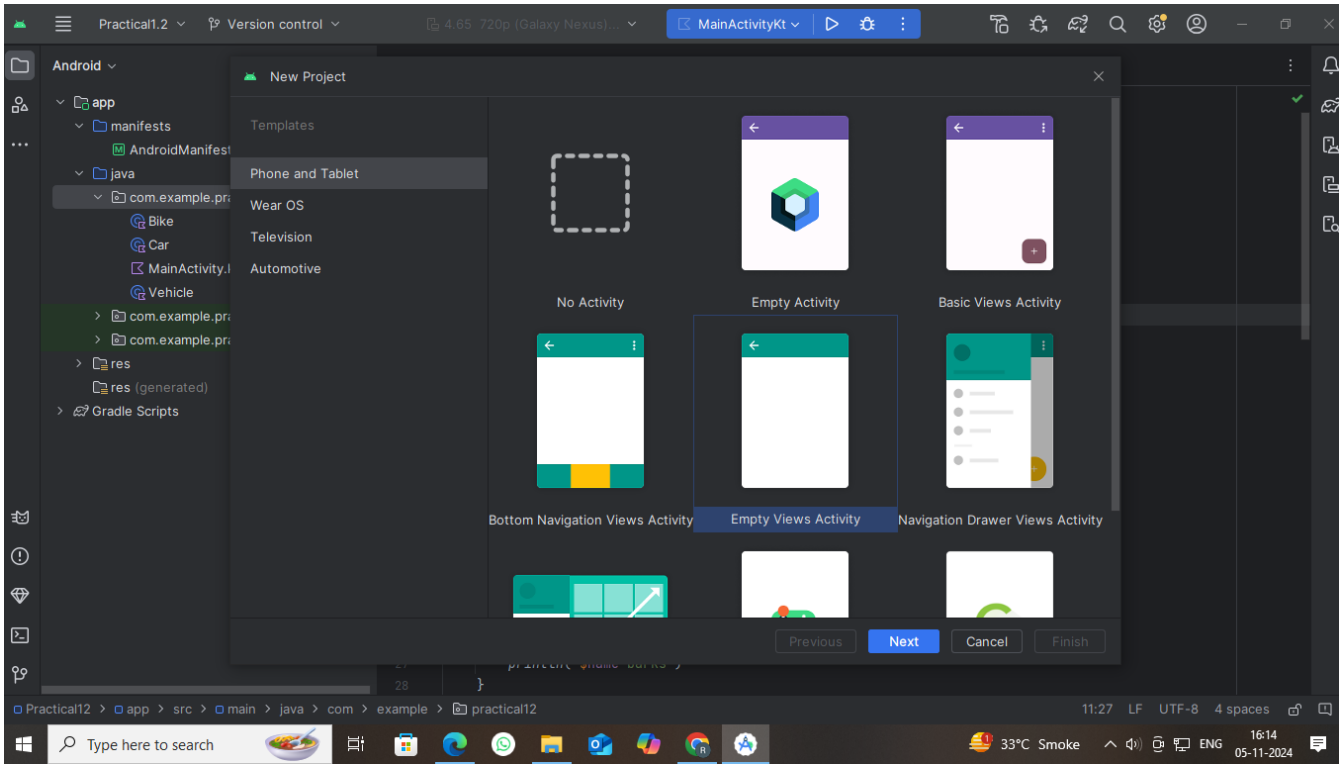
2

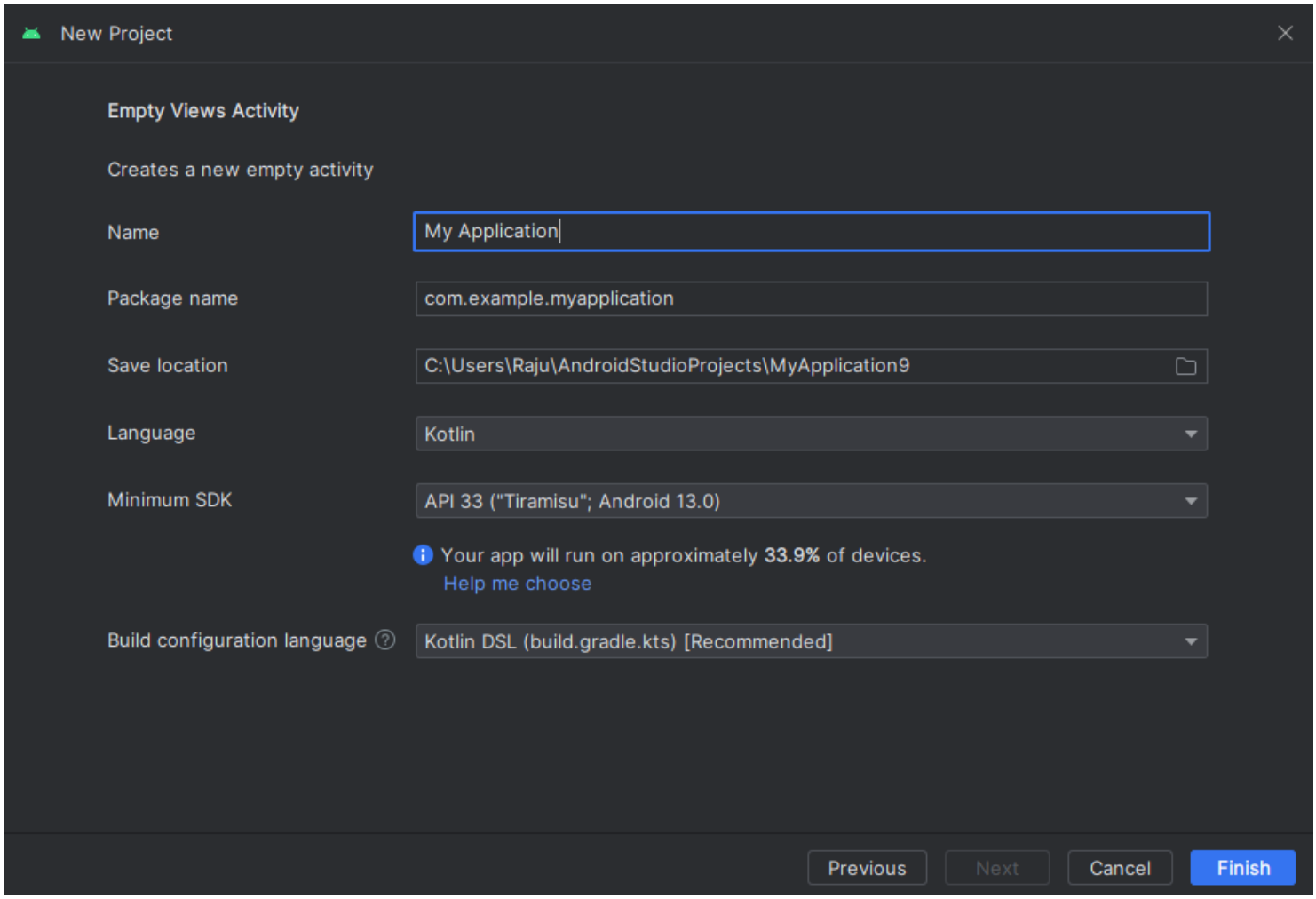
1

Lift off!

2) Write a program to implement object-oriented using Kotlin







✓ package com.example.practical12

This line declares a package named com.example.practical12, which helps organize your code and prevents naming conflicts.

✓ Class: MainActivity

```
open class MainActivity(val name: String, private var age: Int) {

    // Encapsulation: Age is private and can only be accessed through methods

    fun getAge(): Int {

        return age
```

```
    }
    fun setAge(newAge: Int) {
        if (newAge >= 0) {
            age = newAge
        } else {
            println("Age cannot be negative.")
        }
    }
    open fun makeSound() {
        println("Animal makes a sound")
    }
}
```

1) **Open Class**: The open keyword allows this class to be inherited by other classes. By default, classes in Kotlin are final, meaning they cannot be inherited unless specified.
2) **Constructor Parameters**: The class takes two parameters: name (a String) and age (an Int). The name parameter is public, while age is private, demonstrating encapsulation. The private modifier means that age can only be accessed and modified through methods within this class.
3) **Encapsulation**:

- getAge(): A public method that returns the value of the private age.
- setAge(newAge: Int): A public method to set the age. It includes a check to ensure the new age is non-negative, demonstrating good practice in data validation.

4) **Virtual Method**: The makeSound() method is declared open, allowing derived classes to override it. The default implementation prints a generic sound message.

✓ Derived Class: Dog

```
class Dog(name: String, age: Int) : MainActivity(name, age) {
    // Overriding the makeSound method
    override fun makeSound() {
        println("$name barks")
    }
    fun fetch() {
        println("$name is fetching the ball!")
    }
}
```

1) **Inheritance**: The Dog class inherits from MainActivity, meaning it gets all properties and methods of the base class.

2) **Overriding**: The makeSound() method is overridden to provide a specific implementation for dogs, printing a message that includes the dog's name and the sound it makes.

3) **Additional Method**: The fetch() method is defined to provide specific behavior for the Dog class, allowing the dog to perform a fetching action.

✓ Another Derived Class: Cat

```
class Cat(name: String, age: Int) : MainActivity(name, age) {
    // Overriding the makeSound method
    override fun makeSound() {
        println("$name meows")
```

```
    }

    fun scratch() {

        println("$name is scratching the furniture!")

    }

}
```

1) **Similar Structure**: The Cat class also inherits from MainActivity, and it follows the same structure as Dog.
2) **Overriding**: It overrides makeSound() to provide a specific implementation for cats.
3) **Specific Method**: The scratch() method allows the cat to demonstrate behavior specific to its class.

✓ Main Function

```
fun main() {

    // Creating instances of Dog and Cat

    val myDog = Dog("Buddy", 3)

    val myCat = Cat("Whiskers", 2)

    // Using the methods

    myDog.makeSound() // Output: Buddy barks

    myCat.makeSound() // Output: Whiskers meows



    // Using encapsulated age property

    println("${myDog.name} is ${myDog.getAge()} years old.")

    myDog.setAge(4) // Changing age
```

```
    println("${myDog.name} is now ${myDog.getAge()} years old.")

    // Using specific methods

    myDog.fetch()    // Output: Buddy is fetching the ball!

    myCat.scratch()  // Output: Whiskers is scratching the furniture!

}
```

1) **Instance Creation**: Two instances are created: myDog of type Dog and myCat of type Cat, each initialized with a name and age.
2) **Method Calls**:

- The makeSound() method is called on both instances, demonstrating polymorphism; the correct method is invoked based on the object's actual type.

- The age is accessed using getAge() and modified using setAge() to illustrate encapsulation in practice.

3) **Specific Behaviors**: The specific methods fetch() and scratch() demonstrate additional behaviors unique to each derived class.

Conclusion

This program effectively demonstrates key OOP principles in Kotlin:

- **Encapsulation**: The age property is protected from direct access.

- **Inheritance**: Dog and Cat inherit from MainActivity.

- **Polymorphism**: The same method (makeSound()) behaves differently based on the object's class.

- **Class Design**: The structure promotes code reuse and organization.

CODE

```
package com.example.practical12

// Base class
open class MainActivity(val name: String, private var age: Int) {
    // Encapsulation: Age is private and can only be accessed through methods
    fun getAge(): Int {
        return age
    }

    fun setAge(newAge: Int) {
        if (newAge >= 0) {
            age = newAge
        } else {
            println("Age cannot be negative.")
        }
    }

    open fun makeSound() {
        println("Animal makes a sound")
    }
}

// Derived class
class Dog(name: String, age: Int) : MainActivity(name, age) {
    // Overriding the makeSound method
    override fun makeSound() {
        println("$name barks")
    }

    fun fetch() {
        println("$name is fetching the ball!")
    }
}

// Another derived class
class Cat(name: String, age: Int) : MainActivity(name, age) {
    // Overriding the makeSound method
    override fun makeSound() {
```

```
        println("$name meows")
    }

    fun scratch() {
        println("$name is scratching the furniture!")
    }
}

fun main() {
    // Creating instances of Dog and Cat
    val myDog = Dog("Buddy", 3)
    val myCat = Cat("Whiskers", 2)

    // Using the methods
    myDog.makeSound() // Output: Buddy barks
    myCat.makeSound() // Output: Whiskers meows

    // Using encapsulated age property
    println("${myDog.name} is ${myDog.getAge()} years old.")
    myDog.setAge(4) // Changing age
    println("${myDog.name} is now ${myDog.getAge()} years old.")

    // Using specific methods
    myDog.fetch()    // Output: Buddy is fetching the ball!
    myCat.scratch()  // Output: Whiskers is scratching the furniture!
}
```

OUTPUT

Buddy barks

Whiskers meows

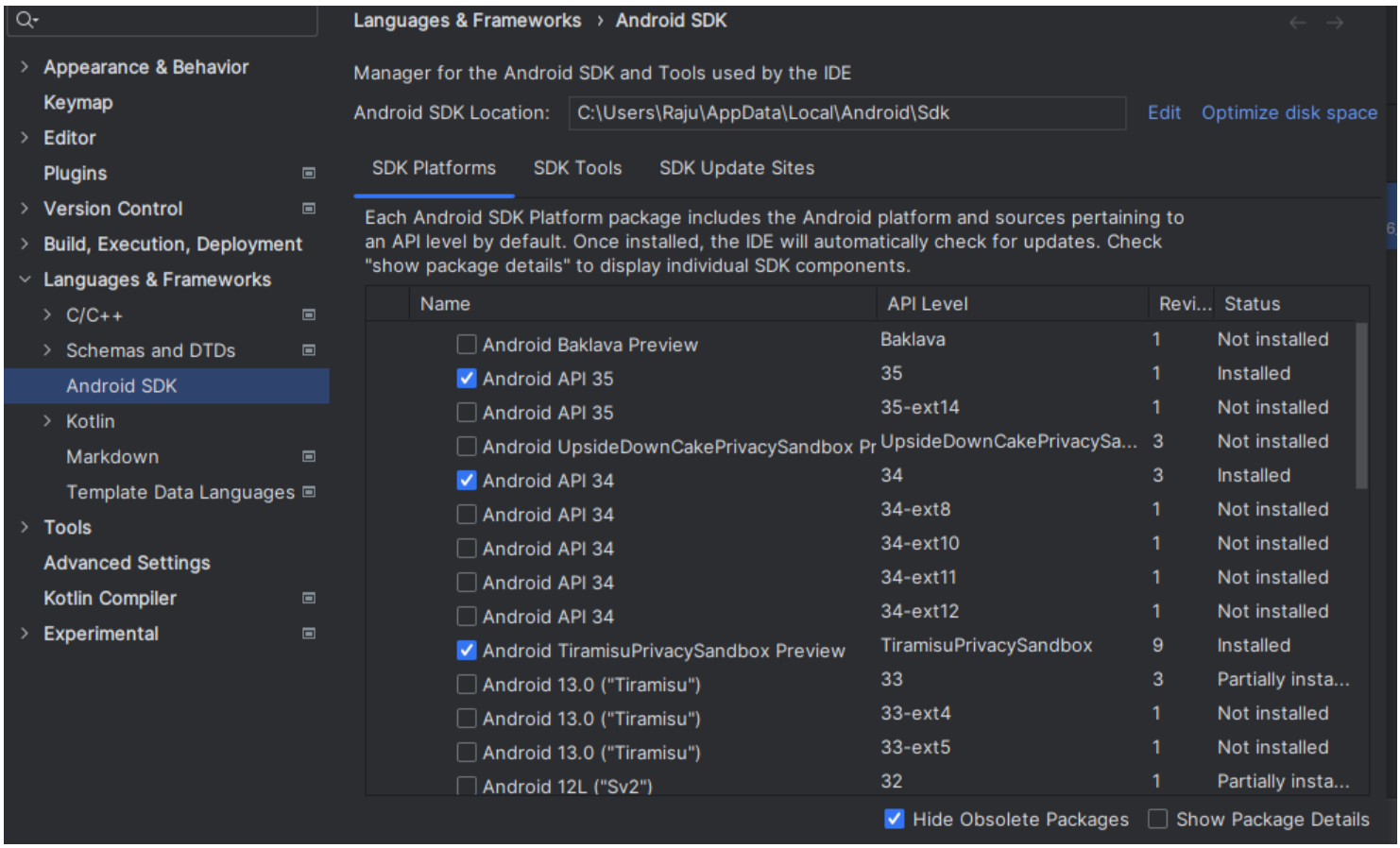
Buddy is 3 years old.

Buddy is now 4 years old.

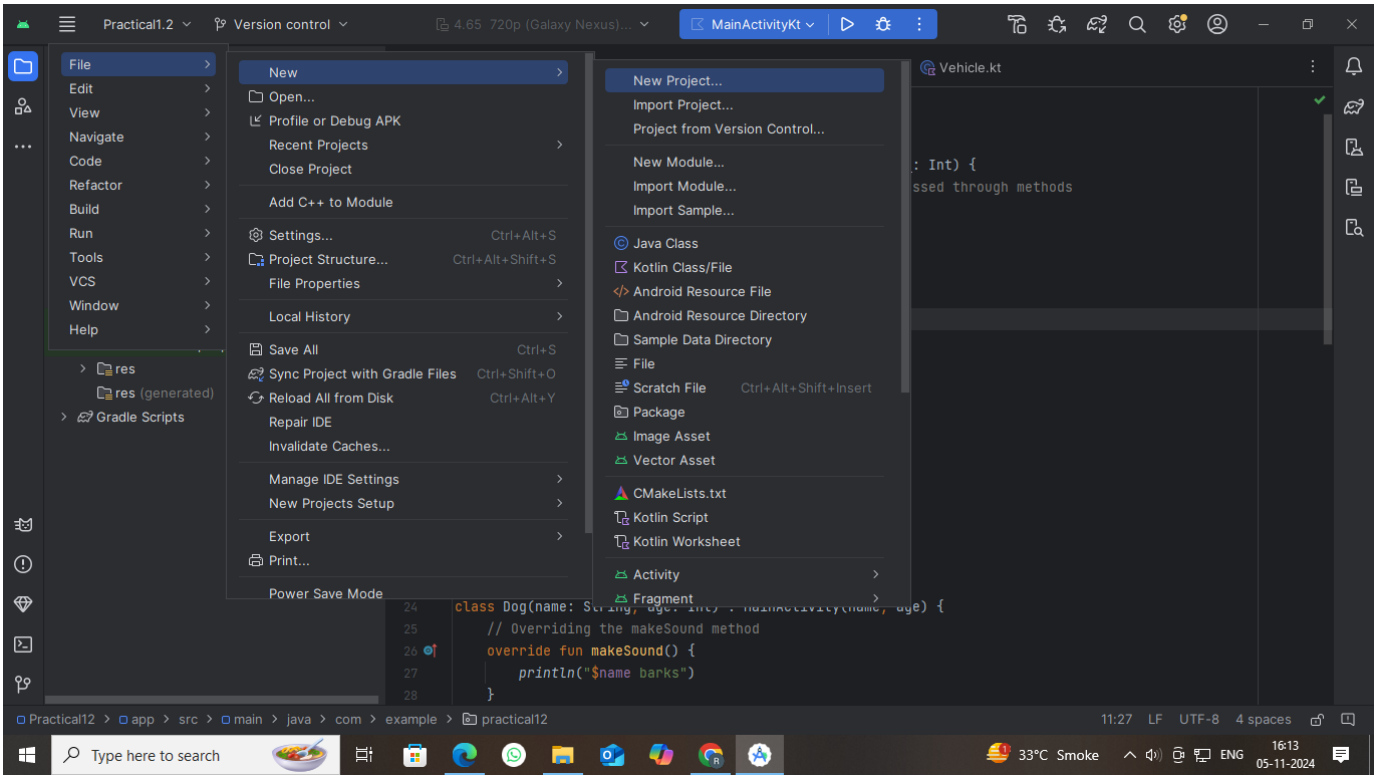
Buddy is fetching the ball!

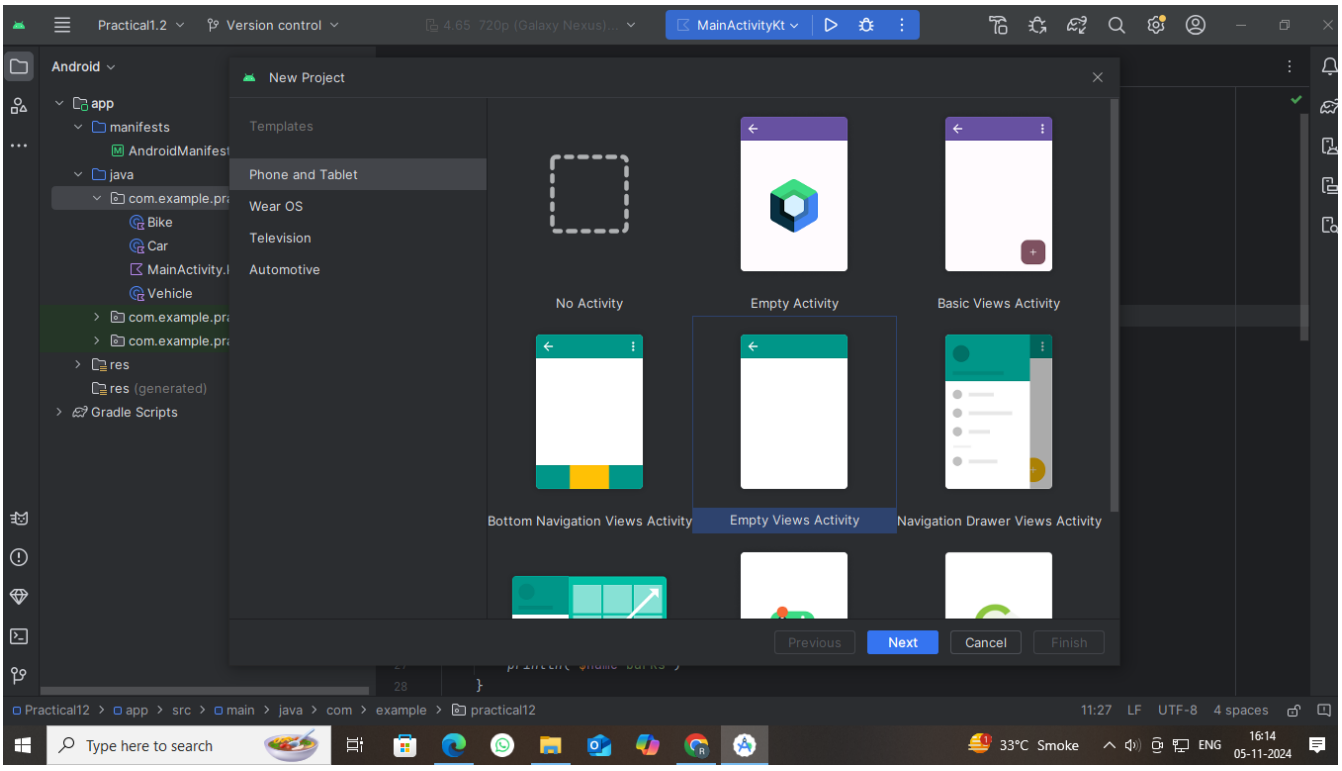
Whiskers is scratching the furniture!

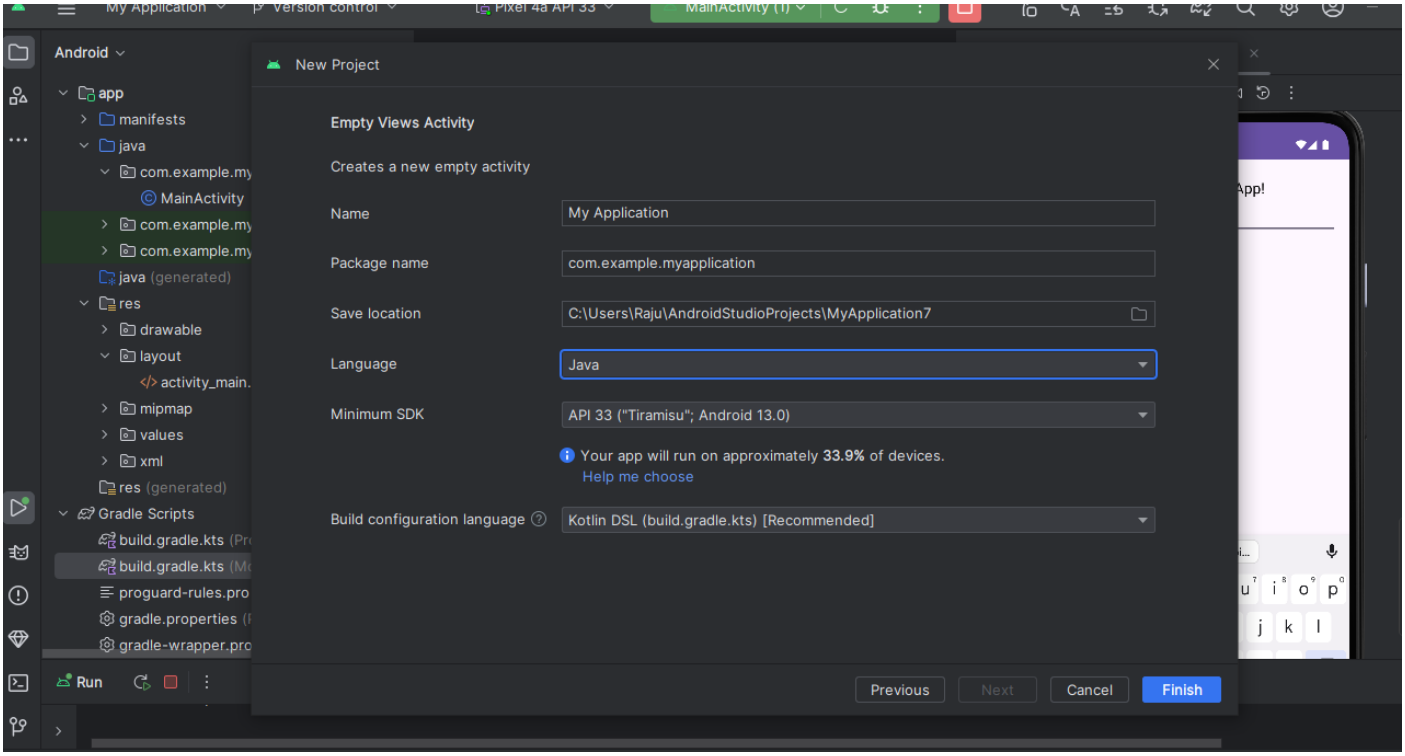
Note:- For SDK Manager ,go to setting and just click on the below.

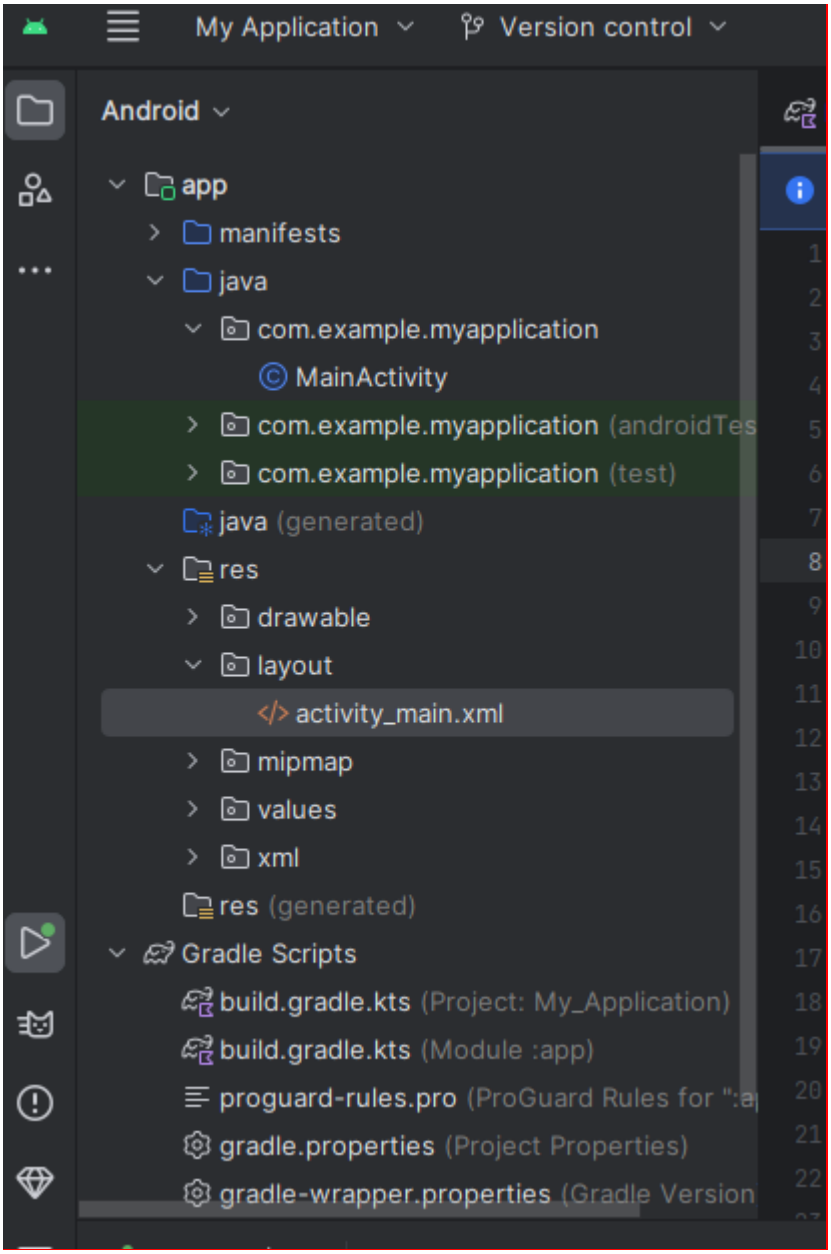


3) Create an android application to design screens using different layouts and UI including button, edittext, textview, radio button etc.

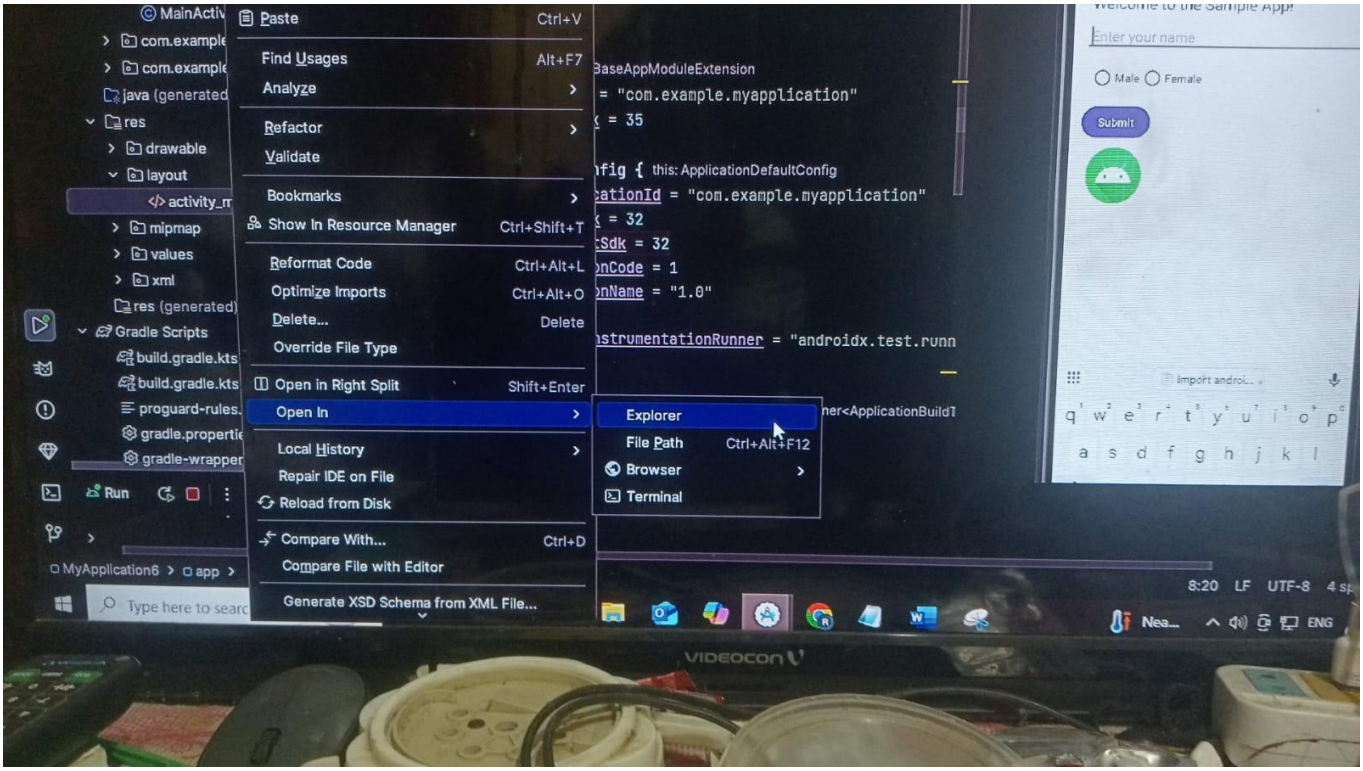

Practical1.2
Version control
MainActivityKt
File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
Window
Help
New
Open...
Profile or Debug APK
Recent Projects
Close Project
Add C++ to Module
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Sync Project with Gradle Files Ctrl+Shift+O
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Export
Print...
Power Save Mode
New Project...
Import Project...
Project from Version Control...
New Module...
Import Module...
Import Sample...
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
CMakeLists.txt
Kotlin Script
Kotlin Worksheet
Activity
Fragment
Vehicle.kt
res
res (generated)
Gradle Scripts
class Dog(name: String, age: Int) : MainActivity(name, age) {
// Overriding the makeSound method
override fun makeSound() {
println("$name barks")
}
Practical12 > app > src > main > java > com > example > practical12
11:27 LF UTF-8 4 spaces
Type here to search
33°C Smoke
ENG
16:13
05-11-2024


Practical1.2
Version control
MainActivityKt
Android
app
manifests
java
com.example.pra
Bike
Car
Vehicle
res
res (generated)
Gradle Scripts
New Project
Templates
Phone and Tablet
Wear OS
Television
Automotive
No Activity
Empty Activity
Basic Views Activity
Bottom Navigation Views Activity
Empty Views Activity
Navigation Drawer Views Activity
Previous
Next
Cancel
Finish
Practical12 > app > src > main > java > com > example > practical12
11:27 LF UTF-8 4 spaces
Type here to search
33°C Smoke
ENG
16:14
05-11-2024

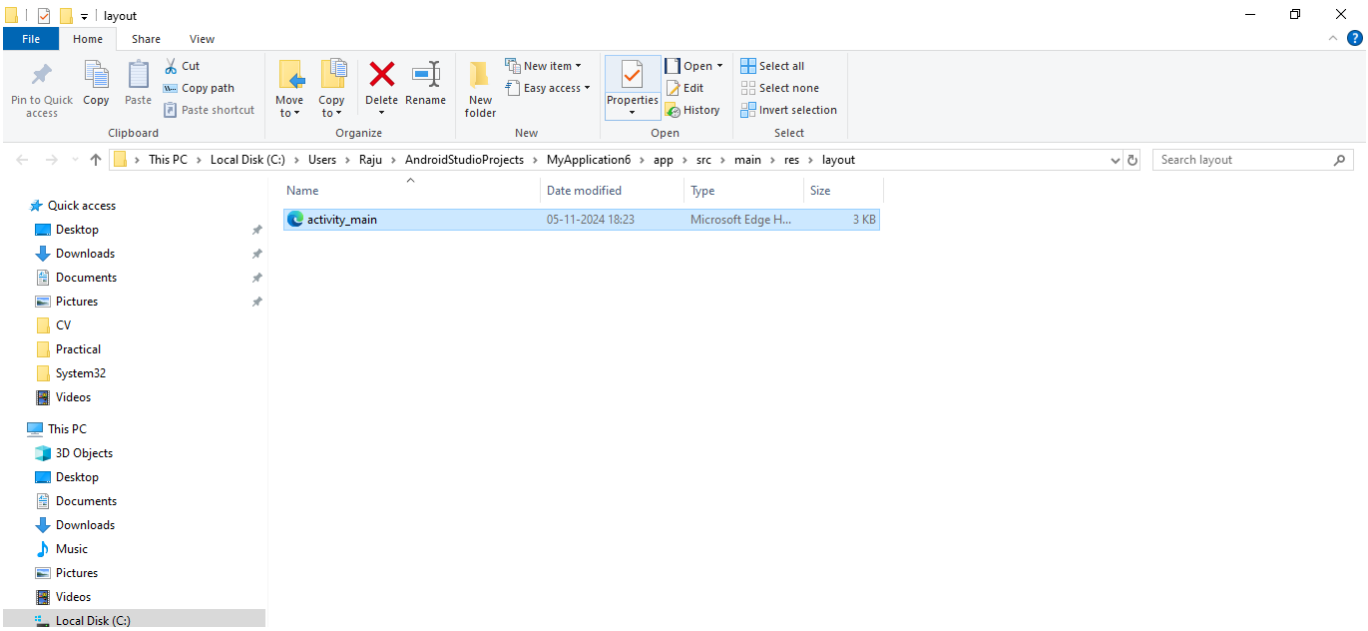

New Project
Empty Views Activity
Creates a new empty activity
Name
My Application
Package name
com.example.myapplication
Save location
C:\Users\Raju\AndroidStudioProjects\MyApplication7
Language
Java
Minimum SDK
API 33 ("Tiramisu"; Android 13.0)
Your app will run on approximately 33.9% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
Previous
Next
Cancel
Finish
Android
app
manifests
java
MainActivity
res
drawable
layout
mipmap
values
xml
Gradle Scripts
Run



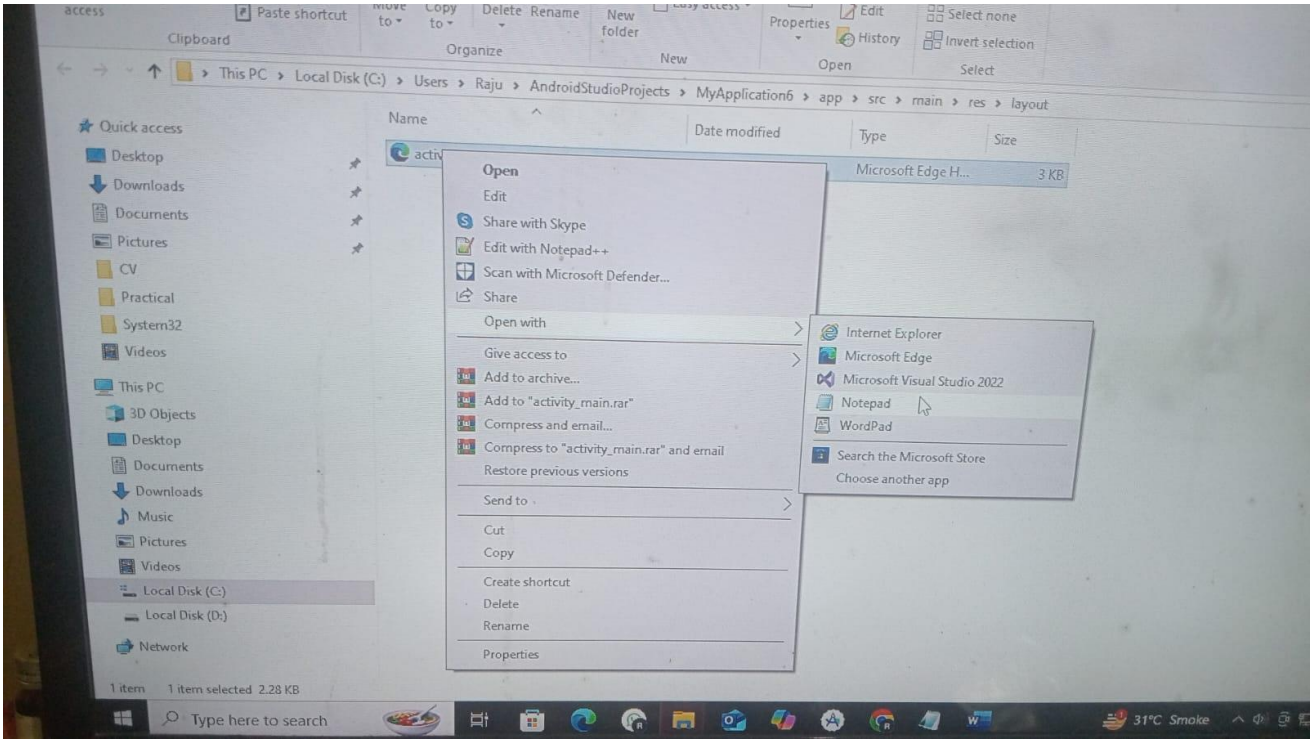
Go to Layout and Right click on activity_main.xml and go here….



Right click on explorer you will see this picture….



Right clik on activity_main and go to open with notepad..



Paste the xml code in notepad and then save it by clicking clt+s.

XML FILE

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
android:orientation="vertical"

android:padding="16dp">


<!-- TextView -->

<TextView

    android:id="@+id/textView"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Welcome to the Sample App!"

    android:textSize="20sp"

    android:textColor="#000000"

    android:padding="8dp" />


<!-- EditText -->

<EditText

    android:id="@+id/editText"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:hint="Enter your name"

    android:padding="8dp" />
```

```
<!-- RadioGroup with RadioButtons -->
<RadioGroup
    android:id="@+id/radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="8dp">

    <RadioButton
        android:id="@+id/radioButtonMale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Male" />

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

```
    </RadioGroup>

    <!-- Button -->
    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:padding="8dp" />

    <!-- ImageView -->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"
        android:contentDescription="Sample Image"
        android:padding="8dp" />

</LinearLayout>
```

```
</ScrollView>
```

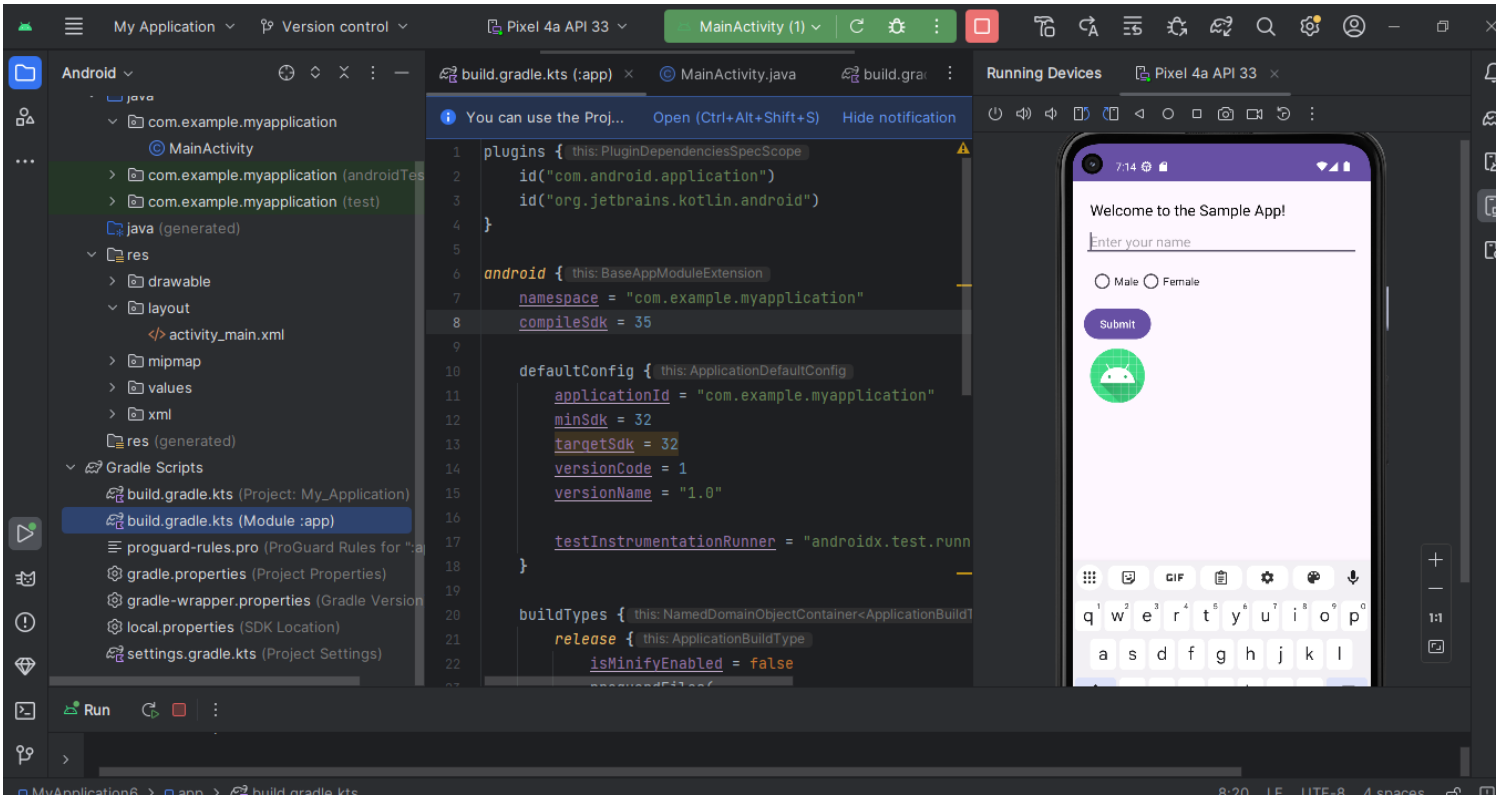
In the java directory, open MainActivity.java (or MainActivity.kt if using Kotlin) and add code to set up actions for your UI components:

```
package com.example.yourapp;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
```

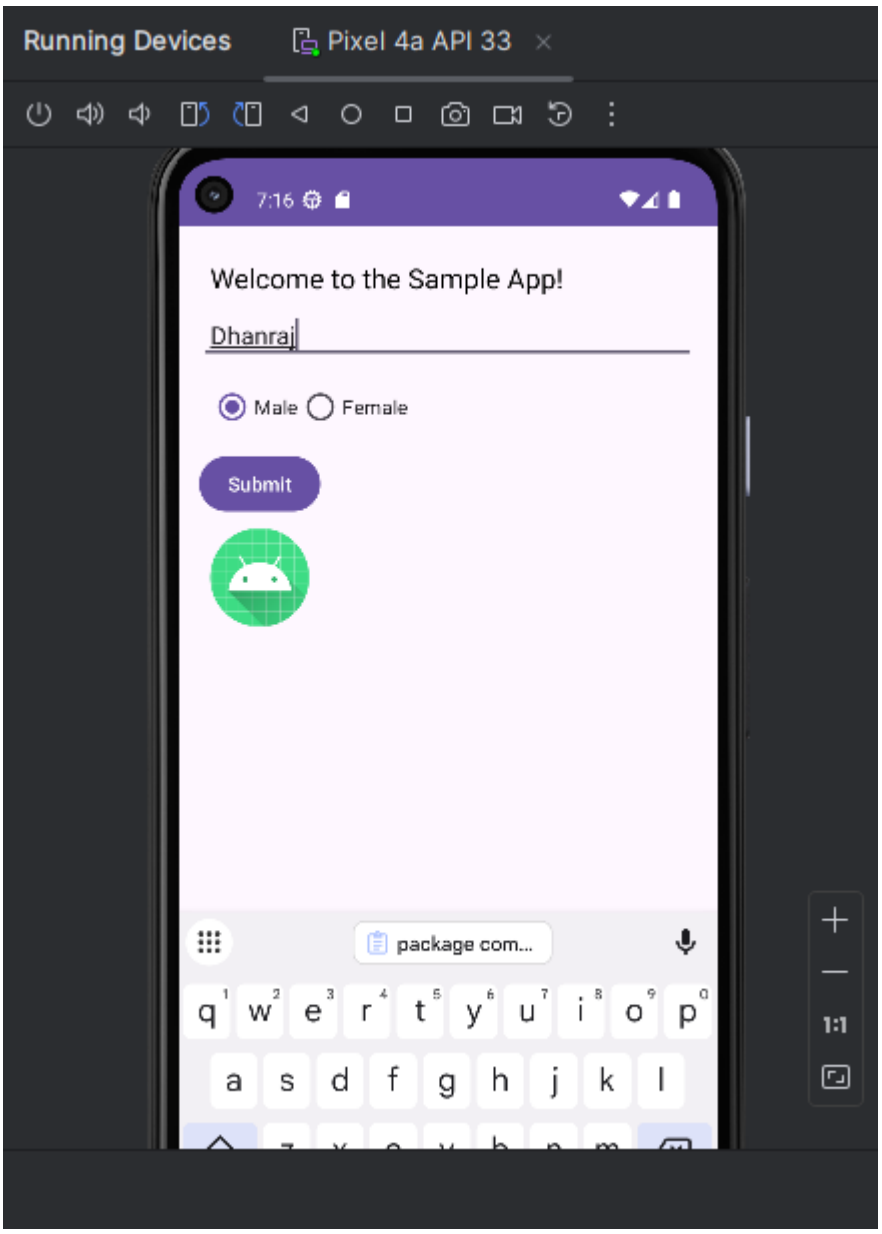
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Find views by ID
    EditText editText = findViewById(R.id.editText);
    RadioGroup radioGroup = findViewById(R.id.radioGroup);
    Button buttonSubmit = findViewById(R.id.buttonSubmit);
    TextView textView = findViewById(R.id.textView);

    // Button onClickListener
    buttonSubmit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String name = editText.getText().toString();

            // Check selected radio button
            int selectedId = radioGroup.getCheckedRadioButtonId();
            RadioButton radioButton = findViewById(selectedId);
            String gender = (radioButton != null) ? radioButton.getText().toString() : "Not selected";
```

```
        // Display result in a Toast
        String message = "Name: " + name + "\nGender: " + gender;
        Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();
    }
});
  }
}
```

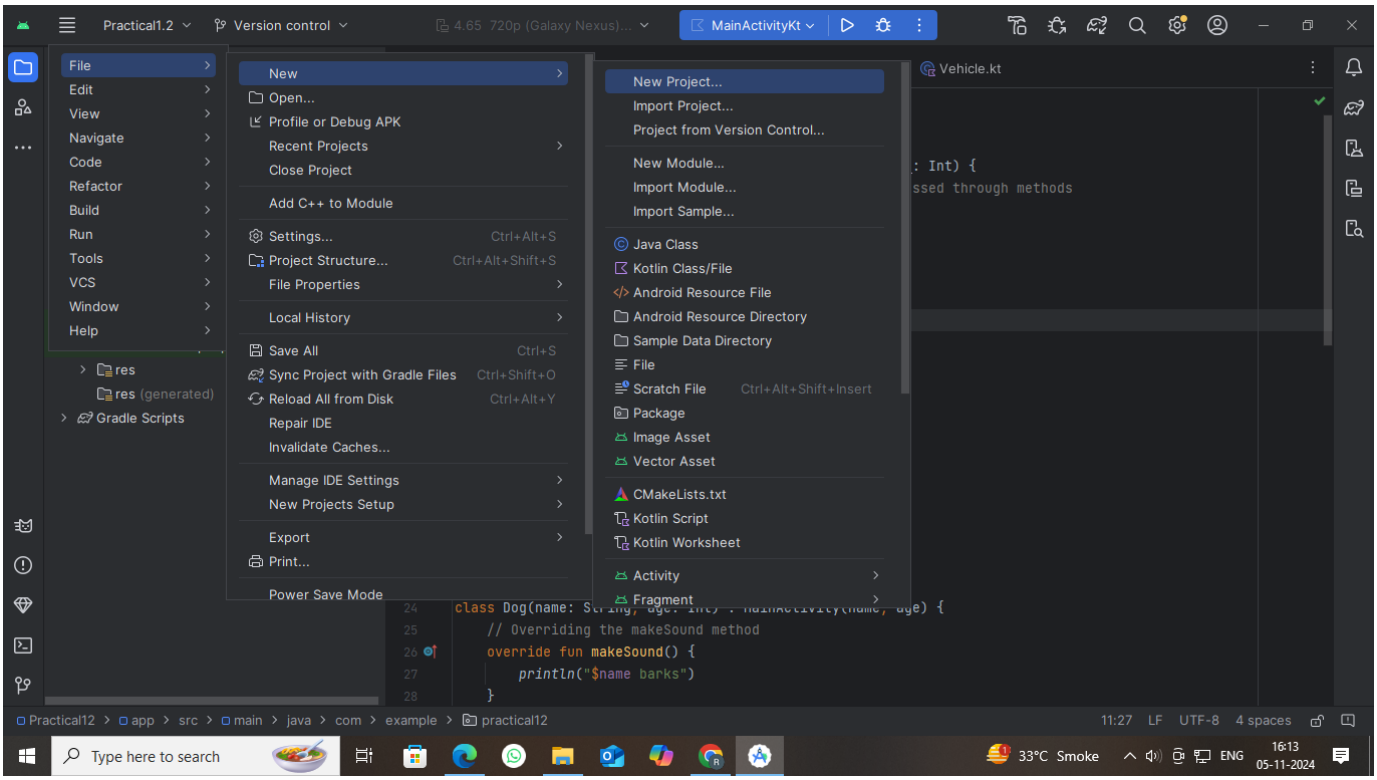
Note:- You have to click on synch now for build.gradle.kt update

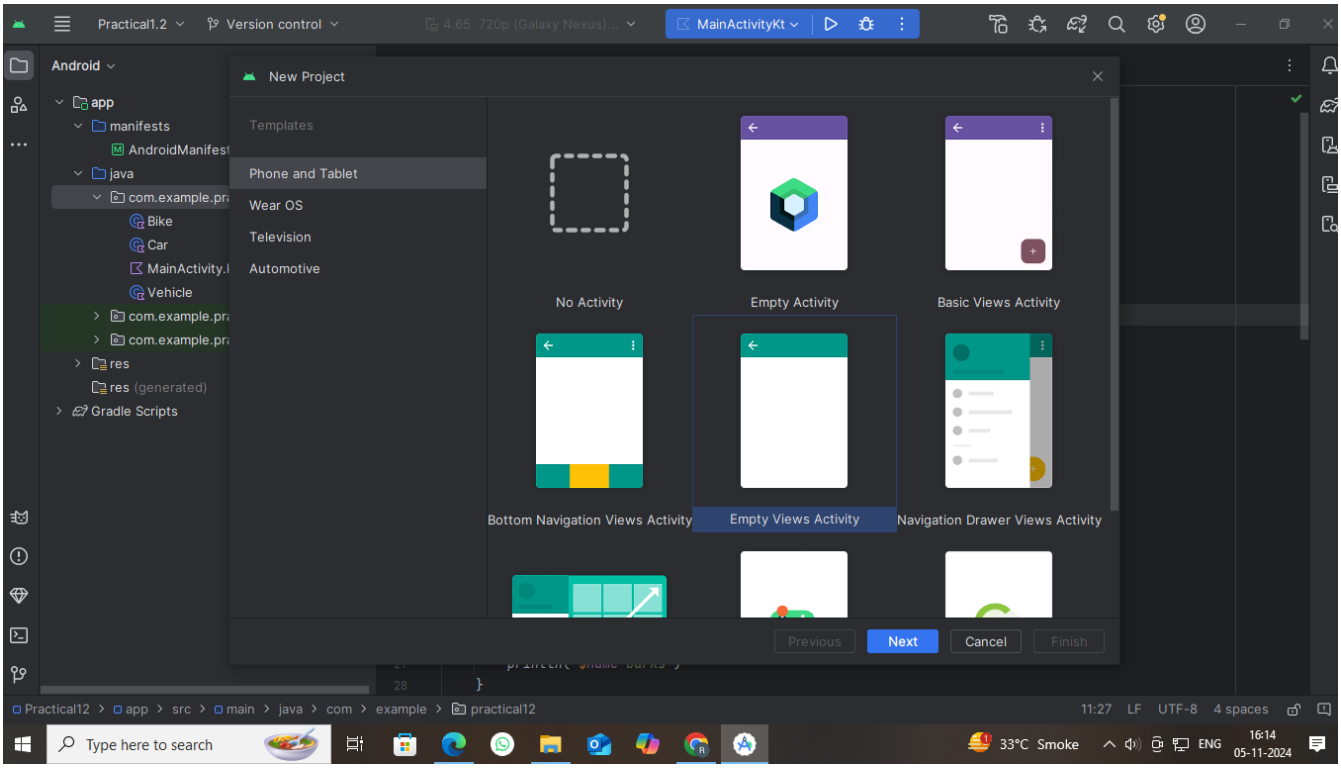


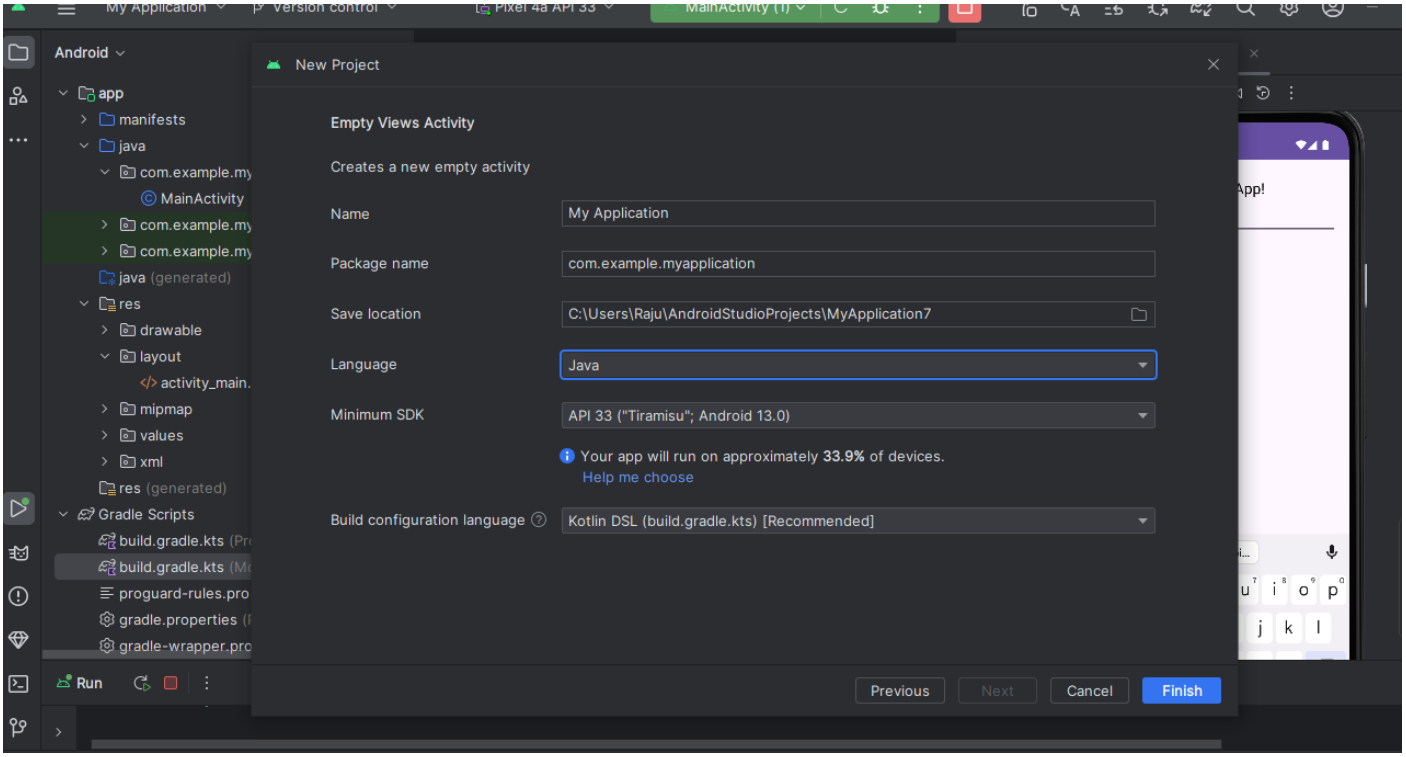
OUTPUT:-

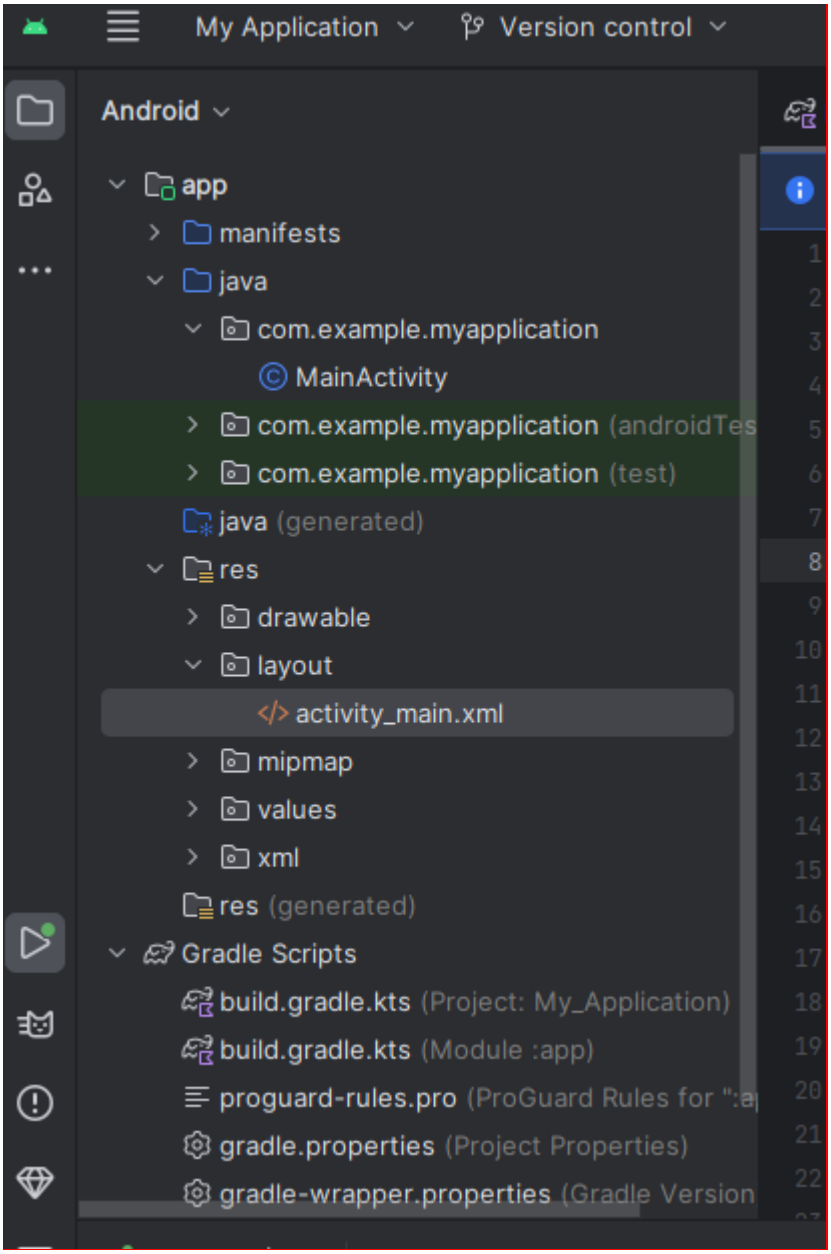

Running Devices
Pixel 4a API 33
7:16
Welcome to the Sample App!
Dhanraj
Male
Female
Submit
package com...
1:1

4) write an android demonstrating response to event/user interaction for checkbox, radio button, button.

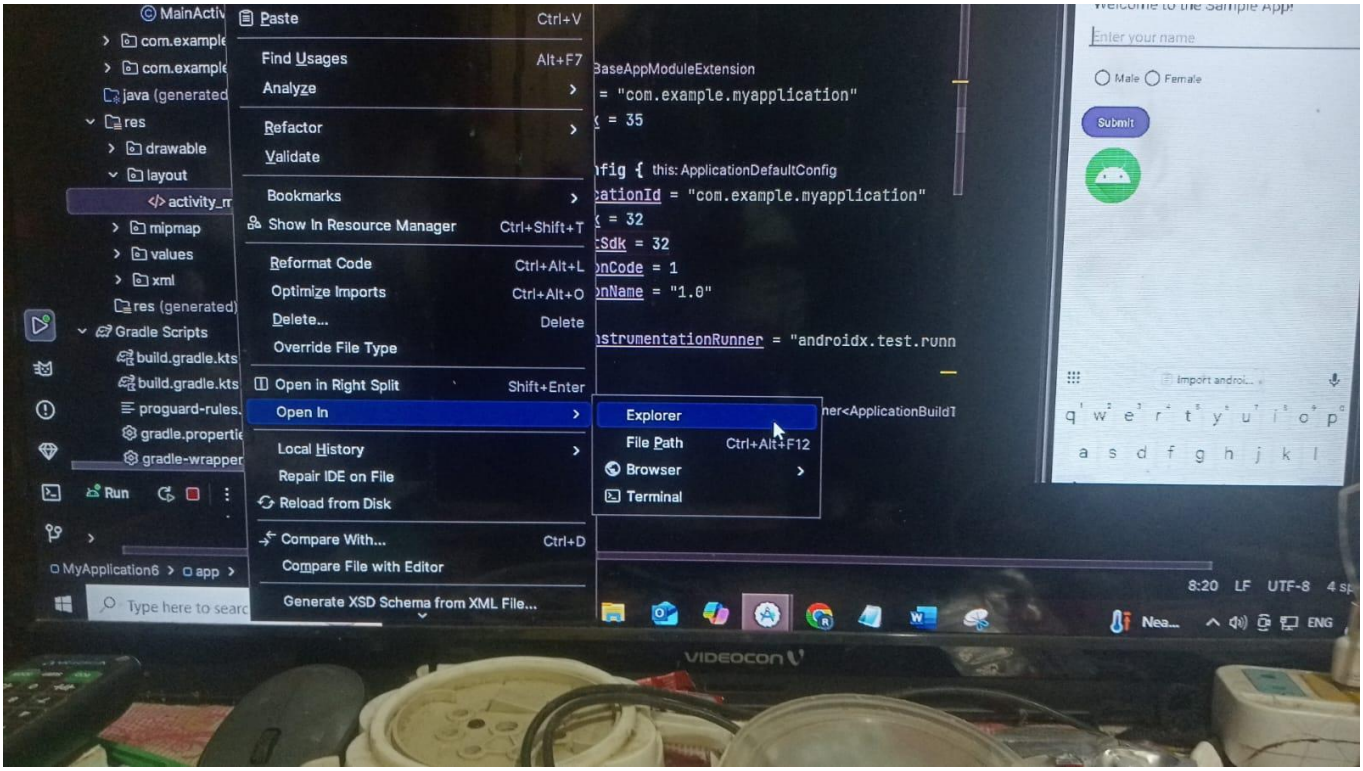




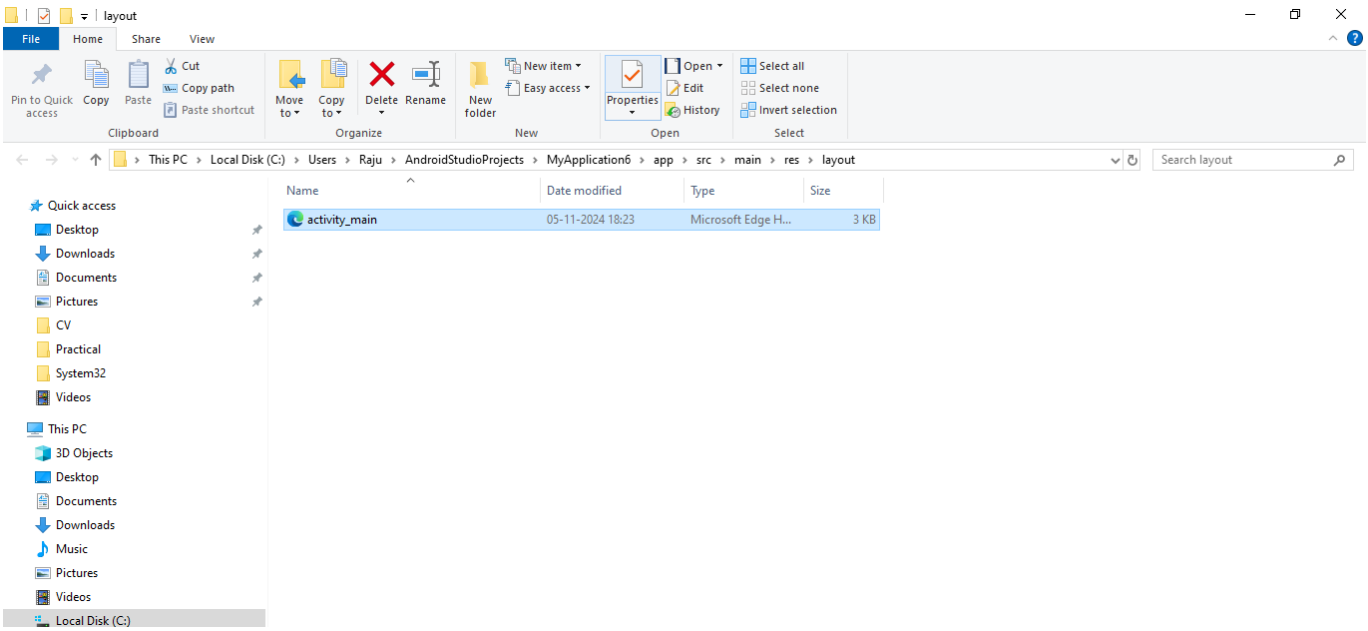

New Project
Empty Views Activity
Creates a new empty activity
Name
My Application
Package name
com.example.myapplication
Save location
C:\Users\Raju\AndroidStudioProjects\MyApplication7
Language
Java
Minimum SDK
API 33 ("Tiramisu"; Android 13.0)
Your app will run on approximately 33.9% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
Previous
Next
Cancel
Finish
Android
app
manifests
java
MainActivity
java (generated)
res
drawable
layout
mipmap
values
xml
res (generated)
Gradle Scripts
build.gradle.kts
proguard-rules.pro
gradle.properties
gradle-wrapper.pro
Run



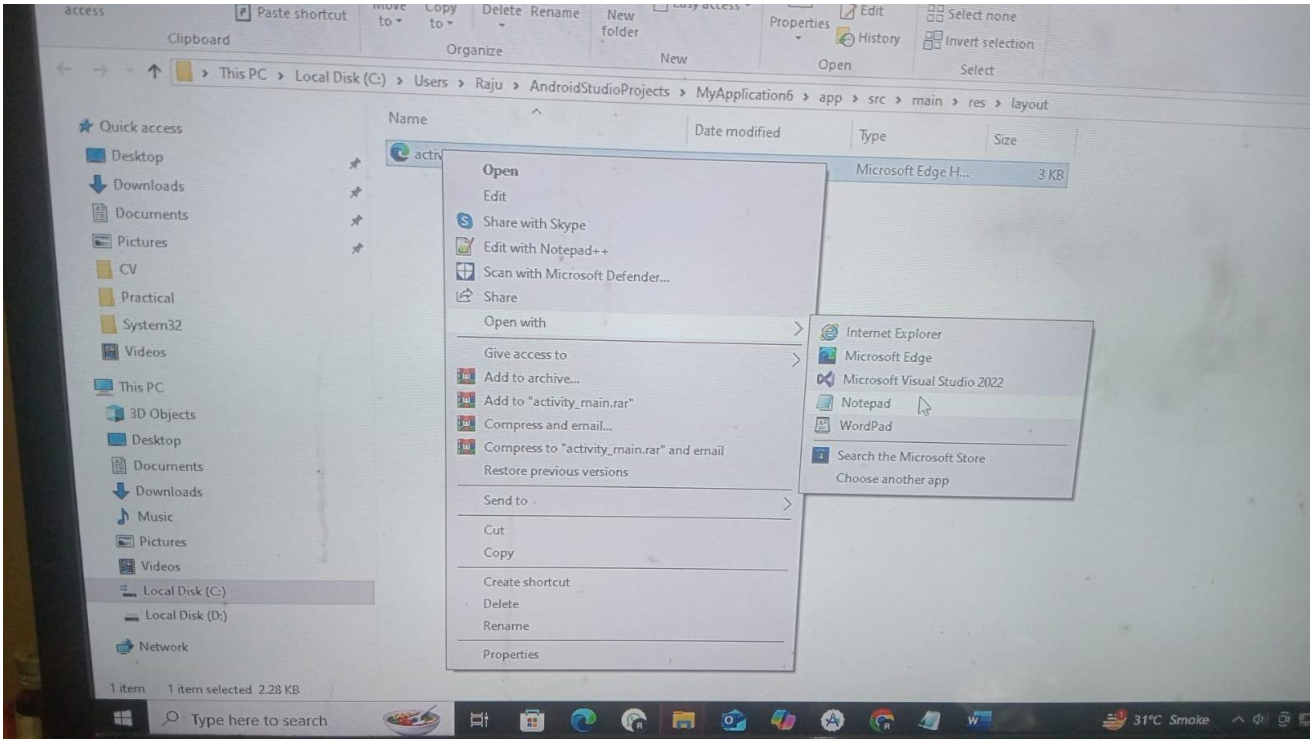
Go to Layout and Right click on activity_main.xml and go here….



Right click on explorer you will see this picture….

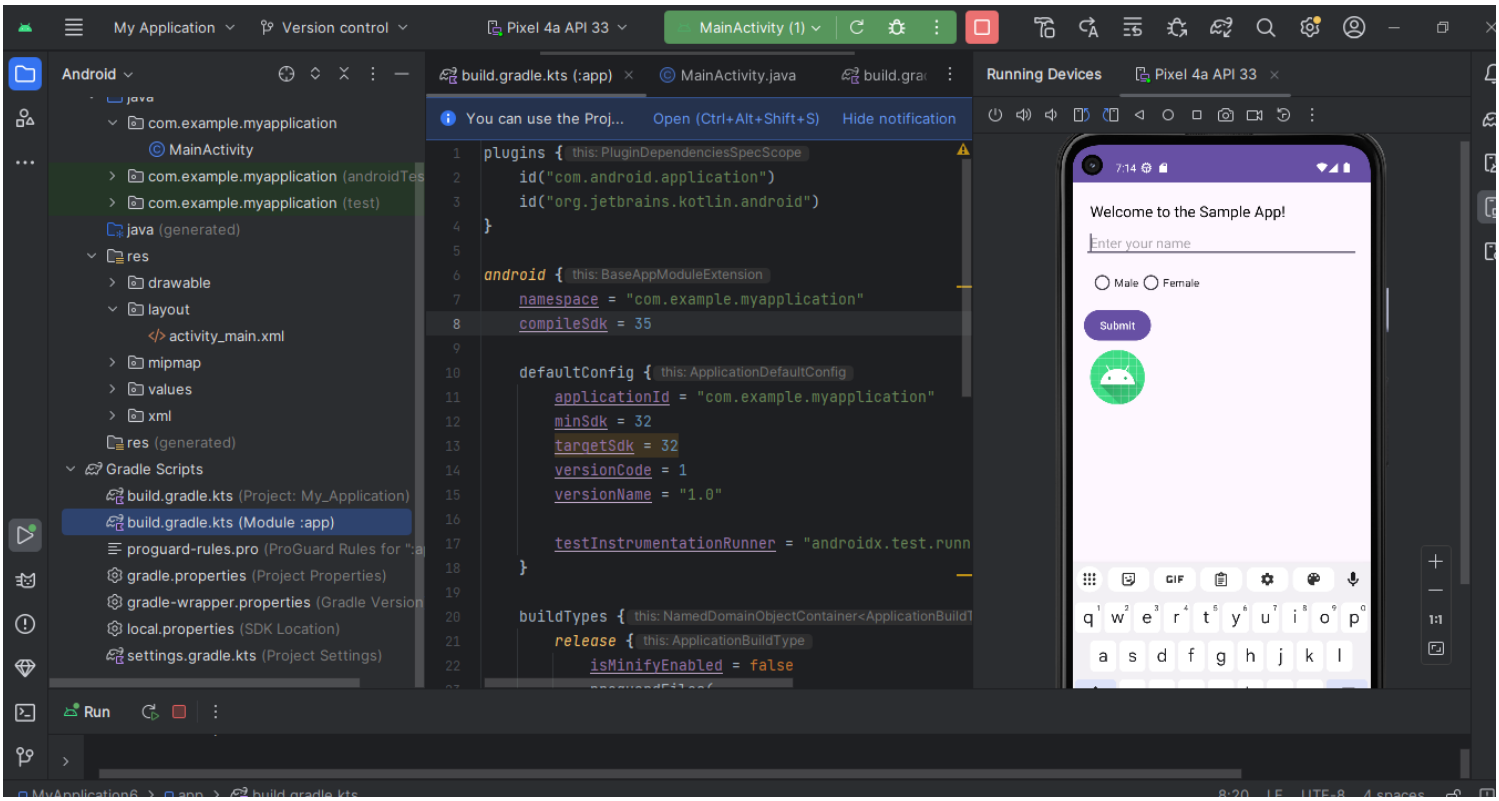


Right clik on activity_main and go to open with notepad..



Paste the xml code in notepad and then save it by clicking clt+s.

Note:- You have to click on synch now for build.gradle.kt update



XML FILE CODE

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

    <!-- Checkbox -->
    <CheckBox
        android:id="@+id/checkbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Check me" />

    <!-- Radio Group with two Radio Buttons -->
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:orientation="vertical"
    android:layout_marginTop="16dp">

    <RadioButton
        android:id="@+id/radioButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 1" />

    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 2" />
</RadioGroup>

<!-- Button -->
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.interactiondemo;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private CheckBox checkBox;
```

```
private RadioGroup radioGroup;

private Button button;


@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);


    // Initialize UI elements

    checkBox = findViewById(R.id.checkbox);

    radioGroup = findViewById(R.id.radioGroup);

    button = findViewById(R.id.button);


    // Set up checkbox listener

    checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {

        if (isChecked) {

            Toast.makeText(MainActivity.this, "Checkbox checked!",
Toast.LENGTH_SHORT).show();

        } else {

            Toast.makeText(MainActivity.this, "Checkbox unchecked!",
Toast.LENGTH_SHORT).show();
```

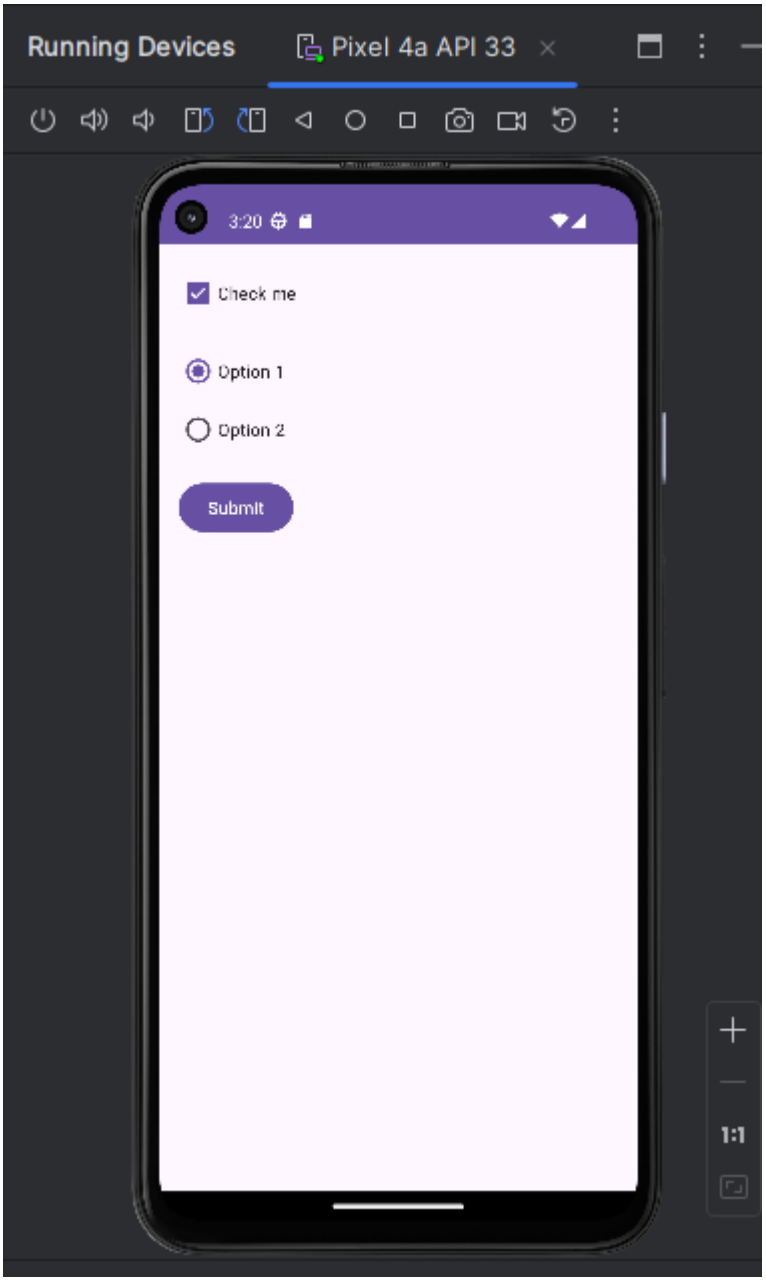
```
        }
    });

    // Set up radio group listener
    radioGroup.setOnCheckedChangeListener((group, checkedId) -> {
        RadioButton selectedRadioButton = findViewById(checkedId);
        Toast.makeText(MainActivity.this, "Selected: " + selectedRadioButton.getText(),
Toast.LENGTH_SHORT).show();
    });

    // Set up button listener
    button.setOnClickListener(v -> {
        boolean isCheckBoxChecked = checkBox.isChecked();
        int selectedRadioId = radioGroup.getCheckedRadioButtonId();
        if (selectedRadioId != -1) {
            RadioButton selectedRadioButton = findViewById(selectedRadioId);
            String message = "Button clicked! Checkbox is " + (isCheckBoxChecked ? "checked" :
"unchecked")
                    + " and selected radio button is " + selectedRadioButton.getText();
            Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();
        } else {
```

```
            Toast.makeText(MainActivity.this, "Please select a radio button!",
    Toast.LENGTH_SHORT).show();

        }

    });

  }

}
```

Explanation

- **Checkbox**: Shows a Toast message whenever checked/unchecked.
- **Radio Buttons**: Displays a Toast with the selected radio button's text.
- **Button**: On click, it checks the checkbox and radio button states, then displays a Toast with their statuses.

OUTPUT:-



Step by Step Explanation of the code

1. Import Statements

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

import android.os.Bundle;

import android.view.View;
```

```
import android.widget.Button;

import android.widget.CheckBox;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.Toast;
```

These imports bring in essential classes for creating an Android app, such as AppCompatActivity (base class for activities), UI elements (Button, CheckBox, RadioButton, RadioGroup), and Toast (for displaying temporary messages).

2. MainActivity Class

```
public class MainActivity extends AppCompatActivity {

    // Declare UI element variables

    private CheckBox checkBox;

    private RadioGroup radioGroup;

    private Button button;
```

- **MainActivity** extends AppCompatActivity, making it an activity in the app.
- checkBox, radioGroup, and button are private variables used to reference the UI elements from the XML layout.

3. onCreate Method

```
@Override

protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);
```

- **onCreate**: This method is called when the activity is first created.
- setContentView(R.layout.activity_main): Sets the XML layout file (activity_main.xml) to define the activity's UI.

4. Initializing UI Elements

```
// Initialize UI elements

checkBox = findViewById(R.id.checkbox);

radioGroup = findViewById(R.id.radioGroup);

button = findViewById(R.id.button);
```

- findViewById is used to link each UI element with its corresponding ID in the XML layout file, enabling interactions with them in Java code.

5. Checkbox Listener

```
checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {

    if (isChecked) {

        Toast.makeText(MainActivity.this, "Checkbox checked!",
Toast.LENGTH_SHORT).show();

    } else {

        Toast.makeText(MainActivity.this, "Checkbox unchecked!",
Toast.LENGTH_SHORT).show();

    }

});
```

- **setOnCheckedChangeListener**: Listens for changes to the checkbox state.
- **Lambda expression**: (buttonView, isChecked) -> {...} represents a simplified listener implementation.
- **isChecked**: A boolean value, true if the checkbox is checked and false otherwise.
- Toast: Displays a short message when the checkbox is checked or unchecked, indicating the checkbox's current state.

6. RadioGroup Listener

```
radioGroup.setOnCheckedChangeListener((group, checkedId) -> {

    RadioButton selectedRadioButton = findViewById(checkedId);

    Toast.makeText(MainActivity.this, "Selected: " + selectedRadioButton.getText(),
Toast.LENGTH_SHORT).show();

});
```

- **setOnCheckedChangeListener**: Monitors changes in the RadioGroup.
- checkedId: The ID of the currently selected RadioButton.
- findViewById(checkedId): Retrieves the selected RadioButton based on checkedId.
- selectedRadioButton.getText(): Gets the text of the selected RadioButton.
- Toast: Displays the text of the selected RadioButton.

7. Button Click Listener

```
button.setOnClickListener(v -> {

    boolean isCheckBoxChecked = checkBox.isChecked();

    int selectedRadioId = radioGroup.getCheckedRadioButtonId();

    if (selectedRadioId != -1) {

        RadioButton selectedRadioButton = findViewById(selectedRadioId);

        String message = "Button clicked! Checkbox is " + (isCheckBoxChecked ? "checked" :
"unchecked")
```

```
                + " and selected radio button is " + selectedRadioButton.getText();

            Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();

        } else {

            Toast.makeText(MainActivity.this, "Please select a radio button!",
Toast.LENGTH_SHORT).show();

        }

    });
```

setOnClickListener: Triggers when the button is clicked.

Checkbox state: isCheckBoxChecked captures the checkbox state (true if checked, false if not).

RadioButton state: getCheckedRadioButtonId retrieves the selected radio button's ID.

- If selectedRadioId is -1, no radio button is selected, so a Toast message requests the user to select one.
- If a radio button is selected, findViewById(selectedRadioId) retrieves it, and getText() fetches its text.

Message display: The final message includes the checkbox's status (checked/unchecked) and the selected radio button's text.

Conclusion

The code effectively demonstrates handling interactions for a checkbox, radio buttons, and a button in Android. The Toast messages provide immediate feedback based on the user's actions, making the app interactive and responsive.

5) create an android application to create image flipper and image gallery. on click on the image display the information about the image.

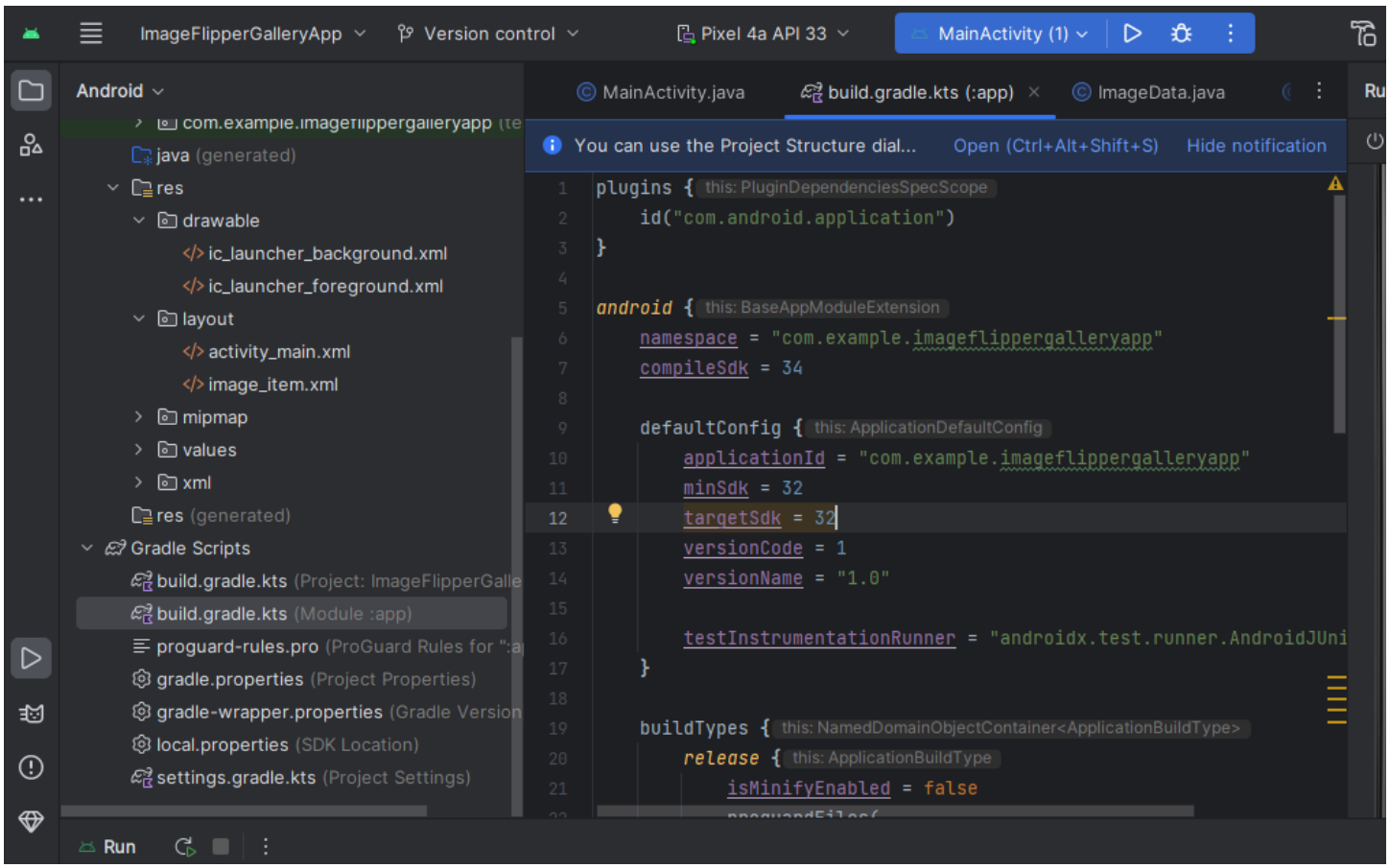
Description :-

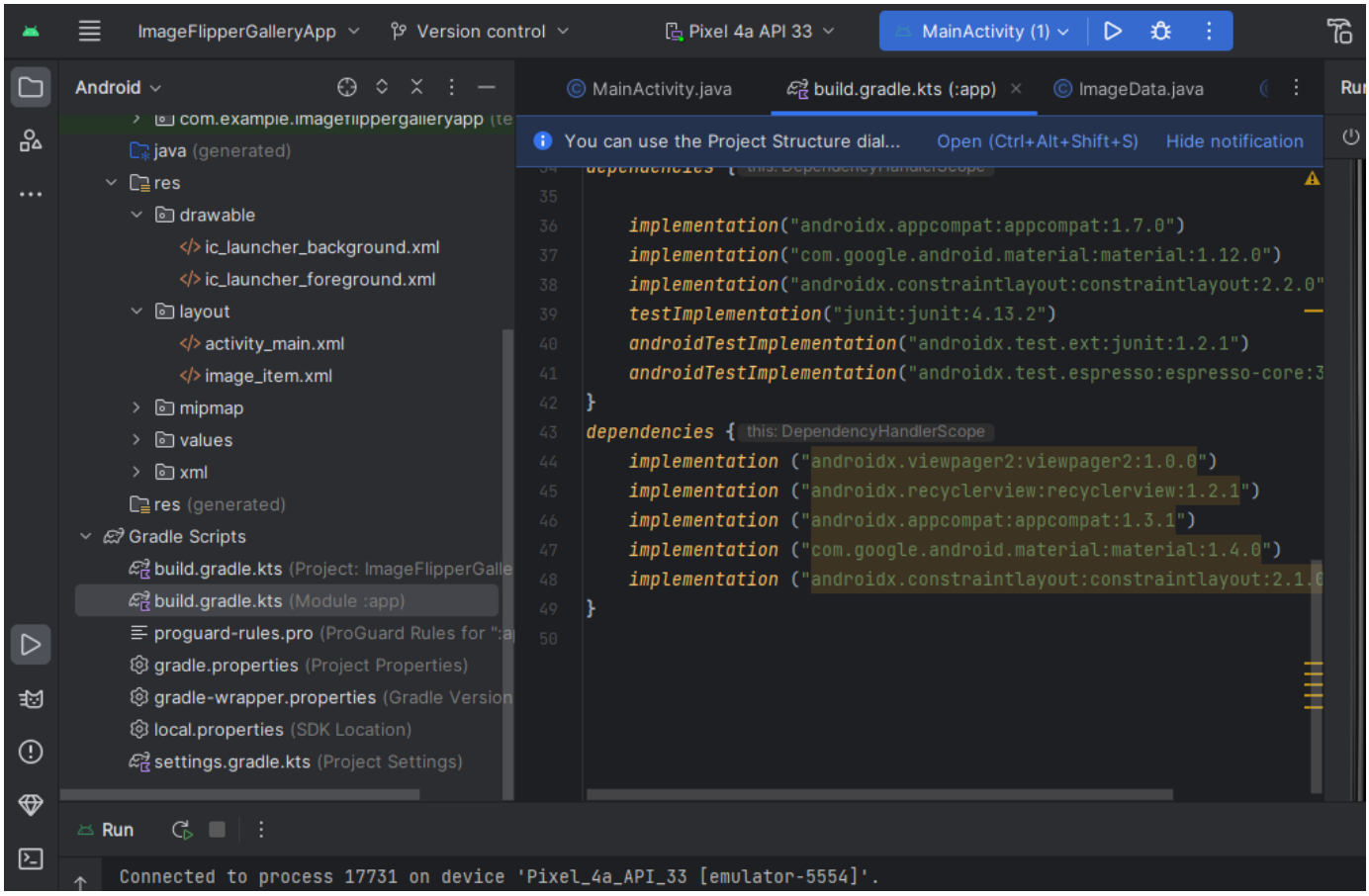
The creation of an Android application that combines an **Image Flipper** (using ViewPager2) and an **Image Gallery** (using RecyclerView). Additionally, when the user clicks on an image, it will display the **information about the image** (like title and description).

Here is the step by step code.

1. How to Setup a project

Dependencies: You'll need the following dependencies in your build.gradle (Module: app) file:



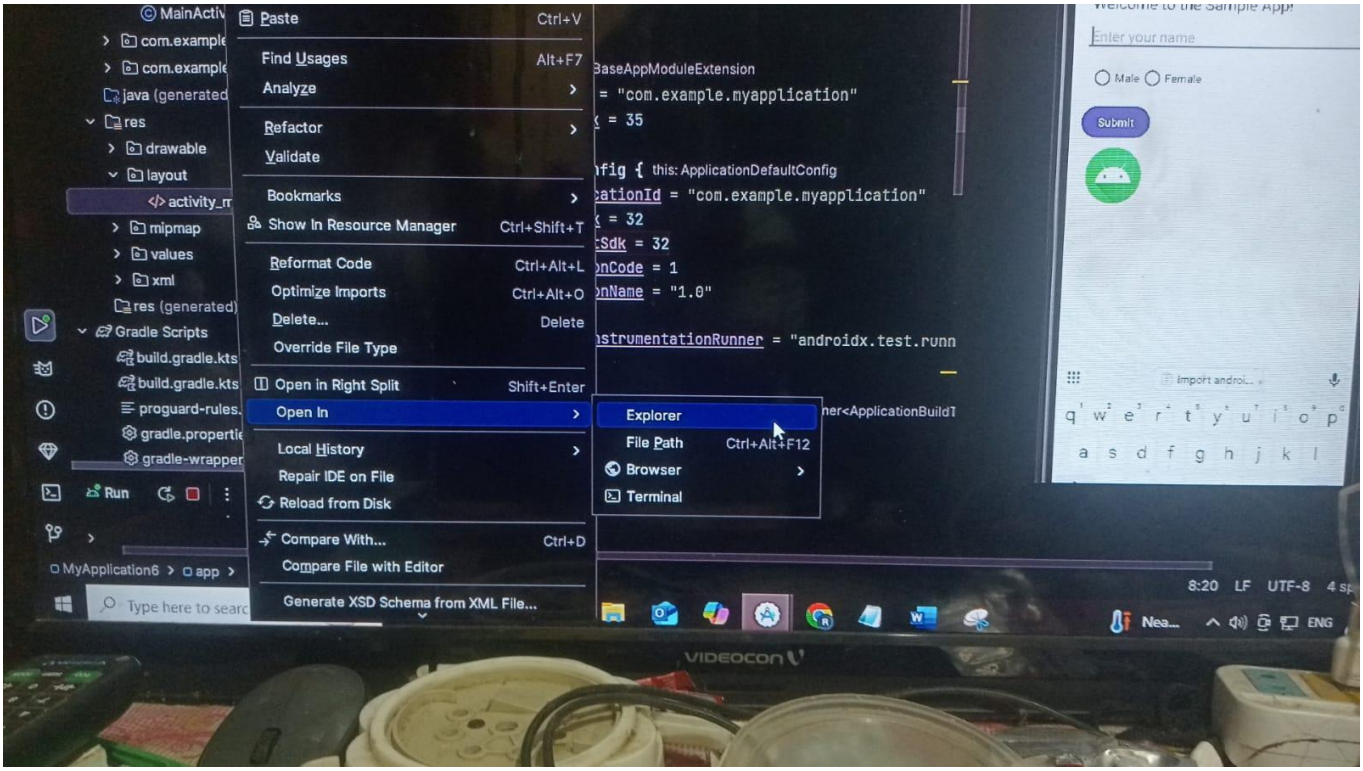


dependencies {
    implementation ("androidx.viewpager2:viewpager2:1.0.0")
    implementation ("androidx.recyclerview:recyclerview:1.2.1")
    implementation ("androidx.appcompat:appcompat:1.3.1")
    implementation ("com.google.android.material:material:1.4.0")
    implementation ("androidx.constraintlayout:constraintlayout:2.1.0")
}

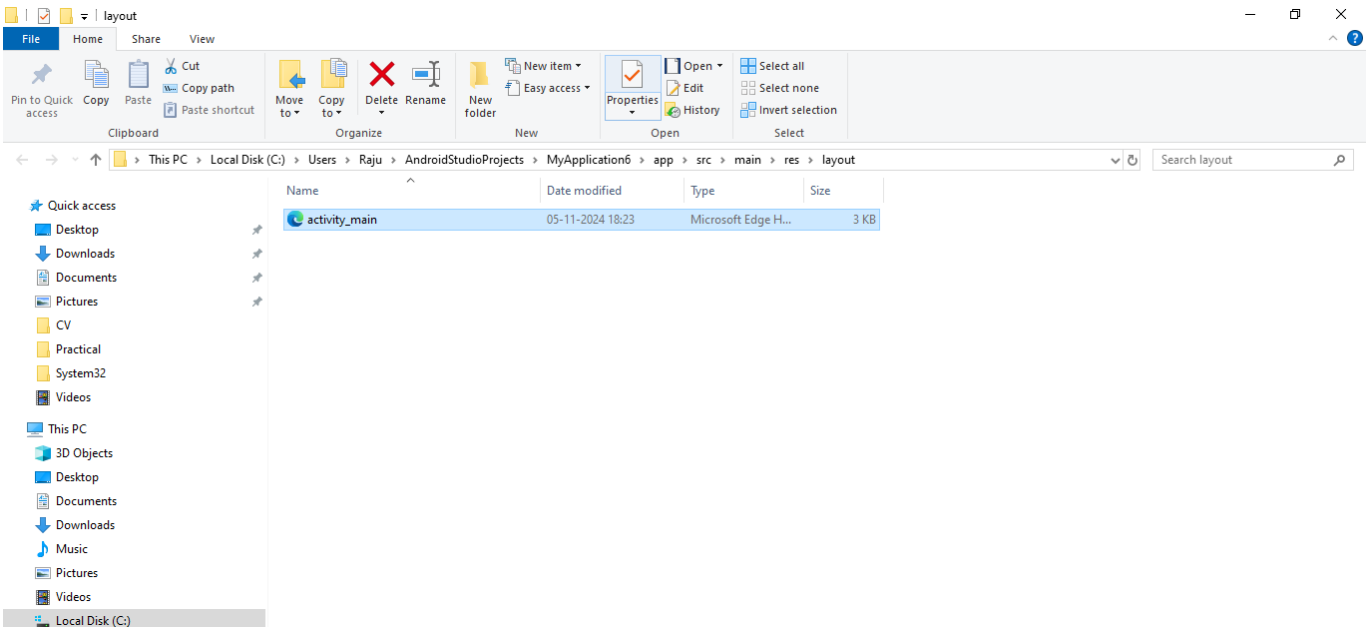
2. Layout Design (activity_main.xml)

This layout will have:

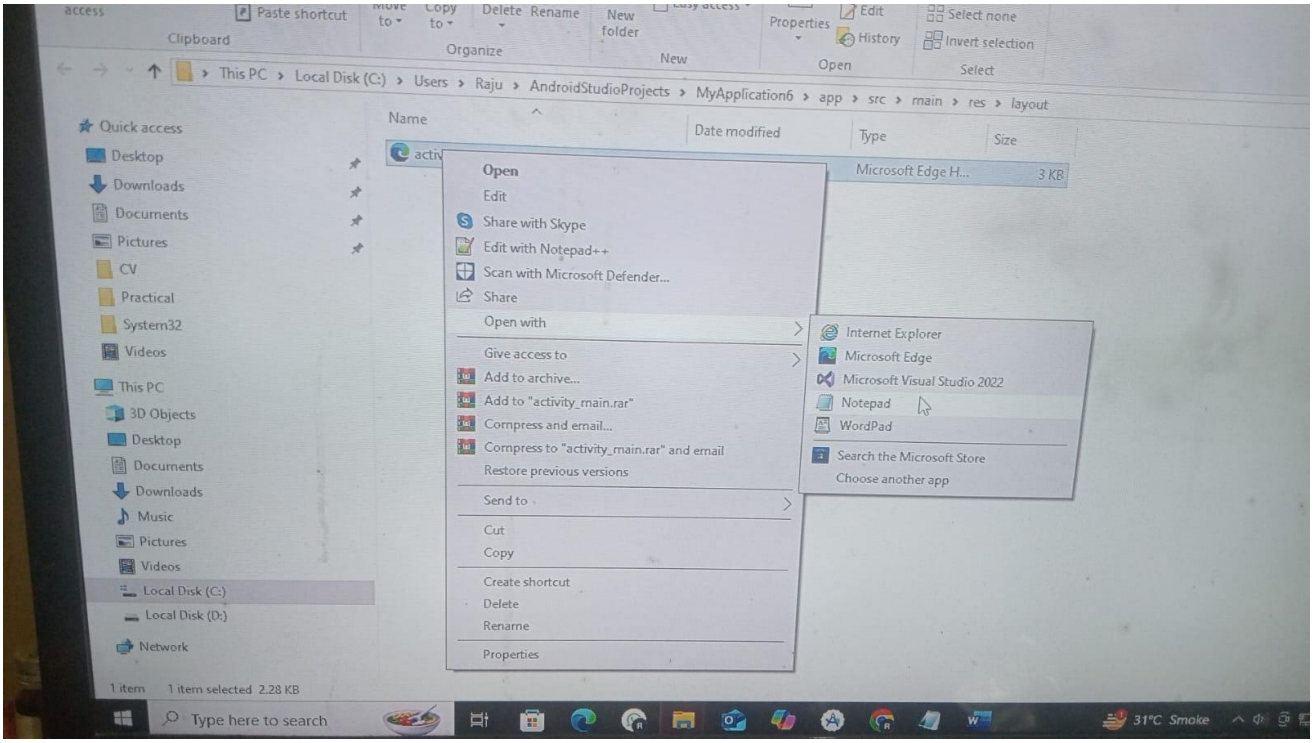
- A ViewPager2 for image flipping.
- A RecyclerView for displaying images in a grid (gallery).
- A TextView to display image information when an image is clicked.



- Right click on explorer you will see this picture….



- Right clik on activity_main and go to open with notepad..

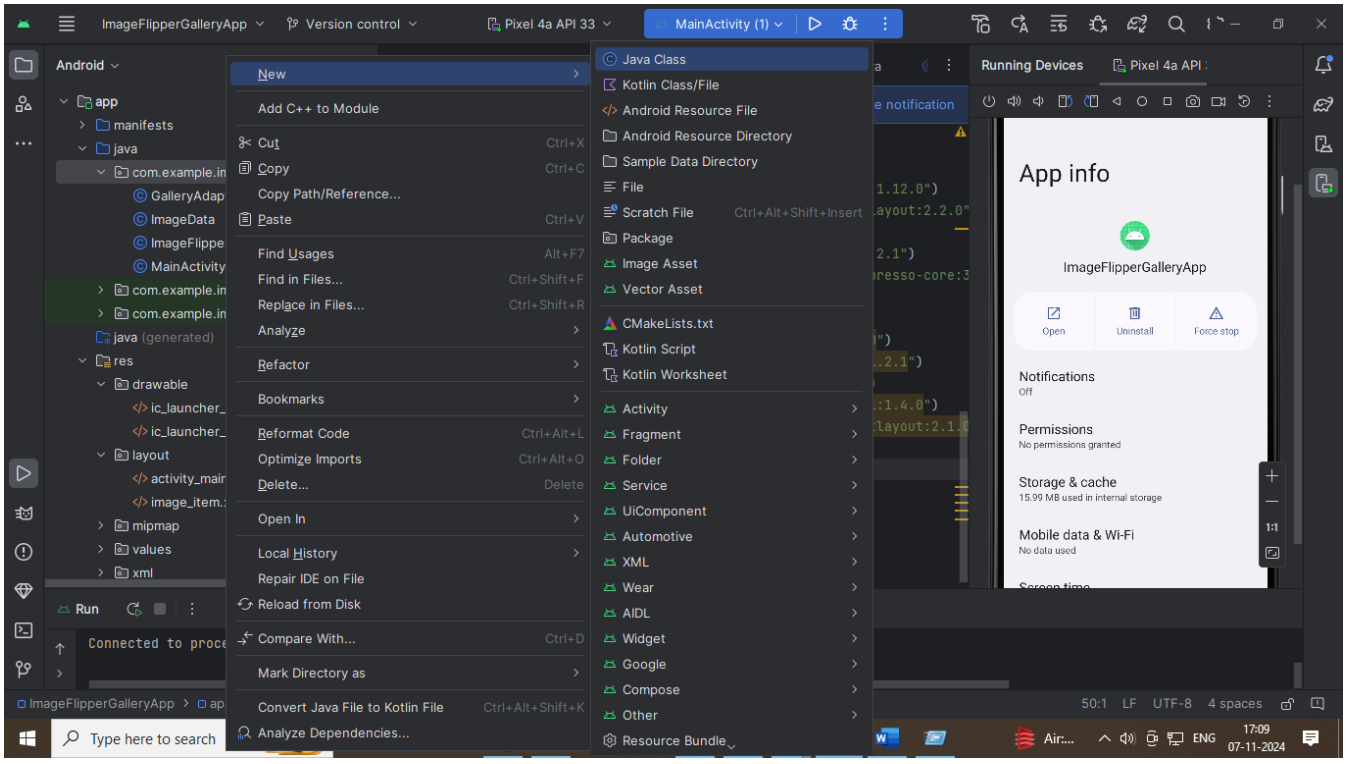



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- ViewPager2 for Image Flipper -->
    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="250dp" />

    <!-- TextView for displaying information about the clicked image -->
    <TextView
        android:id="@+id/imageInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click an image for details"
        android:textSize="16sp"
        android:padding="8dp"
        android:layout_marginTop="16dp"
        android:gravity="center" />
```

```
    <!-- RecyclerView for Image Gallery -->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:visibility="gone" />

    <!-- Button to toggle between gallery and flipper -->
    <Button
        android:id="@+id/btnToggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Gallery"
        android:layout_marginTop="16dp" />
</LinearLayout>
```

3. Create the Class (ImageData.java)

This class will hold the image data, including the image resource, title, and description.

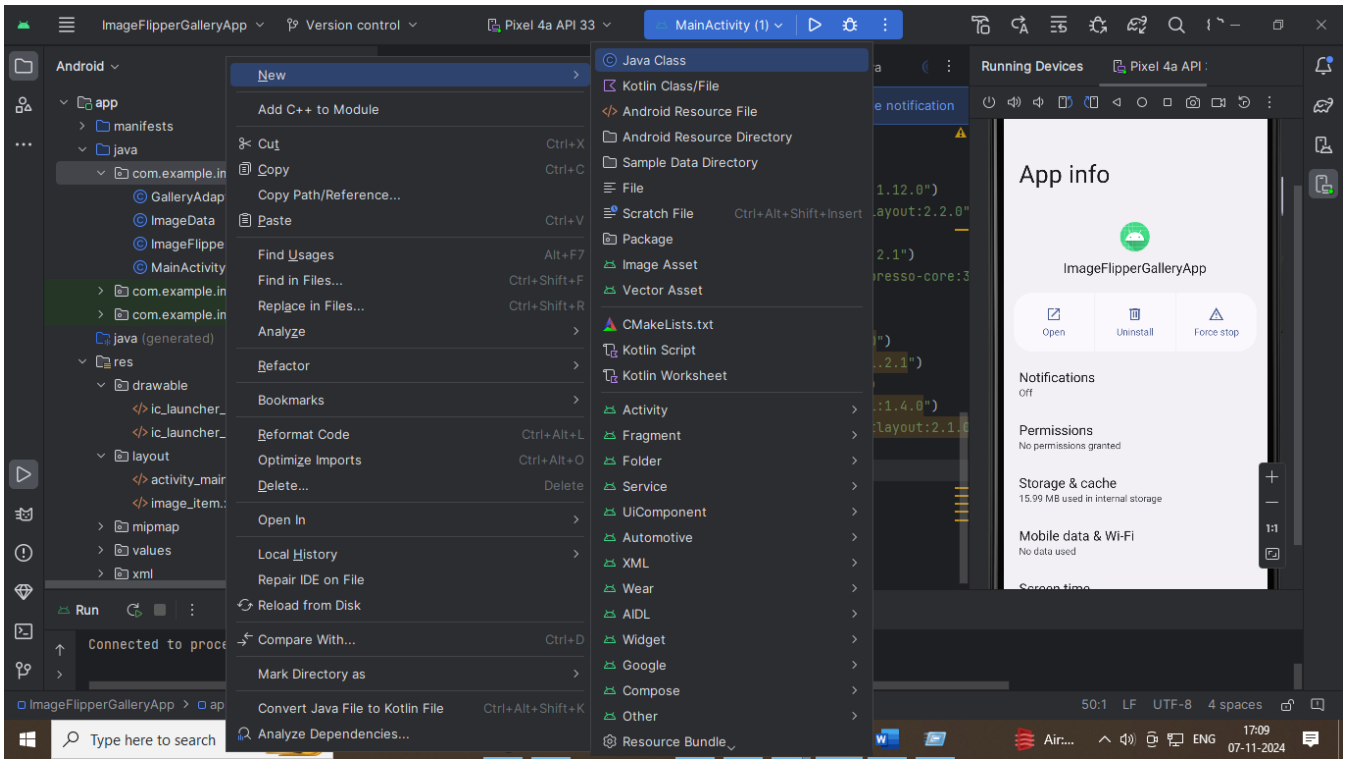


```
public class ImageData {
    private int imageResId;
    private String title;
    private String description;

    // Constructor
    public ImageData(int imageResId, String title, String description) {
        this.imageResId = imageResId;
        this.title = title;
        this.description = description;
    }

    // Getters
    public int getImageResId() {
        return imageResId;
    }

    public String getTitle() {
        return title;
    }
```

```
    public String getDescription() {
        return description;
    }
}
```

4. Create another class (ImageFlipperAdapter.java)

This adapter binds the image data to the ViewPager2 for image flipping.




```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
```

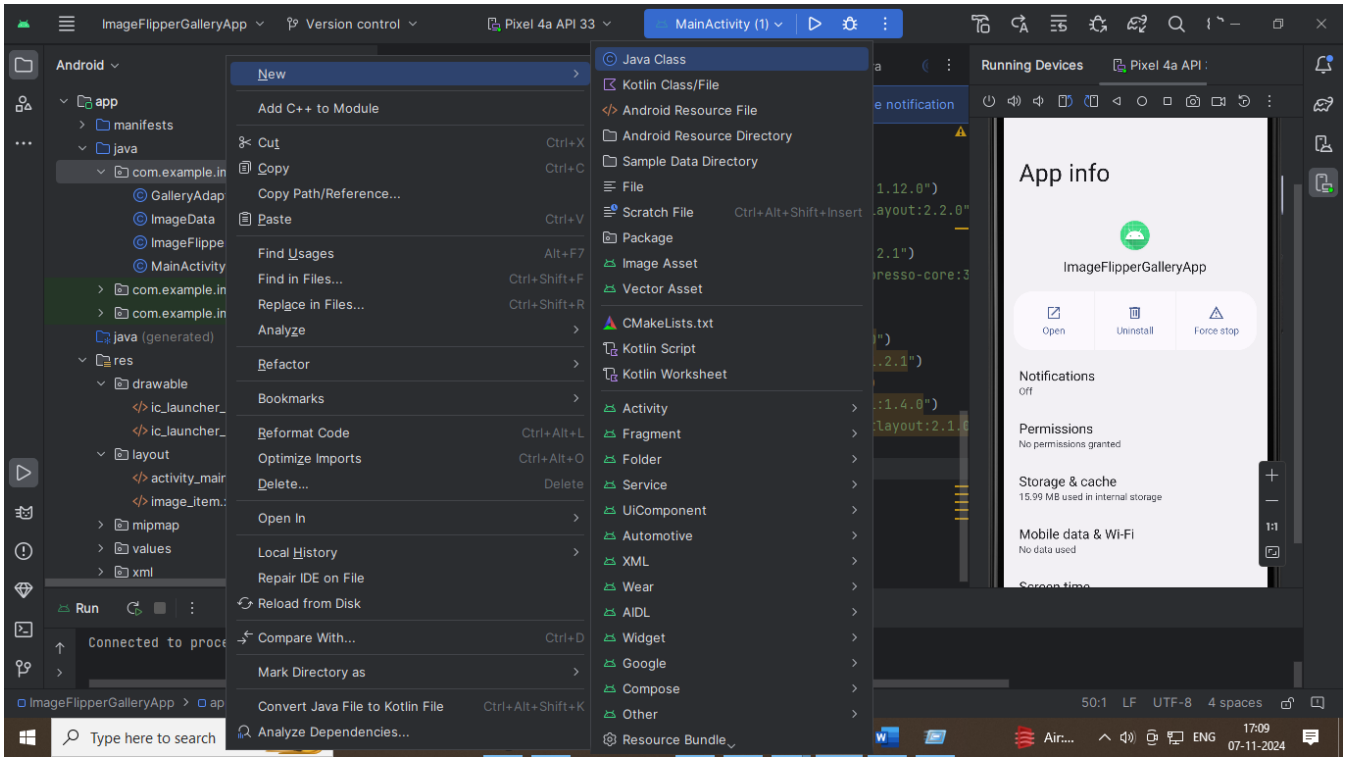
```
import android.view.ViewGroup;

import android.widget.ImageView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;


import java.util.List;


public class ImageFlipperAdapter extends
RecyclerView.Adapter<ImageFlipperAdapter.ImageViewHolder> {

    private Context context;

    private List<ImageData> imageDataList;


    public ImageFlipperAdapter(Context context, List<ImageData> imageDataList) {

        this.context = context;

        this.imageDataList = imageDataList;

    }


    @NonNull

    @Override

    public ImageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
```

```
    View view = LayoutInflater.from(context).inflate(R.layout.image_item, parent, false);
    return new ImageViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull ImageViewHolder holder, int position) {
    ImageData imageData = imageDataList.get(position);
    holder.imageView.setImageResource(imageData.getImageResId());
}

@Override
public int getItemCount() {
    return imageDataList.size();
}

public static class ImageViewHolder extends RecyclerView.ViewHolder {
    ImageView imageView;

    public ImageViewHolder(View itemView) {
        super(itemView);
```

```
        imageView = itemView.findViewById(R.id.imageView);
    }
  }
}
```

5. Create again another (GalleryAdapter.java)

This adapter binds the image data to the RecyclerView for the gallery view.




```
import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;
```

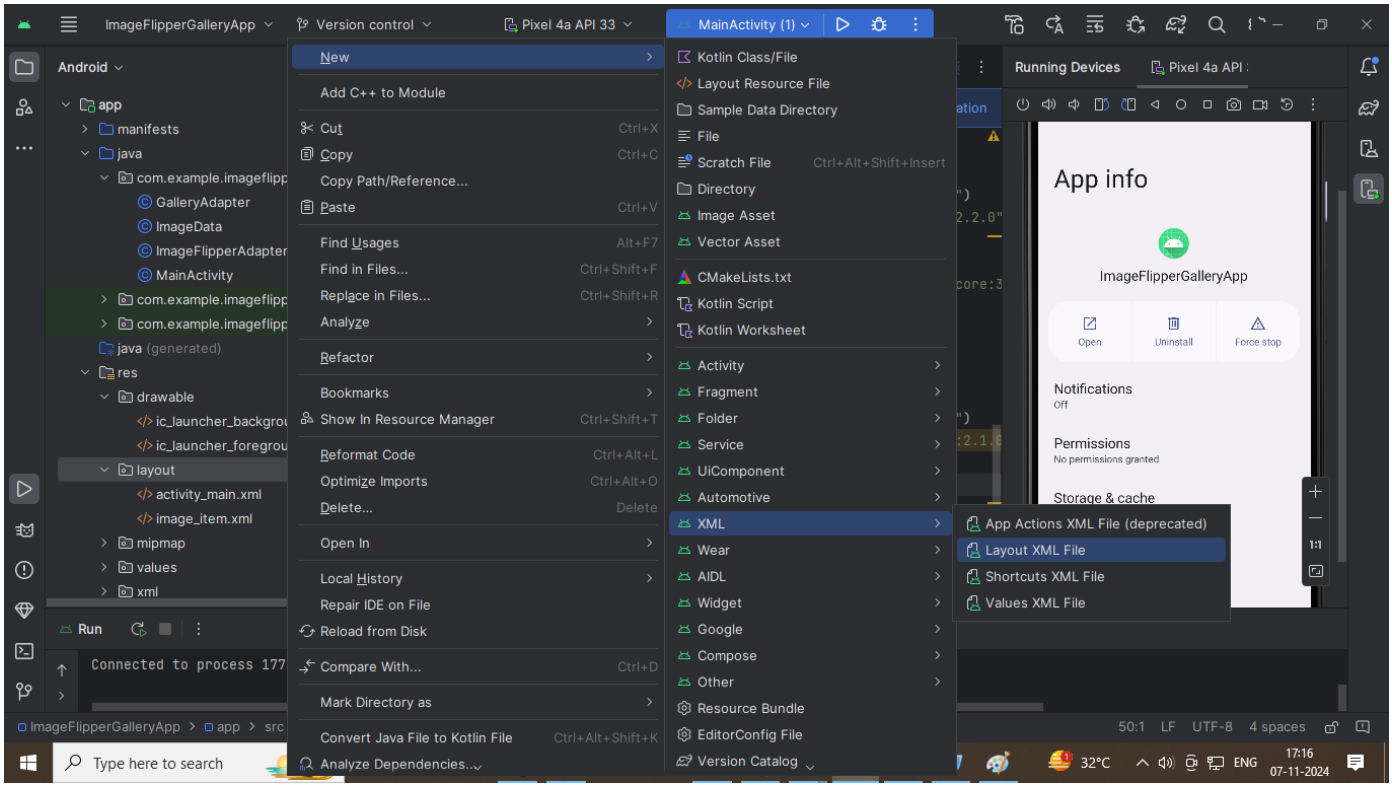
```
import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;



import java.util.List;



public class GalleryAdapter extends RecyclerView.Adapter<GalleryAdapter.GalleryViewHolder>
{



    private Context context;

    private List<ImageData> imageDataList;

    private OnImageClickListener onImageClickListener;



    public GalleryAdapter(Context context, List<ImageData> imageDataList,
OnImageClickListener listener) {

        this.context = context;

        this.imageDataList = imageDataList;

        this.onImageClickListener = listener;

    }
```

```
    @NonNull
    @Override
    public GalleryViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view = LayoutInflater.from(context).inflate(R.layout.image_item, parent, false);
        return new GalleryViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull GalleryViewHolder holder, int position) {
        ImageData imageData = imageDataList.get(position);
        holder.imageView.setImageResource(imageData.getImageResId());
        holder.title.setText(imageData.getTitle());

        holder.itemView.setOnClickListener(v -> onImageClickListener.onImageClick(imageData));
    }

    @Override
    public int getItemCount() {
        return imageDataList.size();
```

```
    }

    public interface OnImageClickListener {
        void onImageClick(ImageData imageData);
    }

    public static class GalleryViewHolder extends RecyclerView.ViewHolder {
        ImageView imageView;
        TextView title;

        public GalleryViewHolder(View itemView) {
            super(itemView);
            imageView = itemView.findViewById(R.id.imageView);
            title = itemView.findViewById(R.id.imageTitle);
        }
    }
}
```

6. Define image_item.xml (Shared Layout for Flipper and Gallery)

To create layout.xml you have to create a new image_item xml file(any file name you can give it)



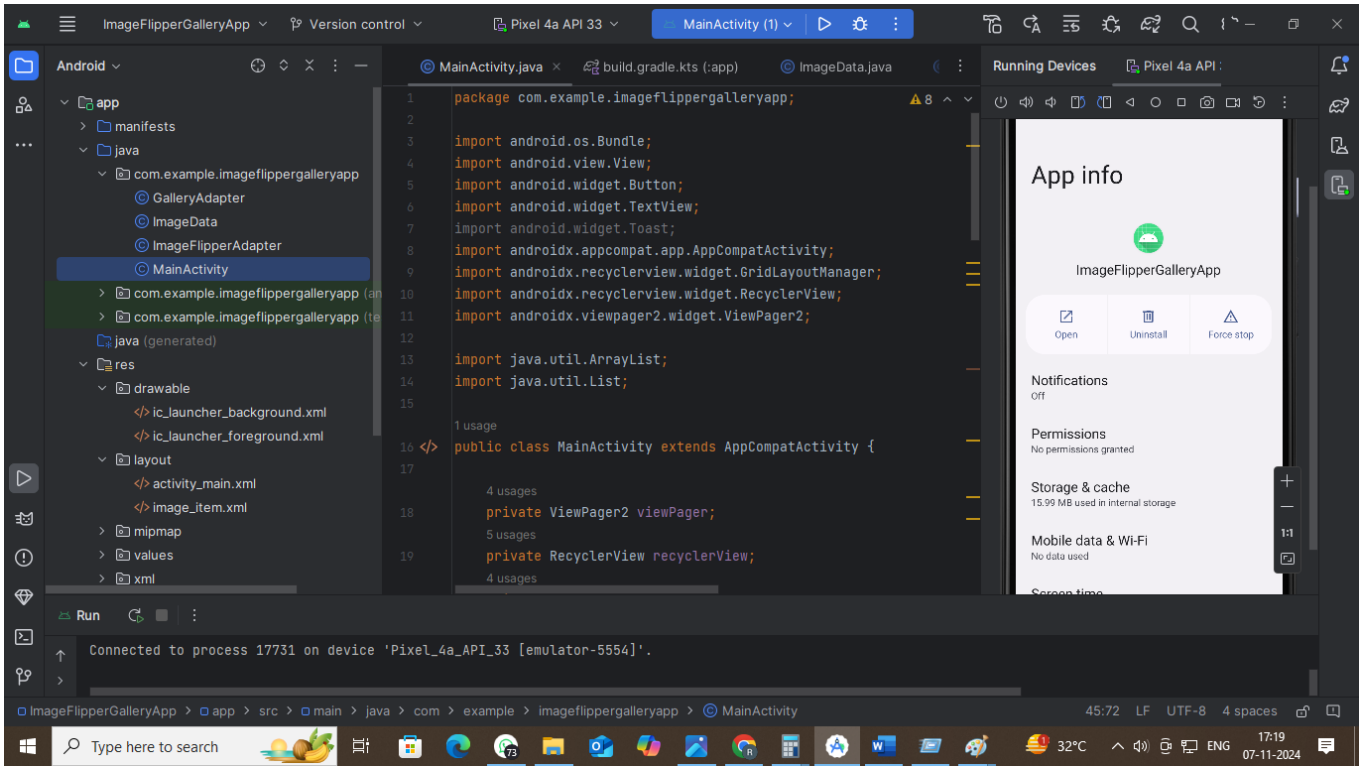

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="8dp"
    android:gravity="center">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="100dp"
```

```
        android:layout_height="100dp"
        android:src="@drawable/ic_launcher_background" />

    <TextView
        android:id="@+id/imageTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Image Title"
        android:textSize="14sp"
        android:layout_marginTop="8dp" />
</LinearLayout>
```

7. MainActivity.java

In this file, we will:

- Set up both the (image flipper) and (image gallery).
- Handle the toggling between the flipper and gallery.
- Display image information when an image is clicked.




```
import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.GridLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import androidx.viewpager2.widget.ViewPager2;



import java.util.ArrayList;
```

```
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private ViewPager2 viewPager;
    private RecyclerView recyclerView;
    private Button btnToggle;
    private TextView imageInfo;
    private List<ImageData> imageList;
    private ImageFlipperAdapter imageFlipperAdapter;
    private GalleryAdapter galleryAdapter;
    private boolean isGalleryVisible = false;

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

        // Initialize UI components
        viewPager = findViewById(R.id.viewPager);
```

```
    recyclerView = findViewById(R.id.recyclerView);
    btnToggle = findViewById(R.id.btnToggle);
    imageInfo = findViewById(R.id.imageInfo);

    // Sample data for the image flipper and gallery
    imageList = new ArrayList<>();
    imageList.add(new ImageData(R.drawable.image1, "Image 1", "Description of Image 1"));
    imageList.add(new ImageData(R.drawable.image2, "Image 2", "Description of Image 2"));
    imageList.add(new ImageData(R.drawable.image3, "Image 3", "Description of Image 3"));

    // Set up ViewPager2 for Image Flipper
    imageFlipperAdapter = new ImageFlipperAdapter(this, imageList);
    viewPager.setAdapter(imageFlipperAdapter);

    // Set up RecyclerView for Image Gallery
    galleryAdapter = new GalleryAdapter(this, imageList, imageData -> {
        // Display image information when clicked
        imageInfo.setText("Title: " + imageData.getTitle() + "\nDescription: " +
imageData.getDescription());
    });
```

```
        recyclerView.setLayoutManager(new GridLayoutManager(this, 3));
        recyclerView.setAdapter(galleryAdapter);

        // Button click listener to toggle between gallery and flipper
        btnToggle.setOnClickListener(v -> {
            if (isGalleryVisible) {
                viewPager.setVisibility(View.VISIBLE);
                recyclerView.setVisibility(View.GONE);
                btnToggle.setText("Show Gallery");
            } else {
                viewPager.setVisibility(View.GONE);
                recyclerView.setVisibility(View.VISIBLE);
                btnToggle.setText("Show Flipper");
            }
            isGalleryVisible = !isGalleryVisible;
        });
    }
}
```

Conclusion:-

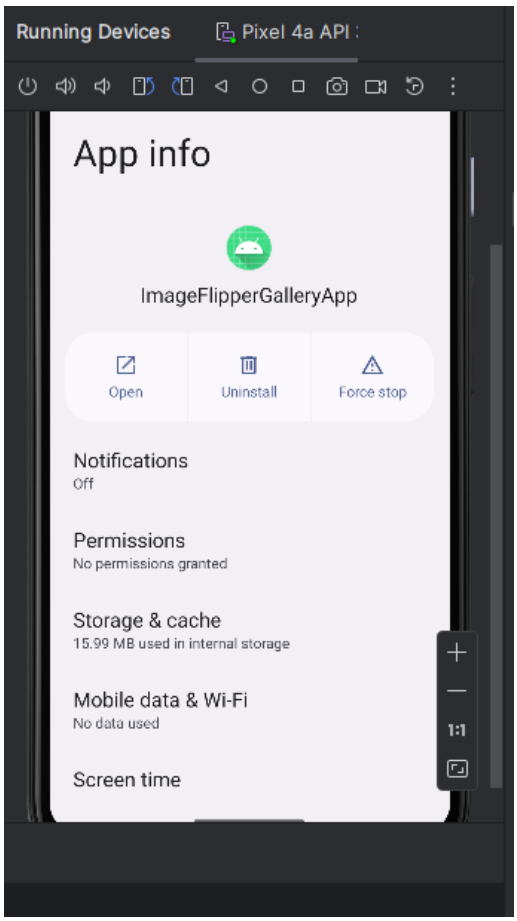
Once you have everything set up:

- When the app starts, you'll see the ViewPager2 displaying images.
- There's a button below that toggles between the image flipper and gallery view.
- If you click on an image in the gallery, it will show the title and description of that image.

Conclusion

This app shows how you can combine an **Image Flipper** (ViewPager2) with an **Image Gallery** (RecyclerView) in one activity. When a user clicks on an image in the gallery, it displays information (like title and description) about the image.

OUTPUT:-

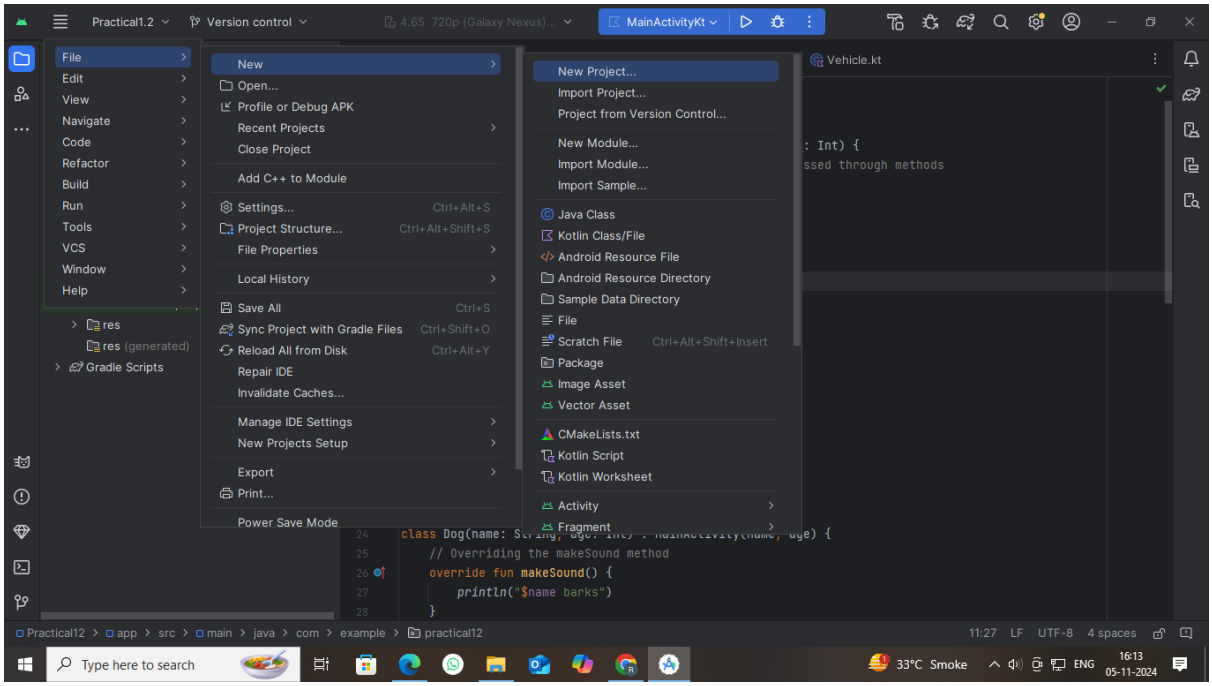


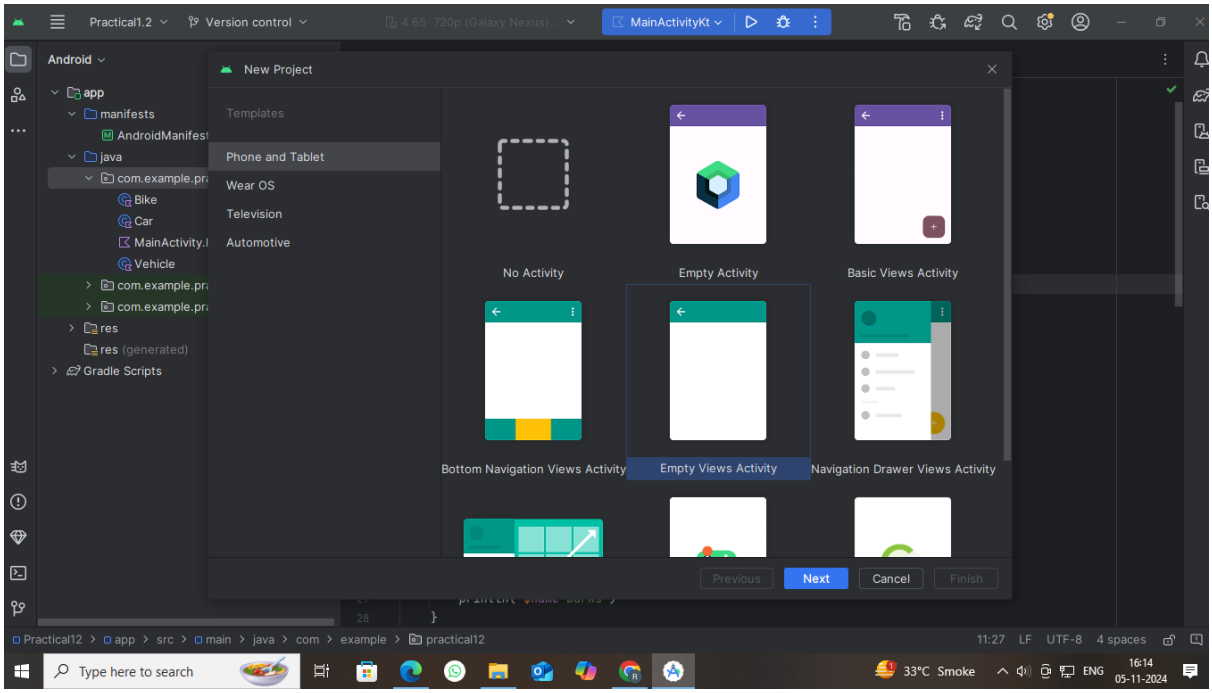
6) create an application to use gridview for shopping cart application

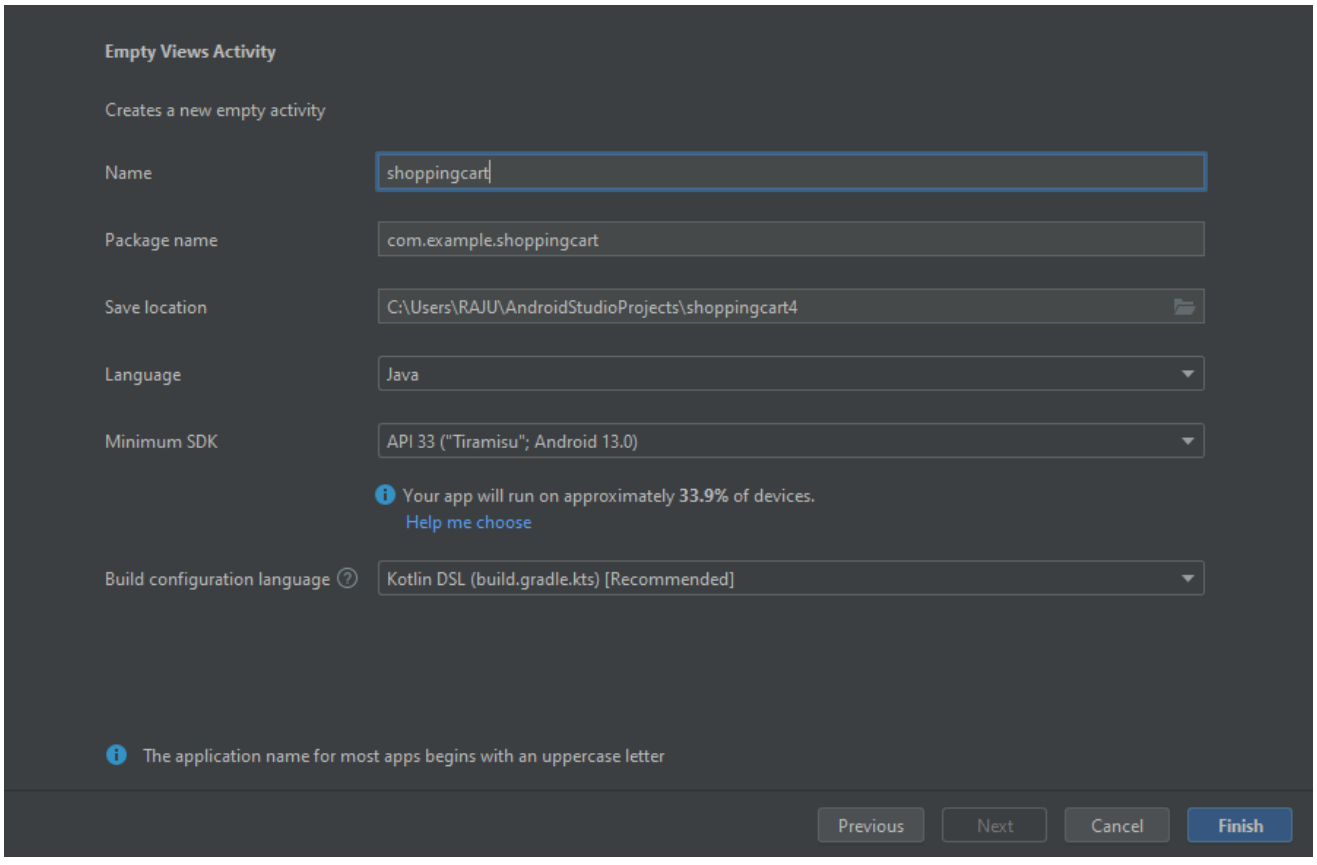
To create a **Shopping Cart Android Application** using a GridView, I will guide you through the process step by step. This will include creating a basic layout for displaying products, adding items to a cart, and handling the necessary code for functionality.

Steps:

1. **Create the Project**: Start by creating a new Android project in Android Studio. Use the default "Empty Activity" template.
2. **Create Product Model**: This class will hold data for each product, like the product name and the image.
3. **Create Custom Adapter**: A custom adapter will be used to display the products in the GridView.
4. **Create Layouts**: You will have two layouts: one for the main activity and another for the item in the grid.
5. **Handle User Interaction**: When a product is clicked, it will be added to a shopping cart, and we can display the cart in another activity.



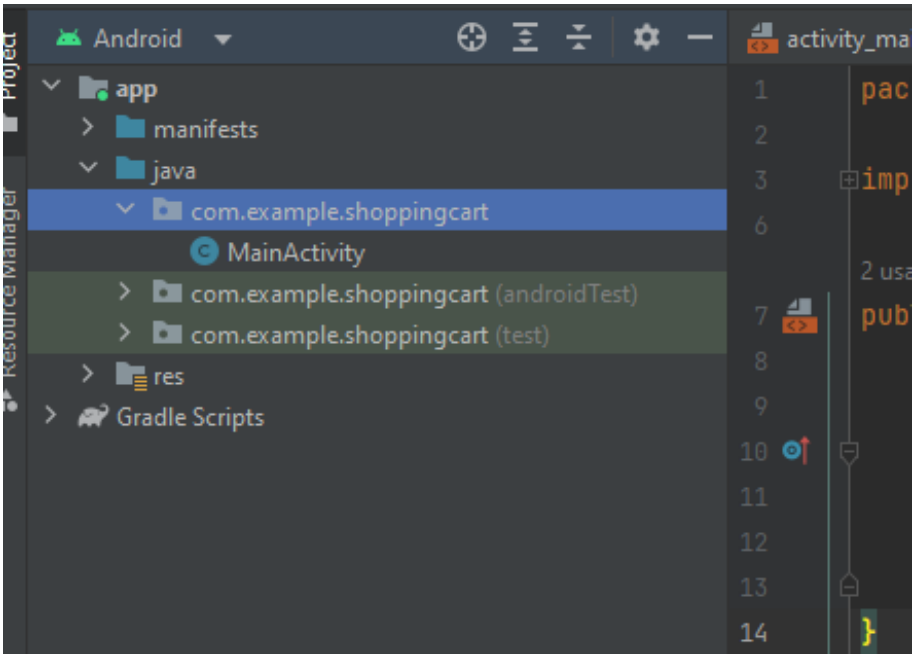

Practical1.2
Version control
MainActivityKt
Android
New Project
Templates
Phone and Tablet
Wear OS
Television
Automotive
app
manifests
AndroidManifest
java
com.example.pra
Bike
Car
MainActivity.
Vehicle
com.example.pra
com.example.pra
res
res (generated)
Gradle Scripts
No Activity
Empty Activity
Basic Views Activity
Bottom Navigation Views Activity
Empty Views Activity
Navigation Drawer Views Activity
Previous
Next
Cancel
Finish
Practical12 > app > src > main > java > com > example > practical12
11:27
LF
UTF-8
4 spaces
Type here to search
33°C Smoke
ENG
16:14
05-11-2024


Empty Views Activity
Creates a new empty activity
Name
shoppingcart
Package name
com.example.shoppingcart
Save location
C:\Users\RAJU\AndroidStudioProjects\shoppingcart4
Language
Java
Minimum SDK
API 33 ("Tiramisu"; Android 13.0)
Your app will run on approximately 33.9% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
The application name for most apps begins with an uppercase letter
Previous
Next
Cancel
Finish

1. Create a Product Model Class

In your project, create a Product class to hold the data for each product.

Step:- To create a class of Product, right click on com.example.shoppingcart and then give the class name as Product.java




```
// Product.java

package com.example.shoppingcart;

public class Product {
    private String name;
    private int imageResourceId;

    public Product(String name, int imageResourceId) {
        this.name = name;
        this.imageResourceId = imageResourceId;
    }

    public String getName() {
        return name;
    }
```

```
    public int getImageResourceId() {
        return imageResourceId;
    }
}
```

2. Create the GridView Adapter

Now create a custom adapter to bind the data to the GridView.

Step:- To create a class of ProductAdapter, right click on com.example.shoppingcart and then give the class name as ProductAdapter.java

```
// ProductAdapter.java

package com.example.shoppingcart;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;

public class ProductAdapter extends BaseAdapter {

    private Context context;
    private List<Product> products;

    public ProductAdapter(Context context, List<Product> products) {
        this.context = context;
        this.products = products;
    }

    @Override
    public int getCount() {
        return products.size();
```

```
    }

    @Override
    public Object getItem(int position) {
        return products.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.grid_item, null);
        }

        TextView productName = convertView.findViewById(R.id.productName);
        ImageView productImage = convertView.findViewById(R.id.productImage);

        Product product = products.get(position);
        productName.setText(product.getName());
        productImage.setImageResource(product.getImageResourceId());

        return convertView;
    }
}
```

3. Create the Layout for Each Grid Item

In res/layout, create a new XML file grid_item.xml to represent each product in the grid.

```
<!-- grid_item.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="8dp"
```

```
    android:gravity="center">

    <ImageView
        android:id="@+id/productImage"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginBottom="8dp"
        android:src="@drawable/ic_launcher_foreground"/>

    <TextView
        android:id="@+id/productName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Product Name"
        android:gravity="center"
        android:textSize="16sp" />
</LinearLayout>
```

4. Create the Main Activity Layout

Now, create the main layout for MainActivity where the GridView will be shown.

```
<!-- activity_main.xml -->

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

    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:numColumns="2"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:columnWidth="120dp"
        android:stretchMode="columnWidth" />
</LinearLayout>
```

5. MainActivity.java

In MainActivity.java, initialize the GridView and add some sample products. Also, handle clicks on items to add them to the cart.

```
// MainActivity.java

package com.example.shoppingcart;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.GridView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private GridView gridView;
    private ArrayList<Product> products;

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

        gridView = findViewById(R.id.gridView);

        // Create some sample products
        products = new ArrayList<>();
        products.add(new Product("Apple", R.drawable.apple));
        products.add(new Product("Banana", R.drawable.banana));
        products.add(new Product("Grapes", R.drawable.grapes));
        products.add(new Product("Orange", R.drawable.orange));

        // Set the GridView adapter
        ProductAdapter adapter = new ProductAdapter(this, products);
        gridView.setAdapter(adapter);
```

```
        // Item click listener
        gridView.setOnItemClickListener((parent, view, position, id) -> {
            Product product = products.get(position);
            // Add to shopping cart (for demonstration, we are showing a toast)
            addToCart(product);
        });
    }

    private void addToCart(Product product) {
        // Display a toast message
        Toast.makeText(this, product.getName() + " added to cart",
Toast.LENGTH_SHORT).show();

        // Pass the product data to the ShoppingCartActivity
        Intent intent = new Intent(MainActivity.this, ShoppingCartActivity.class);
        intent.putExtra("productName", product.getName());
        startActivity(intent);
    }
}
```

6. ShoppingCartActivity.java

Create another activity to display the shopping cart. This will show the list of items added to the cart.

```
// ShoppingCartActivity.java

package com.example.shoppingcart;

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

public class ShoppingCartActivity extends AppCompatActivity {

    private TextView cartTextView;

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

```
        cartTextView = findViewById(R.id.cartTextView);

        // Get the product name passed via intent
        String productName = getIntent().getStringExtra("productName");

        // Display the selected product name
        cartTextView.setText("Product added: " + productName);
    }
}
```

7. Layout for ShoppingCartActivity

Create the layout for ShoppingCartActivity in res/layout/activity_shopping_cart.xml:

```
<!-- activity_shopping_cart.xml -->

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

    <TextView
        android:id="@+id/cartTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Shopping Cart"
        android:textSize="18sp" />
</LinearLayout>
```

8. Update AndroidManifest.xml

Finally, ensure your AndroidManifest.xml is set up correctly to include the ShoppingCartActivity.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shoppingcart">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Shopping Cart"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">

        <!-- MainActivity with intent filter: Specify android:exported -->
        <activity
            android:name=".MainActivity"
            android:exported="true"> <!-- Add android:exported="true" -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- ShoppingCartActivity without intent filter: Specify android:exported -->
        <activity
            android:name=".ShoppingCartActivity"
            android:exported="false"> <!-- Add android:exported="false" -->
        </activity>

    </application>

</manifest>
```

9. Final Output

- When the app starts, the GridView will display a set of products (Apple, Banana, Grapes, Orange).
- When you click on any product, a toast will appear saying that the product is added to the cart, and you will be redirected to ShoppingCartActivity where the product name is displayed.

7) create an application to demonstrate shared preferences using android studio

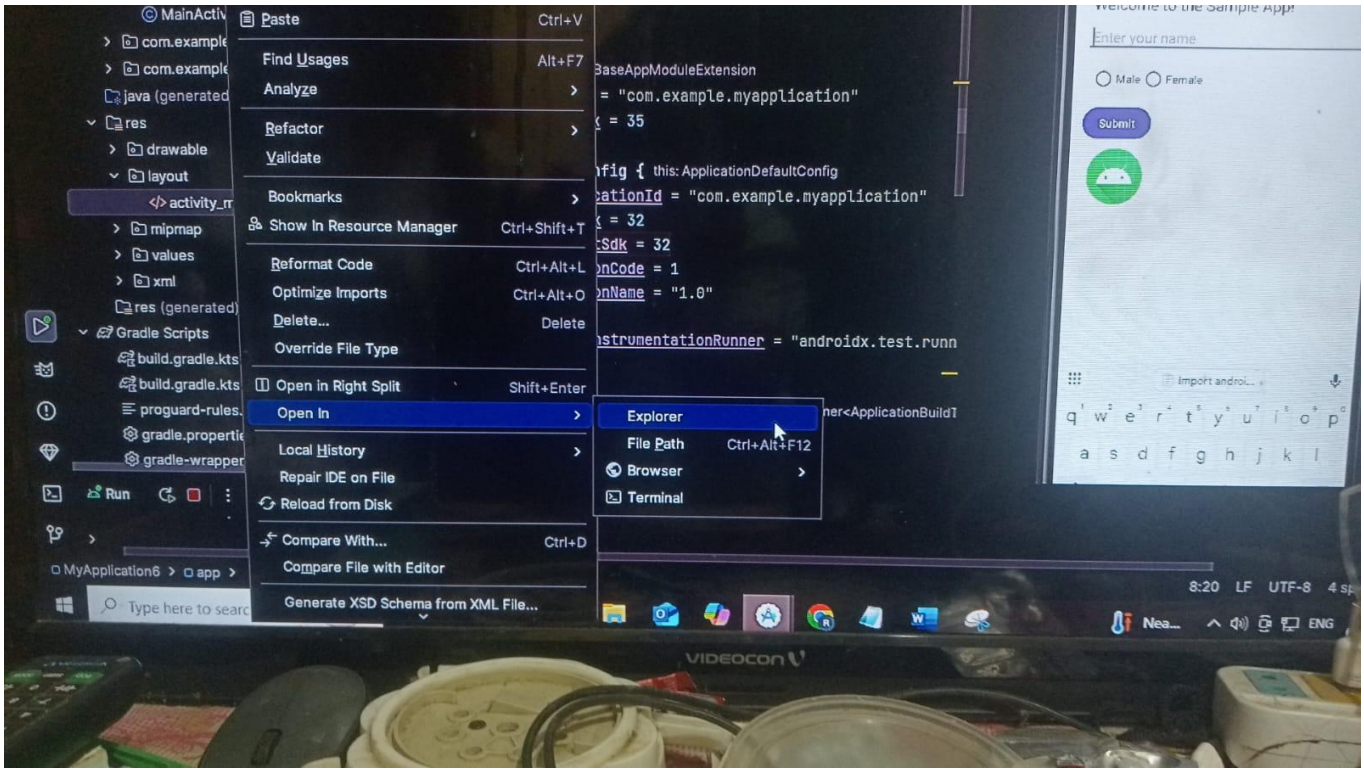
Steps to Create the App:

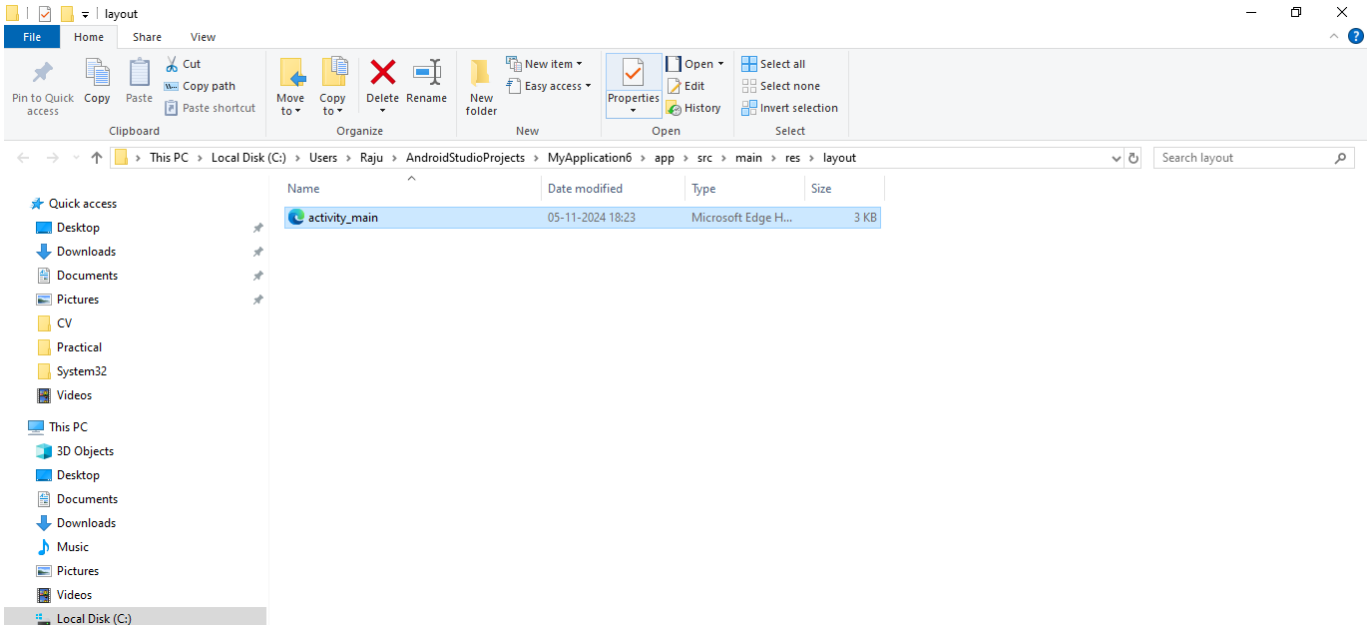
Step 1: Create a New Project

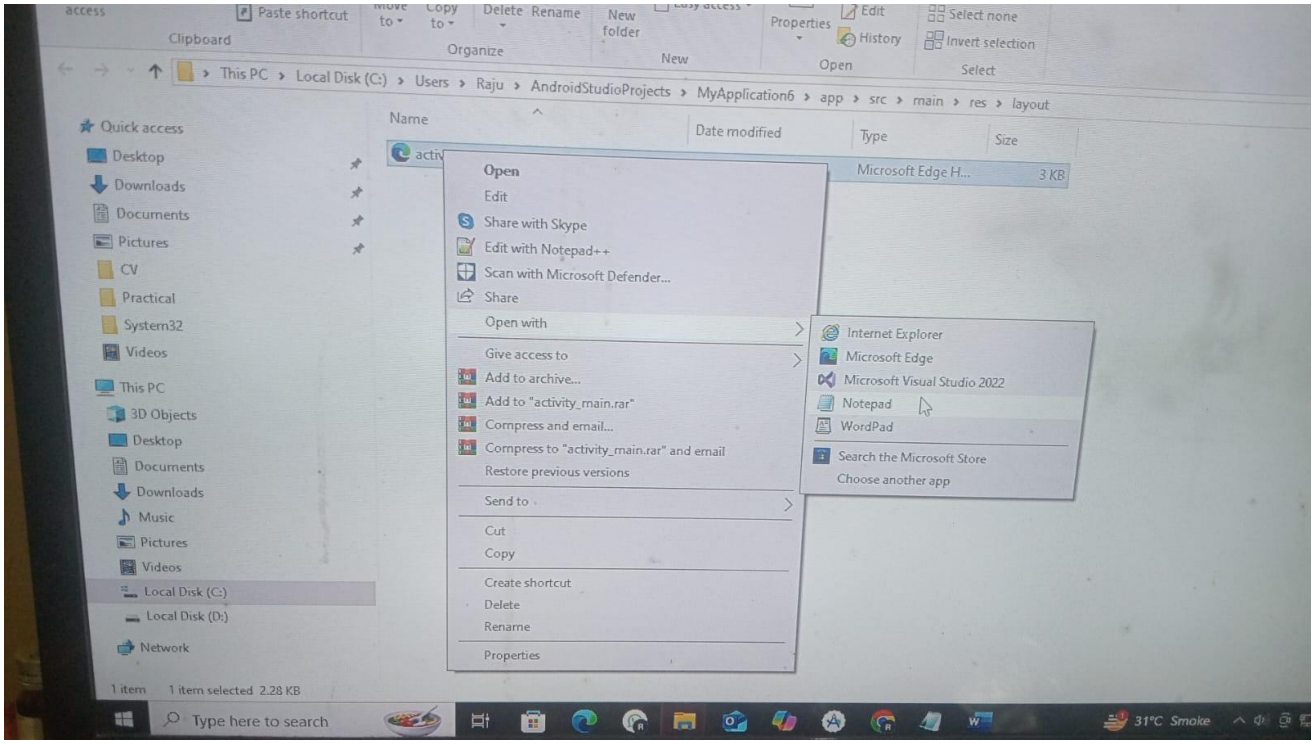
1. Open Android Studio.
2. Click on **File** > **New** > **New Project**.
3. Select **Empty Activity**.
4. Name your application (e.g., **SharedPreferencesDemo**).
5. Set the language to **Java** or **Kotlin** depending on your preference. In this example, I'll use **Java**.

Step 2: Design the User Interface (UI)

In the **res/layout/activity_main.xml** file, create the UI with an EditText to enter data, a Button to save data, and a TextView to display saved data.







CODE:-

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    android:gravity="center">

    <!-- EditText for input -->
    <EditText
        android:id="@+id/editTextData"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        android:inputType="text"/>

    <!-- Button to save data -->
    <Button
        android:id="@+id/buttonSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Save"/>

    <!-- TextView to show saved data -->
    <TextView
        android:id="@+id/textViewSavedData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Saved Data will appear here"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

Step 3: Implement SharedPreferences in MainActivity

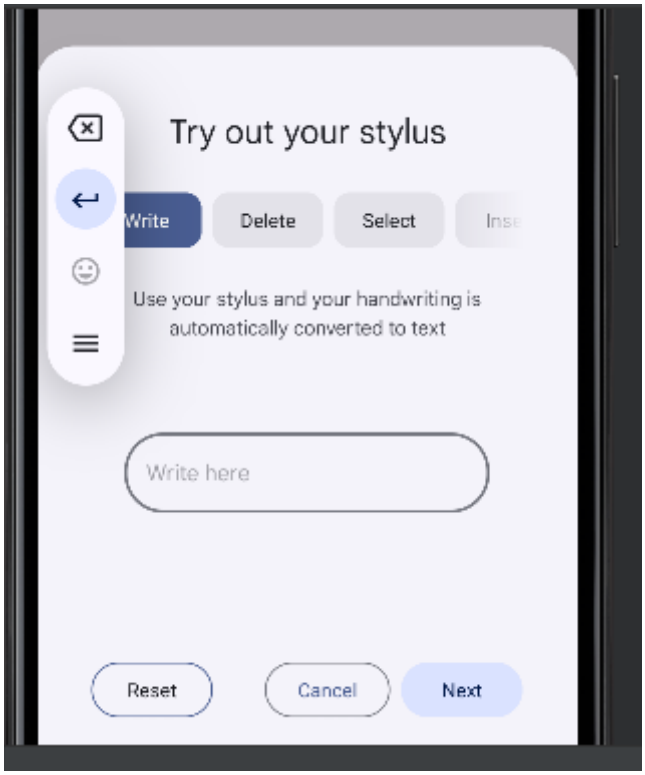
In the MainActivity.java file, use SharedPreferences to save and retrieve data when the user presses the "Save" button.

```
package com.example.demo;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText editTextData;
    private Button buttonSave;
    private TextView textViewSavedData;

    // SharedPreferences name
    private static final String PREFERENCES_NAME = "MyPreferences";
    private static final String KEY_NAME = "name";

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

        // Initialize UI components
        editTextData = findViewById(R.id.editTextData);
        buttonSave = findViewById(R.id.buttonSave);
        textViewSavedData = findViewById(R.id.textViewSavedData);

        // Retrieve saved data from SharedPreferences
        SharedPreferences sharedPreferences = getSharedPreferences(PREFERENCES_NAME,
    MODE_PRIVATE);
        String savedName = sharedPreferences.getString(KEY_NAME, "No name saved");
        textViewSavedData.setText(savedName);

        // Set up the button click listener to save data
        buttonSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = editTextData.getText().toString();

                // Save data to SharedPreferences
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(KEY_NAME, name);
                editor.apply(); // apply() saves the data asynchronously
```

```
            // Update the TextView with saved name
            textViewSavedData.setText(name);
        }
    });
  }
}
```

OUTPUT:-



Step 4: Explanation of Key Concepts

- **SharedPreferences**: This is used to store simple data as key-value pairs. It is persistent across app restarts.
- getSharedPreferences("MyPreferences", MODE_PRIVATE): This retrieves the SharedPreferences file named "MyPreferences".
- **putString(KEY_NAME, name)**: This saves a String value with the key KEY_NAME.
- apply(): This method saves the data asynchronously.

Step 5: Run the Application

1. Press the **Run** button in Android Studio to compile and run the app.
2. Enter a name in the input field and press the "Save" button.

3. The app will display the saved name in the TextView and persist it even when you close and reopen the app.

Step 6: Testing the Application

- The application will persist the entered data even if the user exits and restarts the app.
- If you open the app again, the saved data will be loaded into the TextView.

8) create an android application to create and use services

Steps to Create an Android Application with a Service:

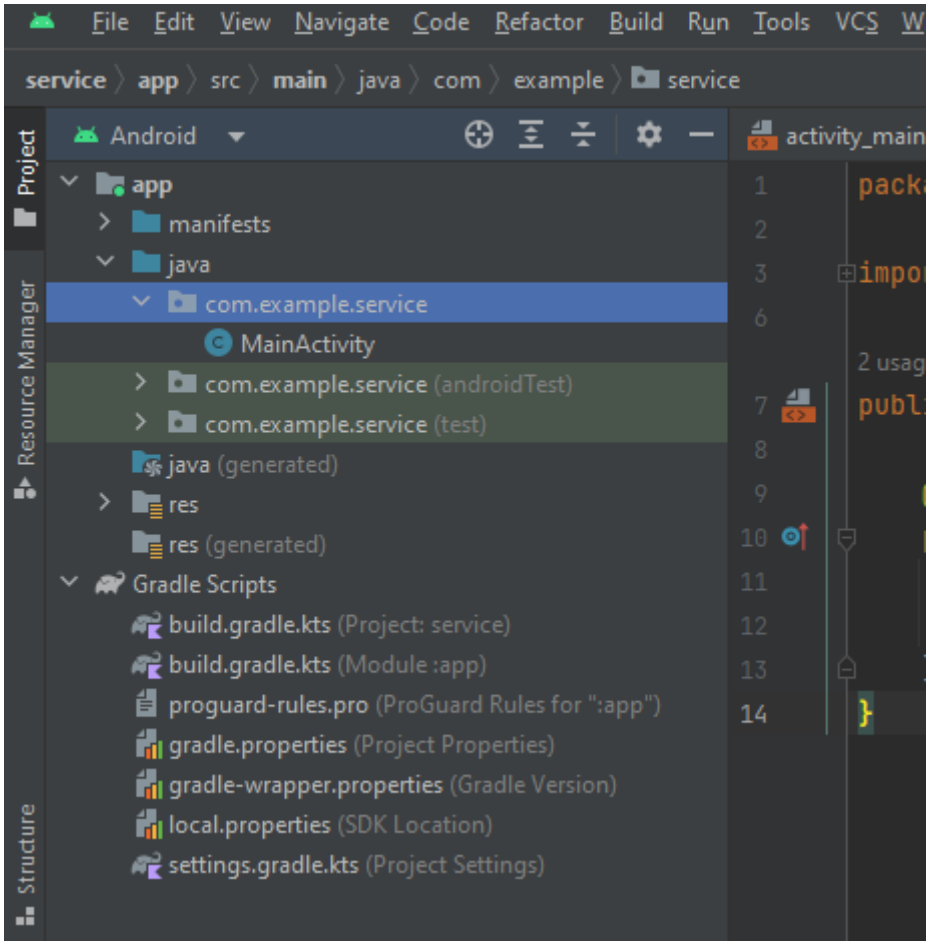
Step 1: Create a New Project

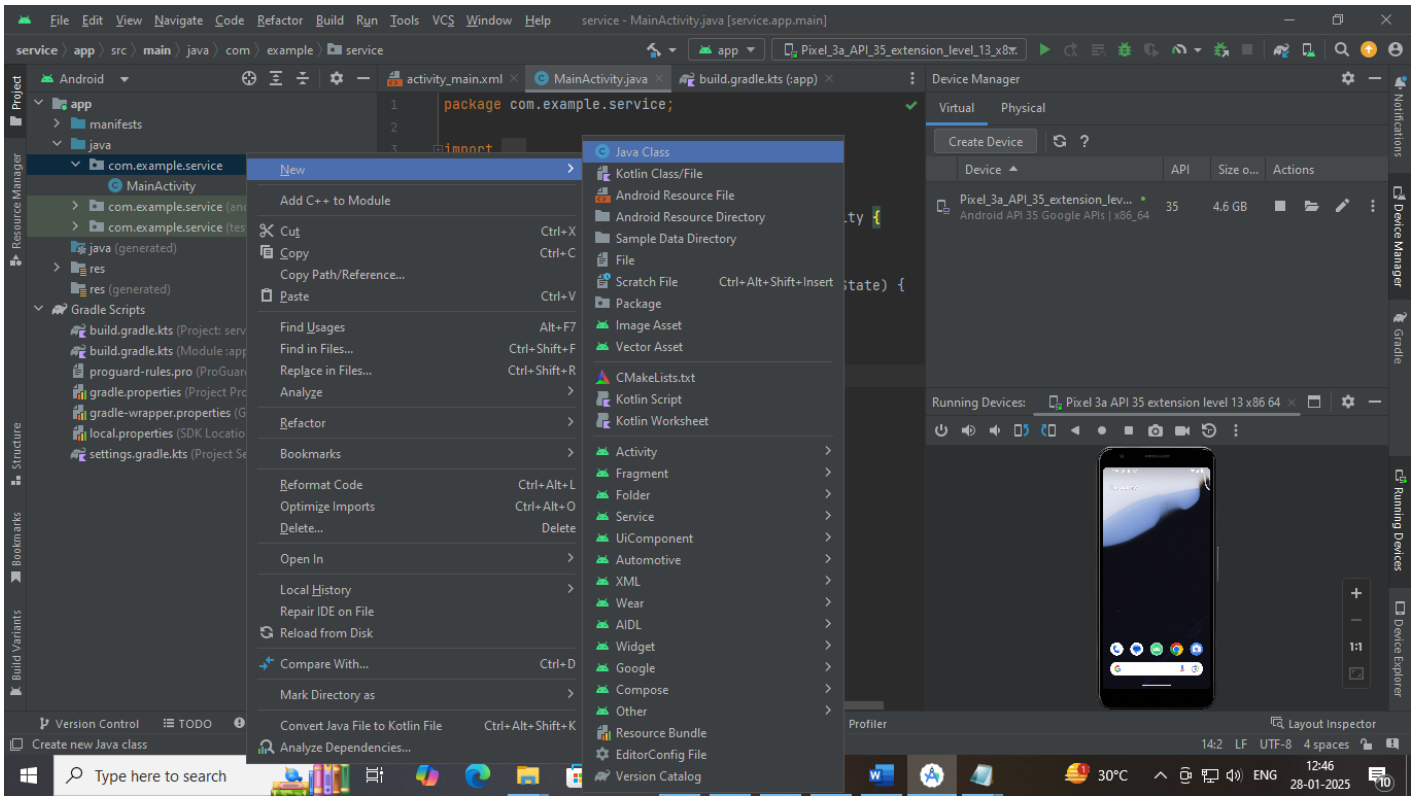
1. Open Android Studio.
2. Click on **File** > **New** > **New Project**.
3. Select **Empty Activity**.
4. Name your application (e.g., **ServiceDemo**).
5. Set the language to **Java** (or **Kotlin**, but I'll use **Java** here).

Step 2: Create the Service

In this example, we'll create a BackgroundService that logs a message every 5 seconds.

1. Right-click the java folder, select **New** > **Java Class**, and name it BackgroundService.


File Edit View Navigate Code Refactor Build Run Tools VCS W
service app src main java com example service
Android
activity_main
Project
Resource Manager
Structure
app
manifests
java
com.example.service
MainActivity
com.example.service (androidTest)
com.example.service (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: service)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
service - MainActivity.java [service.app.main]
package com.example.service;
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Analyze Dependencies...
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
CMakeLists.txt
Kotlin Script
Kotlin Worksheet
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
AIDL
Widget
Google
Compose
Other
Resource Bundle
EditorConfig File
Version Catalog
Device Manager
Virtual
Physical
Create Device
Pixel_3a_API_35_extension_lev...
Android API 35 Google APIs | x86_64
Running Devices:
Pixel 3a API 35 extension level 13 x86 64
Version Control
TODO
Profiler
Layout Inspector
Create new Java class
Type here to search
30°C
ENG
12:46
28-01-2025

CODE:-

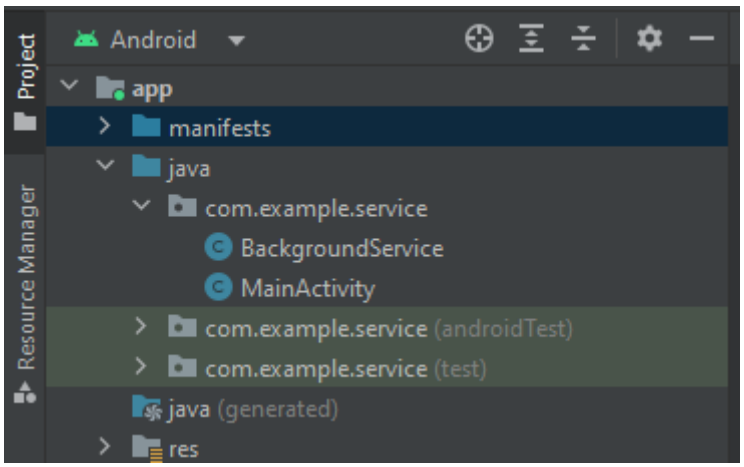
```
package com.example.service;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
public class BackgroundService  extends Service {

    private static final String TAG = "BackgroundService";

    // This method is called when the service is first created
    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "Service Created");
    }

    // This method is called whenever the service is started via startService
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "Service Started");

        // Perform a task (e.g., log messages every 5 seconds)
        new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 5; i++) {
                    try {
                        Thread.sleep(5000); // sleep for 5 seconds
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    Log.d(TAG, "Logging every 5 seconds... " + (i + 1));
                }
                stopSelf(); // Stop the service after the task is completed
            }
        }).start();
```

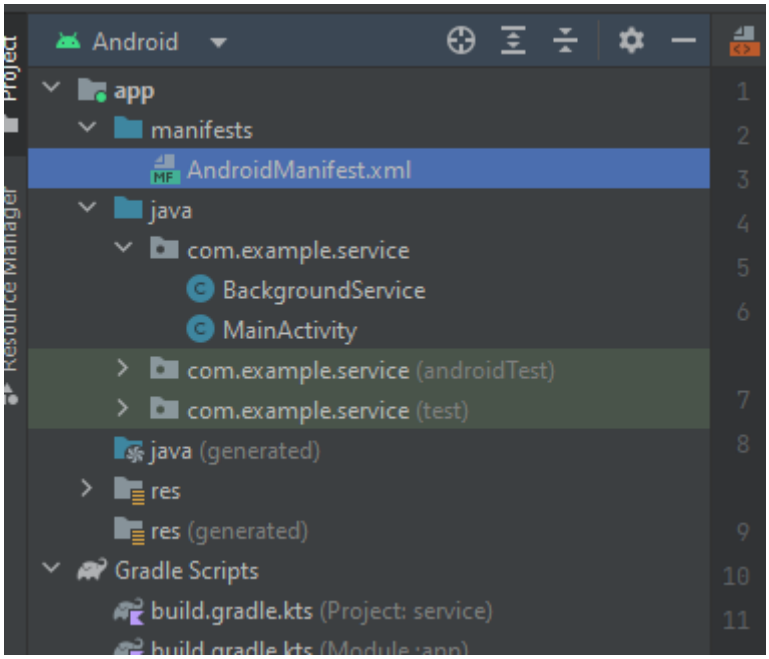
```
        // Return START_STICKY, which tells the system to restart the service if it's killed
        return START_STICKY;
    }

    // This method is called when the service is stopped
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "Service Destroyed");
    }

    // This method must be overridden, but we don't need to use it here
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Step 3: Register the Service in the Android Manifest

In order to use the service, you need to declare it in the AndroidManifest.xml file.

1. Open AndroidManifest.xml and add the service inside the <application> tag.





CODE:-

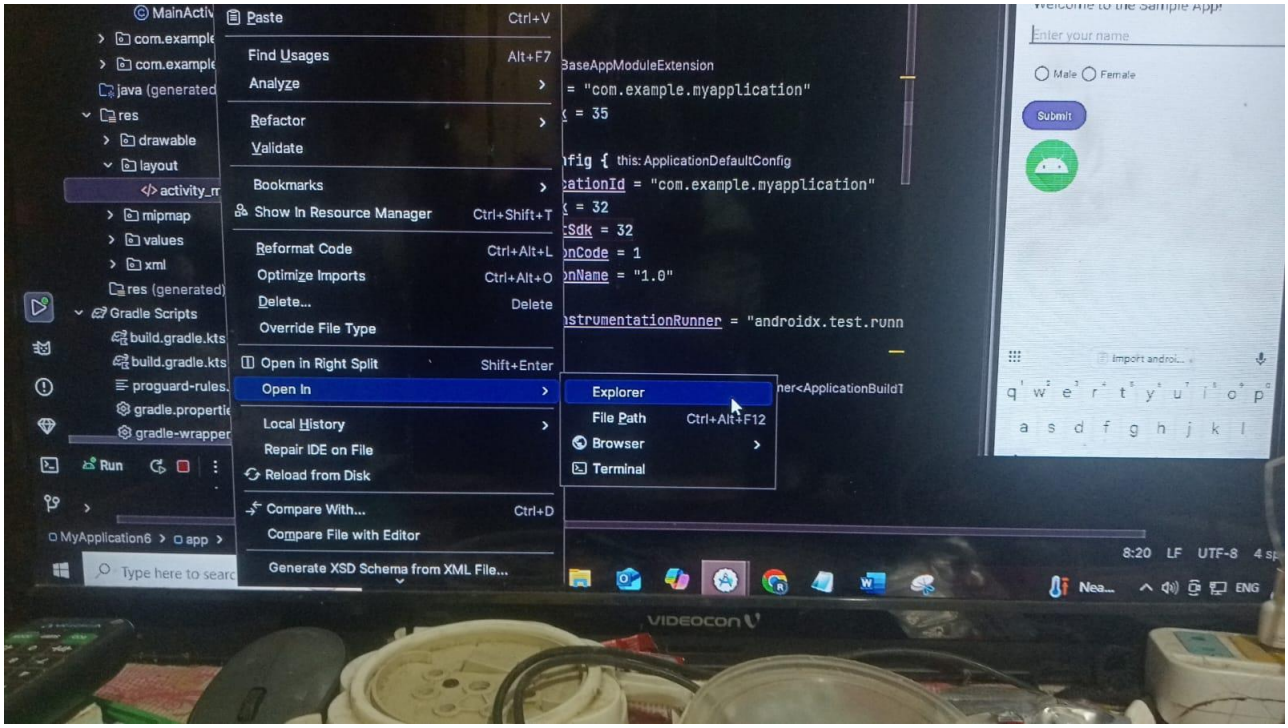
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.service">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ServiceDemo"
        android:theme="@style/Theme.Service">
        <!-- MainActivity with intent-filter (exported) -->
        <activity
            android:name=".MainActivity"
```

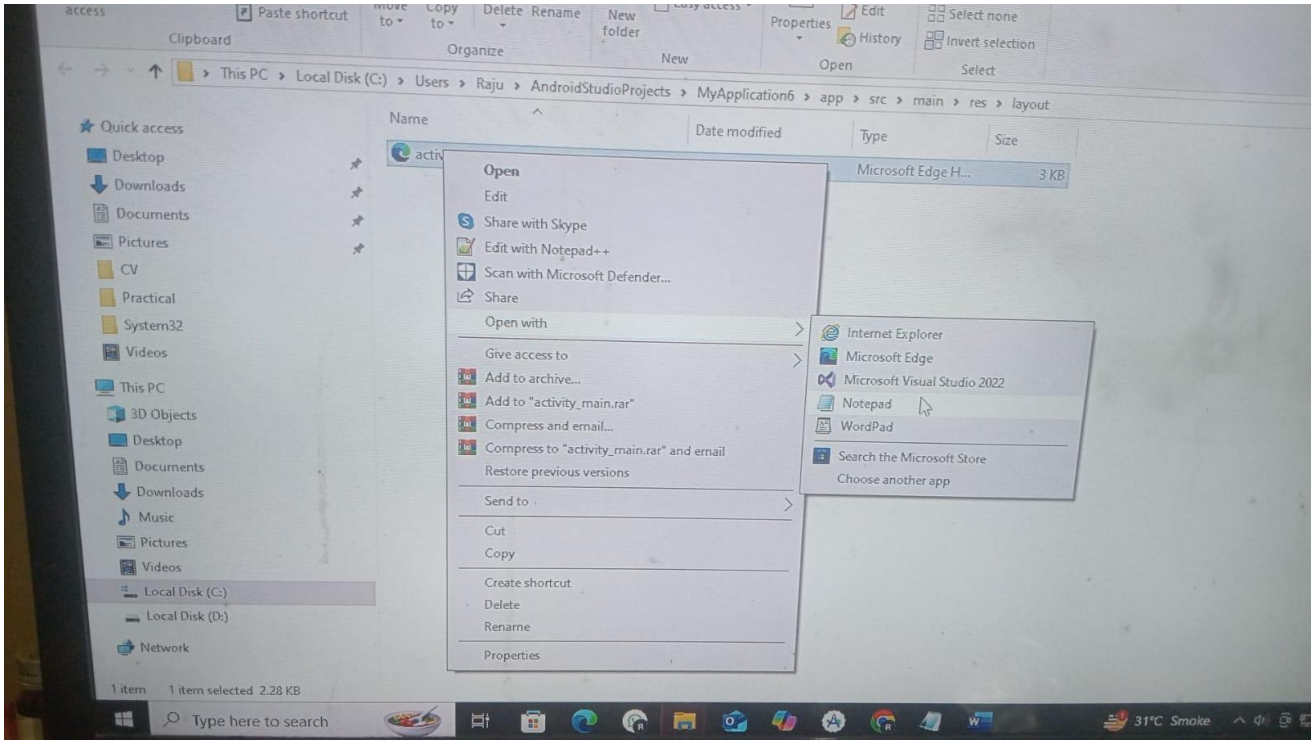
```
        android:label="Service Demo"
        android:theme="@style/Theme.Service"
        android:exported="true"> <!-- Add android:exported here -->
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service android:name=".BackgroundService" android:exported="false"/>
  </application>
</manifest>
```

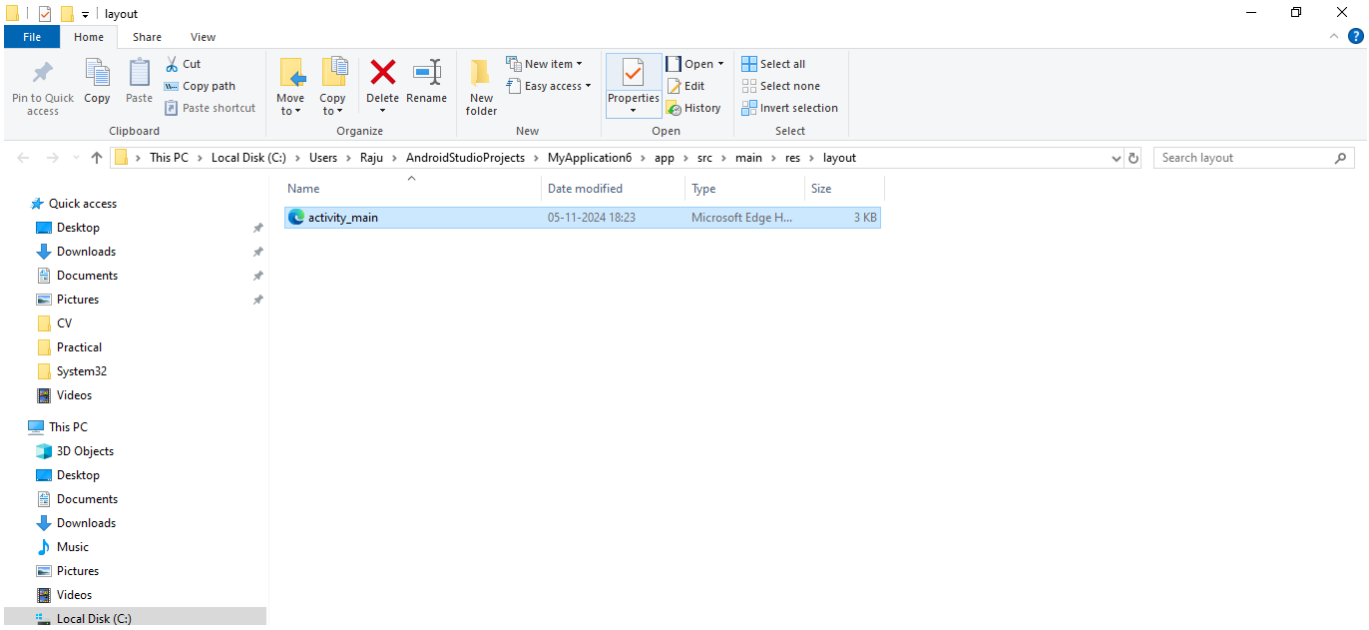
Step 4: Update the MainActivity to Start the Service

Now, we need to create a UI to interact with the service. We'll add a button in MainActivity that starts the service when clicked.

1. Open the activity_main.xml file and design the UI.


Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Validate
Bookmarks
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Override File Type
Open in Right Split Shift+Enter
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Compare File with Editor
Generate XSD Schema from XML File...
Explorer
File Path Ctrl+Alt+F12
Browser
Terminal
Enter your name
Male Female
Submit
8:20 LF UTF-8


This PC › Local Disk (C:) › Users › Raju › AndroidStudioProjects › MyApplication6 › app › src › main › res › layout
Open
Edit
Share with Skype
Edit with Notepad++
Scan with Microsoft Defender...
Share
Open with
Give access to
Add to archive...
Add to "activity_main.rar"
Compress and email...
Compress to "activity_main.rar" and email
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Internet Explorer
Microsoft Edge
Microsoft Visual Studio 2022
Notepad
WordPad
Search the Microsoft Store
Choose another app




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">

    <!-- Button to start the service -->
    <Button
        android:id="@+id/startServiceButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Service" />

    <!-- TextView to show status -->
    <TextView
```

```
        android:id="@+id/statusTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Service not started"
        android:layout_marginTop="20dp" />

</LinearLayout>
```

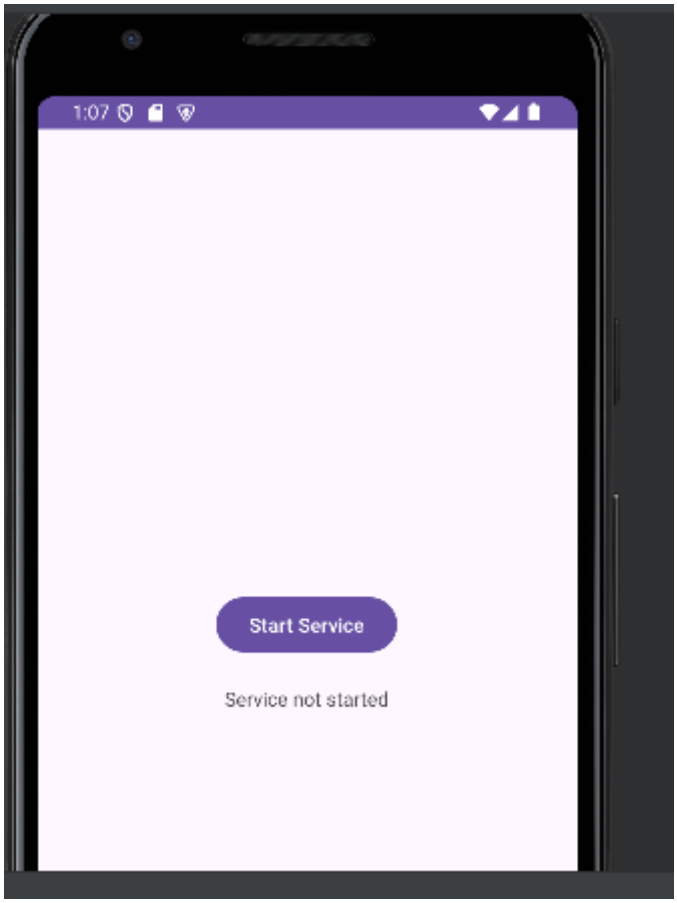
2) Now, In MainActivity.java, implement the logic to start and stop the service when the button is clicked.

CODE:-

```
package com.example.service;

    import android.content.Intent;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.widget.TextView;

    import androidx.appcompat.app.AppCompatActivity;
    public class MainActivity extends AppCompatActivity {

        private Button startServiceButton;
        private TextView statusTextView;

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

            startServiceButton = findViewById(R.id.startServiceButton);
            statusTextView = findViewById(R.id.statusTextView);

            startServiceButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // Start the BackgroundService
                    Intent serviceIntent = new Intent(MainActivity.this, BackgroundService.class);
```

```
            startService(serviceIntent);

            // Update the status
            statusTextView.setText("Service is running...");
        }
    });
  }
}
```

OUTPUT:-



Explanation:

1. **android:exported="true" for MainActivity**:

- The <activity> tag is the entry point of the app, and it has an intent-filter defined with MAIN and LAUNCHER. Since you want this activity to be accessible to the system (and possibly other apps to launch your app), you set android:exported="true".

2. **android:exported="false" for BackgroundService**:

- Your service (BackgroundService) does not have an intent filter, and it should only be accessible within your app. Hence, you set android:exported="false" for the service.

Why is android:exported Important?

- **Android 12 introduced stricter security policies** that require developers to explicitly define whether a component (activity, service, or broadcast receiver) should be available to other apps. This helps prevent potential security vulnerabilities where sensitive components might unintentionally be exposed to other apps.

IMPORTANT POINTS:-

- Add the android:exported attribute to all components (like activities, services, or receivers) that have an intent-filter.
- If a component should be accessible from other apps, set android:exported="true".
- If the component should only be used within your app, set android:exported="false".