INDEX

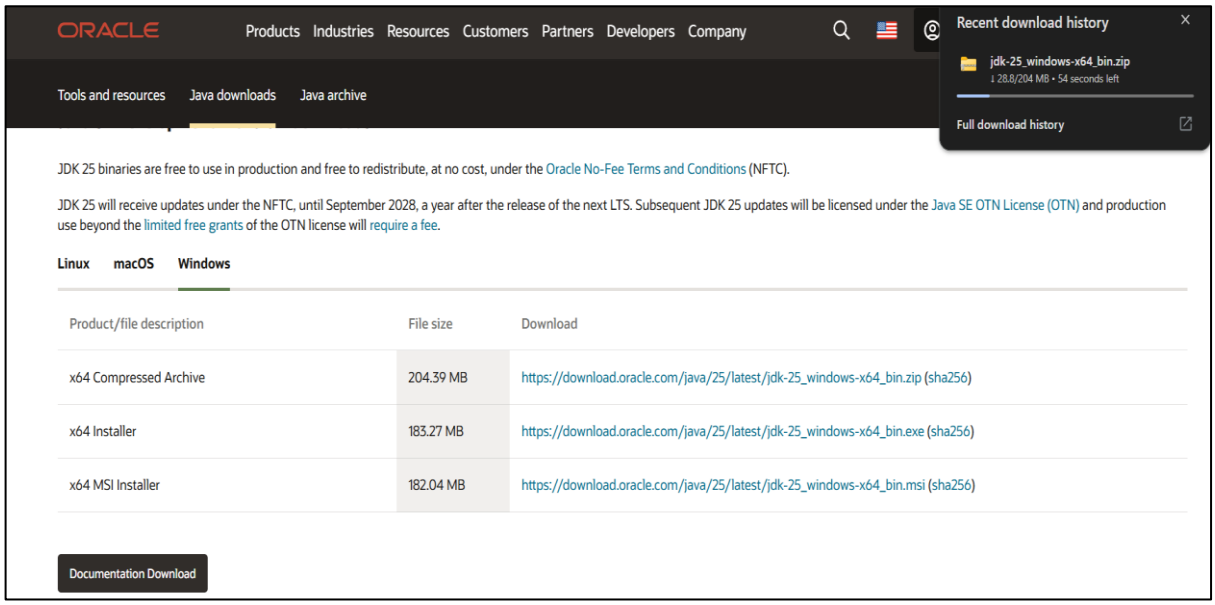
PRACTICAL 1

1. Installation of Java, android Framework.

Part 1: Install the Java Development Kit (JDK)

The JDK is required to compile and run Java programs.

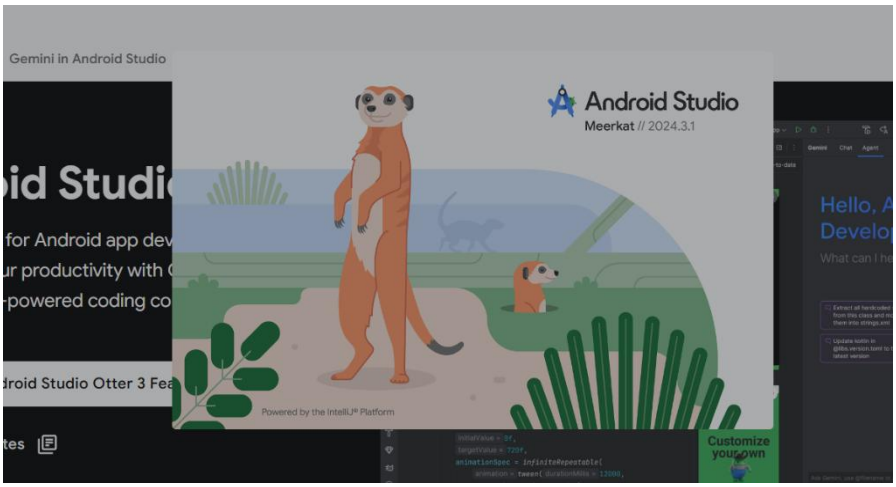
1. **Download the JDK**: Go to the official Oracle Java Downloads page or use a free and open-source option like OpenJDK. Download the latest version compatible with your operating system (Windows, macOS, or Linux).
2. **Install the JDK**: Run the downloaded installer file (.exe, .dmg, etc.) and follow the on-screen instructions, accepting the default installation settings.
3. **Set Environment Variables (Optional but Recommended)**: Many Java applications look for the JAVA_HOME environment variable. You may need to manually set it to your JDK installation directory (e.g., C:\Program Files\Java\jdk-21 on Windows) and ensure the bin directory is in your system's PATH variable.
4. **Verify Installation**: Open a command prompt or terminal and run the command javac -version to confirm the installation was successful and to see the installed version.



Part 2: Install Android Studio and Android SDK

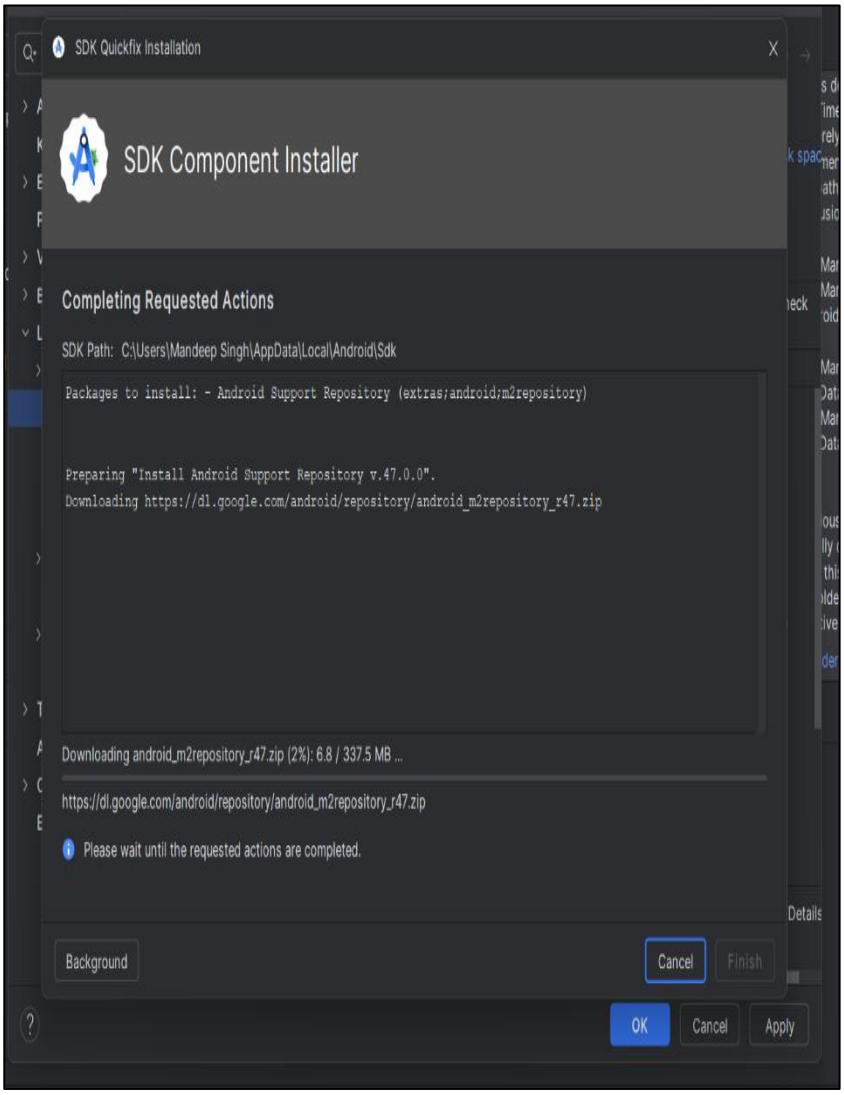
Android Studio is the official Integrated Development Environment (IDE) for Android app development and comes bundled with the necessary Android SDK (Software Development Kit).

1. **Download Android Studio**: Visit the official Android Developers website and click the "Download Android Studio" button.
2. **Install Android Studio**:
    - Run the downloaded installer file.
    - Follow the **Setup Wizard**, accepting the default settings for the most part.
    - The wizard will automatically download and install essential components like the Android SDK Platform-Tools, SDK Build-Tools, and Android Emulator.



3. **Verify Installation and Configure SDK**:
    - Once the installation is complete, launch Android Studio.
    - If prompted, follow the initial setup wizard to ensure all SDK components are installed.
    - You can manually check and manage installed SDK platforms and tools by navigating to **Tools** > **SDK Manager** within Android Studio. Ensure you have the required API levels and tools selected.

You are now ready to build your first Android application using Java within the Android Studio environment.


SDK Quickfix Installation
SDK Component Installer
Completing Requested Actions
SDK Path: C:\Users\Mandeep Singh\AppData\Local\Android\Sdk
Packages to install: - Android Support Repository (extras;android;m2repository)
Preparing "Install Android Support Repository v.47.0.0".
Downloading https://dl.google.com/android/repository/android_m2repository_r47.zip
Downloading android_m2repository_r47.zip (2%): 6.8 / 337.5 MB ...
https://dl.google.com/android/repository/android_m2repository_r47.zip
Please wait until the requested actions are completed.
Background
Cancel
Finish
OK
Cancel
Apply

PRACTICAL 2

2. Android SDK Manager and its all components.

The Android SDK Manager is a tool used to view, install, update, and manage the various packages, tools, and platforms that comprise the Android Software Development Kit (SDK). The Android SDK itself is a comprehensive set of development tools and libraries essential for building, testing, and deploying Android applications.

If you are using Android Studio, you can access the SDK Manager by clicking **Tools > SDK Manager** or by clicking the SDK Manager icon in the toolbar. If you are not using Android Studio, you can use the sdk manager command-line tool.

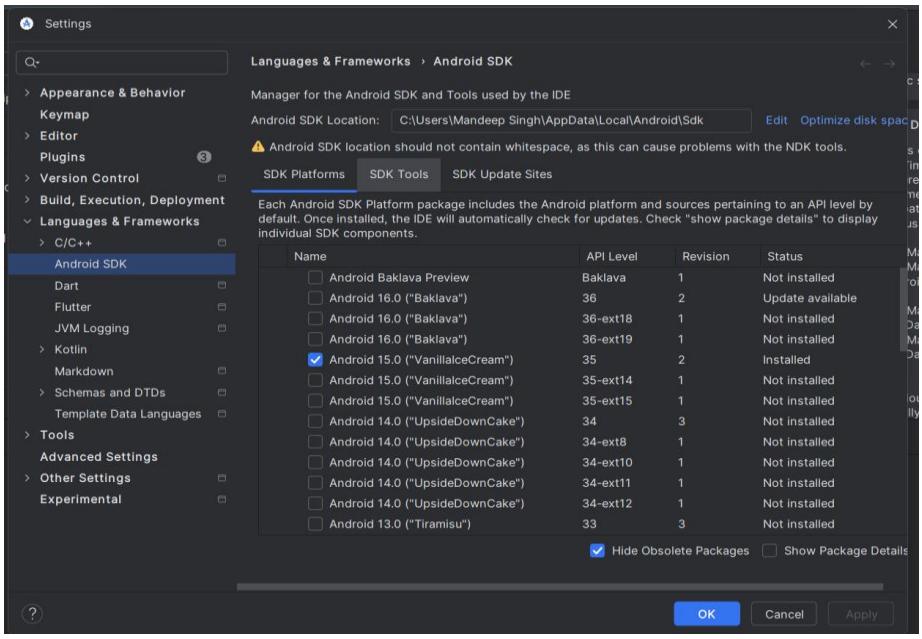
Components of the Android SDK

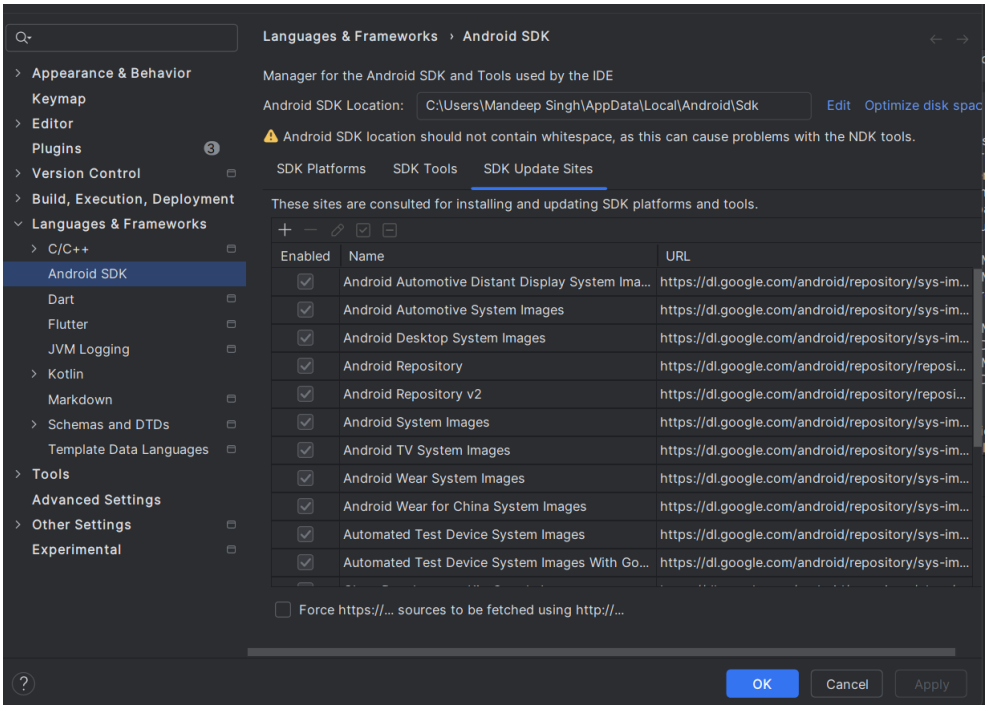
The main components managed by the SDK Manager are categorized into several groups:

- **SDK Platforms**: Each platform package includes the Android operating system itself (for a specific API level), libraries, and source code for that version. This allows developers to build and test apps for different versions of Android (e.g., Android 13, Android 14).
- **SDK Tools**: This category contains essential tools and utilities for the development process. Key components include:
    - **Android SDK Build-Tools**: These are required to build an Android application's actual binaries. They include tools for building, debugging, running, and testing apps.
    - **Android SDK Platform-Tools**: A collection of tools that interface with the Android platform, primarily the Android Debug Bridge (ADB) and fastboot. ADB is a command-line tool for communicating with an actual or virtual device for installing and debugging apps.
    - **Android SDK Command-Line Tools**: A set of tools for managing SDK packages and building apps from the command line, which includes the sdk manager itself.
    - **Android Emulator**: A virtual device that simulates an actual Android mobile device, allowing developers to run and test applications on a wide range of devices (phones, tablets, Wear OS, Android TV) without needing physical hardware.
    - **Android Virtual Device (AVD) Manager**: A graphical user interface (GUI) tool within Android Studio for creating and managing these virtual devices, configuring their hardware properties and system images.

- **System Images**: These are necessary to run the Android Emulator. They contain the guest operating system for a virtual Android device.

- **Support Libraries / AndroidX**: A collection of libraries that provide backward compatibility for newer features and APIs on older Android versions, and offer additional components for modern app design (e.g., Material Design components).
- **NDK (Native Development Kit)**: A set of tools that allows developers to implement parts of their apps using native-code languages like C and C++, which can be useful for performance-intensive applications like games.





PRACTICAL 3

3. Programs based on the overriding, constructor, classes in Java.

In Java, constructors cannot be overridden because they are not inherited by subclasses.
However, you can achieve method overriding within an inheritance hierarchy.
The super keyword is used to call a parent class's constructor or method from a subclass.

Below are Java programs demonstrating classes, constructors, and method overriding.

Classes and Constructors

This program demonstrates the use of a class and multiple constructors (constructor overloading) to initialize objects in different ways.

```
class Car {
    String model;
    int year;

    // Default constructor (no arguments)
    Car() {
        this.model = "Unknown";
        this.year = 0;
    }

    // Parameterized constructor
    Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    void displayDetails() {
        System.out.println("Model: " + model + ", Year: " + year);
    }
}

public class ConstructorExample {
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2020); // Calls parameterized constructor
        Car car2 = new Car();               // Calls default constructor

        car1.displayDetails();
        car2.displayDetails();
    }
}
```

Output

```
Model: Toyota, Year: 2020
Model: Unknown, Year: 0
```

Method Overriding

This program demonstrates method overriding, where a subclass provides a specific implementation of a method that is already defined in its superclass. It also uses the super keyword to call the parent class's method from the child class's overridden method.

```
class Animal {
    // Overridden method
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    // Overriding method
    @Override
    void makeSound() {
        super.makeSound(); // Call parent's method
        System.out.println("Dog barks");
    }
}

public class OverridingExample {
    public static void main(String[] args) {
        Animal myDog = new Dog(); // Runtime polymorphism
        myDog.makeSound();        // Calls the Dog's overridden method
    }
}
```

Output

```
Animal makes a sound
Dog barks
```

Constructor Chaining with super()

While constructors cannot be overridden, the super() keyword is crucial in a subclass's constructor to call a corresponding constructor in the parent class, ensuring proper initialization of inherited members.

```
class ParentClass {
    ParentClass(int value) {
        System.out.println("Parent Constructor, value: " + value);
    }
}

class ChildClass extends ParentClass {
    ChildClass() {
        super(10); // Calls the parent's parameterized constructor
        System.out.println("Child Constructor");
    }
}
S
public class ConstructorChaining {
    public static void main(String[] args) {
        ChildClass obj = new ChildClass();
    }
}
```

Output

```
Parent Constructor, value: 10
Child Constructor
```

PRACTICAL 4

4. Programs based on the Final, this and static keyword in Java.

The final, this, and static keywords in Java are modifiers used in various contexts to define class members, control inheritance, or reference the current object. Here are the core concepts and programs demonstrating their use:

final Keyword

The final keyword is used to restrict a class, method, or variable. A final entity cannot be modified after it has been initialized.

```
class FinalClassExample {
    // Final variable (constant)
    final int MAX_SPEED = 100;

    // Final method: cannot be overridden
    public final void display() {
        System.out.println("This is a final method. Max Speed: " + MAX_SPEED + " mph.");
    }
}

// Trying to inherit a final class would cause a compilation error
// final class AnotherClass extends FinalClassExample {} // Error

class SubClassExample extends FinalClassExample {
    // Trying to override the final method would cause a compilation error
    // public final void display() { System.out.println("Overridden method"); } // Error
}

public class FinalDemo {
    public static void main(String[] args) {
        FinalClassExample obj = new FinalClassExample();
        obj.display();

        // Trying to change the final variable would cause a compilation error
        // obj.MAX_SPEED = 120; // Error: cannot assign a value to final variable
MAX_SPEED
    }
}
```

Output

The this Keyword

The this keyword is a reference to the current object. It is primarily used to:

```
This is a final method. Max Speed: 100 mph.
```

- Differentiate between instance variables and local variables/parameters with the same name.
- Invoke the current class's constructor (using this()).
- Invoke the current class's method (using this.methodName()).

```
class Student {
    int rollno;
    String name;
    float fee;

    // Constructor using 'this' to refer to instance variables
    Student(int rollno, String name, float fee) {
        this.rollno = rollno;
        this.name = name;
        this.fee = fee;
    }

    // Another constructor using 'this()' to call the first constructor (constructor
chaining)
    Student() {
        this(999, "Default", 0.0f); // Calls the parameterized constructor
        System.out.println("Default constructor called.");
    }

    void display() {
        // Using 'this' explicitly to call another method
        this.showDetails();
    }

    void showDetails() {
        System.out.println("ID: " + rollno + ", Name: " + name + ", Fee: " + fee);
    }
}

public class ThisDemo {
    public static void main(String[] args) {
        Student s1 = new Student(101, "Karan", 50000f);
        Student s2 = new Student(); // Uses the default constructor, which calls the
parameterized one
```

```
        s1.display();
        s2.display();
    }
}
```

Output

```
Default constructor called.
ID: 101, Name: Karan, Fee: 50000.0
ID: 999, Name: Default, Fee: 0.0
```

The static Keyword

The static keyword means that the member belongs to the class itself, rather than to an instance (object) of the class. It is used for:

- static variables (class variables) which are shared by all instances.
- static methods which can be called without creating an object.
- static blocks which are executed when the class is first loaded into memory.

```
class Counter {
    // Static variable: shared by all instances
    static int count = 0;

    // Instance variable: each object has its own copy
    String instanceName;

    // Static block: runs once when the class is loaded
    static {
        System.out.println("Static block executed: Class Counter loaded.");
    }

    Counter(String name) {
        this.instanceName = name;
        count++; // Increment the shared counter
    }
    // Static method: can access static members directly
    static void showCount() {
        System.out.println("Total objects created: " + count);
    }
    // Instance method: can access both static and instance members
    void display() {
        System.out.println("Object Name: " + instanceName + ", Current shared count: " +
count);
```

```
    }
}
public class StaticDemo {
    public static void main(String[] args) {
        // Accessing static method using class name (recommended)
        Counter.showCount();

        Counter c1 = new Counter("Object1");
        c1.display();

        Counter c2 = new Counter("Object2");
        c2.display();

        // Static variables and methods can be accessed from a static context (like main)
        System.out.println("Count is now: " + Counter.count);
    }
}
```

Output

```
Static block executed: Class Counter loaded.
Total objects created: 0
Object Name: Object1, Current shared count: 1
Object Name: Object2, Current shared count: 2
Count is now: 2
```

PRACTICAL 5

5. Directory Structure of an Android Project, Common Default Resources Folders, The Values Folder, Leveraging Android XML.

An Android project uses a specific directory structure and common default resource folders to organize code and assets effectively [1, 2]. The structure separates source files, resources, and configuration files, making the project manageable and scalable [1].

Directory Structure of an Android Project

The primary directories in an Android project are:

- app/: Contains the application code and resources.
- app/src/: Contains source code and resources for the app module, typically including:
    - main/: Contains the primary source code and resources for the application.
        - java/ or kotlin/: This is where all the Kotlin or Java source code files (e.g., activities, services, fragments) are stored [1, 3].
        - res/: Contains all non-code application resources, such as layouts, drawable images, and string values [1, 3].
        - AndroidManifest.xml: The application manifest file, which describes the essential information about the app to the Android system [1, 3].
- build.gradle (or build.gradle.kts): Files that define the project's build configuration, dependencies, and settings for each module [1].
- gradle/ and gradlew (or gradlew.bat): Files related to the Gradle build system itself.

Common Default Resources Folders (res/)

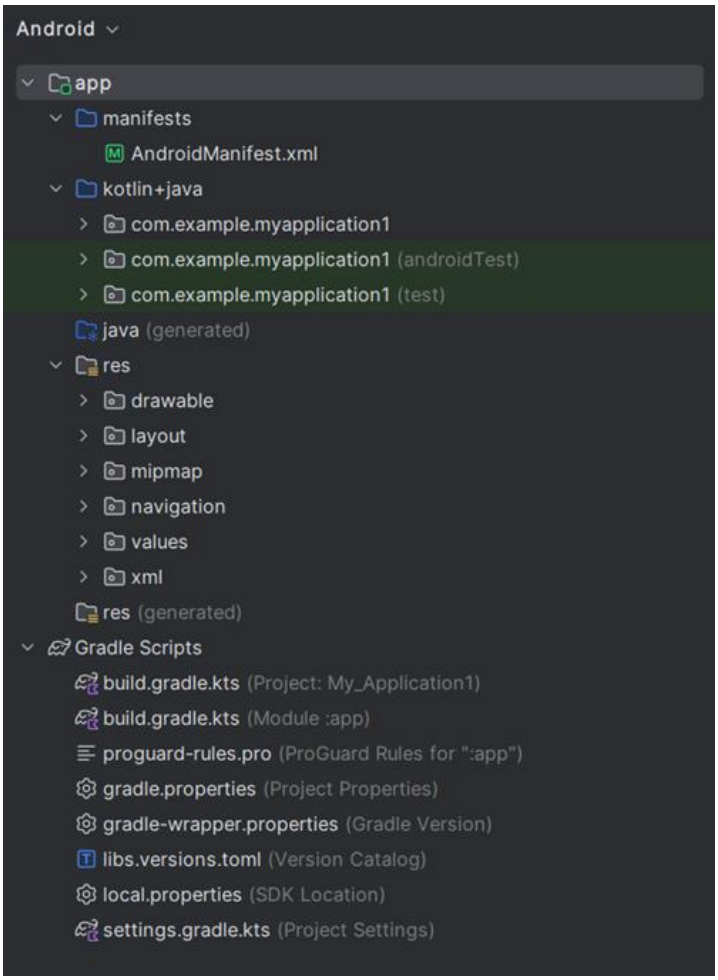
The res/ directory houses various subdirectories, each designated for specific types of resources:

- drawable/: Contains image files (e.g., PNG, JPEG, SVG) and other drawable resources like XML-defined shapes or state lists [1, 3].
- layout/: Contains XML files that define the user interface layouts for your app's activities and fragments [1, 3].
- mipmap/: Used for storing app launcher icons at different densities [1, 3].
- values/: Contains XML files that store simple values like strings, integers, and colors [1, 3]. This folder is essential for defining static data used throughout the application.
- menu/: Contains XML files that define the options and contextual menus in your application [3].
- xml/: Contains various configuration files that can be read at runtime, often used for app preferences or device administration settings [3].
- raw/: Contains arbitrary raw asset files like audio or video files that must be accessed in their raw format [3].

The values/ Folder

The values/ folder is crucial for separating data from code, facilitating easier localization and maintenance. It typically includes:

- colors.xml: Defines color values using hexadecimal codes [3, 4].



- strings.xml: Defines all the text strings used in the user interface. This is vital for internationalization (translating the app into different languages) [1, 3, 4].
- styles.xml: Defines styles and themes for the application's UI elements, promoting consistent design across the app [3, 4].
- dimens.xml: Defines dimension values (e.g., padding, margins, text sizes) [3].
- arrays.xml: Defines arrays of strings or integers [3].

Leveraging Android XML

Android XML is used extensively to define UI layouts, styles, and static values, offering several advantages:

- Separation of Concerns: It cleanly separates the UI structure and design from the underlying logic defined in Kotlin/Java code [2].
- Flexibility and Adaptability: Different XML files can be created for various screen sizes, orientations, and languages. The Android system automatically selects the most appropriate resource based on the device's configuration [1, 2].
- Readability: XML provides a structured, human-readable format for defining UI components and properties [1].

You can manage these resources and the entire project structure using Android Studio. More detailed information on organizing resources is available in the official Android documentation on providing resources and the Android developer guides.

PRACTICAL 6

6. Applications based on Text Boxes and Button.

STEP-1: Open Project
Open your existing Android Studio project
STEP-2: Open Layout File
res → layout → activity_main.xml
STEP-3: Write XML Code (UI DESIGN)

```
<?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="20dp">

    <!-- Text Box -->
    <EditText
        android:id="@+id/etText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Text"/>

    <!-- Button -->
    <Button
        android:id="@+id/btnDisplay"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Display Text"/>
</LinearLayout>
```

What this does

- EditText → takes user input
- Button → performs action on click

STEP-4: Open Java File
java → package name → MainActivity.java
STEP-5: Write Java Code (LOGIC)

```
package com.example.myapplication;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

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

```
public class MainActivity extends AppCompatActivity {

    EditText etText;
    Button btnDisplay;

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

        // Link XML with Java
        etText = findViewById(R.id.etText);
        btnDisplay = findViewById(R.id.btnDisplay);

        // Button click event
        btnDisplay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String text = etText.getText().toString();

                Toast.makeText(MainActivity.this,
                        text,
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

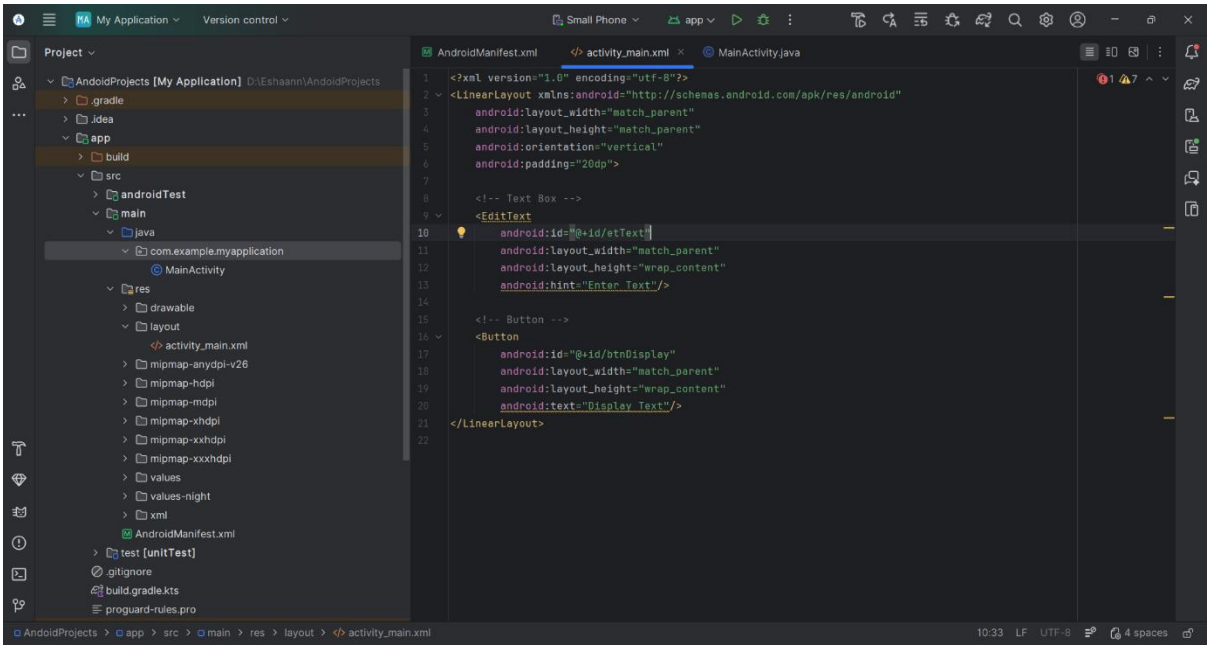
What this does

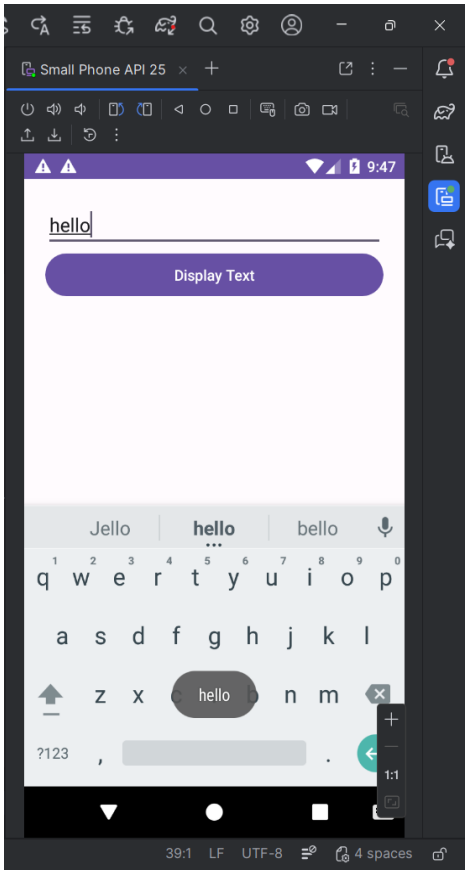
- Reads text from TextBox
- Shows it using Toast when button is clicked

STEP-6: Run the Application

1. Start Emulator
2. Click ▶ Run
3. Enter text in TextBox
4. Click Display Text

OUTPUT


My Application
Version control
Small Phone
app
Project
AndoidProjects [My Application] D:\Eshaann\AndoidProjects
.gradle
.idea
app
build
src
androidTest
main
java
com.example.myapplication
MainActivity
res
drawable
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
AndroidManifest.xml
activity_main.xml
MainActivity.java
<?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="20dp">
<!-- Text Box -->
<EditText
android:id="@+id/etText"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Enter Text"/>
<!-- Button -->
<Button
android:id="@+id/btnDisplay"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Display Text"/>
</LinearLayout>
AndoidProjects > app > src > main > res > layout > activity_main.xml
10:33 LF UTF-8 4 spaces


Small Phone API 25
9:47
hello
Display Text
Jello
hello
bello
q w e r t y u i o p
a s d f g h j k l
z x hello n m
?123
1:1
39:1 LF UTF-8 4 spaces

PRACTICAL 7

7. Applications based on Check Boxes and button.

STEP-1: Open Project

Open the same project used in previous practicals.

STEP-2: Open Layout File

res → layout → activity_main.xml

STEP-3: Write XML Code (UI DESIGN)

```
<?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="20dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select Subjects"
        android:textSize="18sp"/>

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

    <CheckBox
        android:id="@+id/cbAndroid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android"/>

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

    <Button
        android:id="@+id/btnSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit"/>
</LinearLayout>
```

STEP-4: Open Java File

MainActivity.java

STEP-5: Write Java Code (LOGIC)

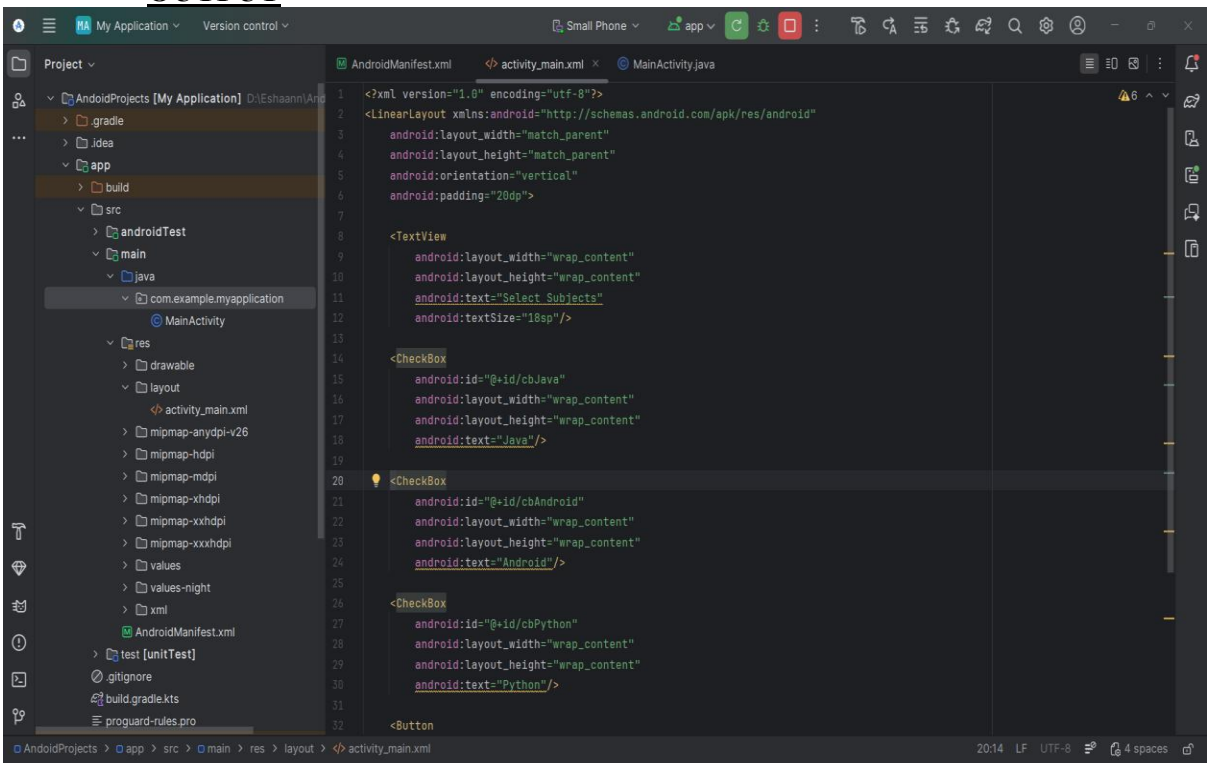
```
package com.example.myapplication;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    CheckBox cbJava, cbAndroid, cbPython;
    Button btnSubmit;

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

        cbJava = findViewById(R.id.cbJava);
        cbAndroid = findViewById(R.id.cbAndroid);
        cbPython = findViewById(R.id.cbPython);
        btnSubmit = findViewById(R.id.btnSubmit);

        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String result = "Selected: ";

                if (cbJava.isChecked())
                    result += "Java ";

                if (cbAndroid.isChecked())
                    result += "Android ";

                if (cbPython.isChecked())
                    result += "Python ";

                Toast.makeText(MainActivity.this,
                        result,
                        Toast.LENGTH_SHORT).show();
            }
```

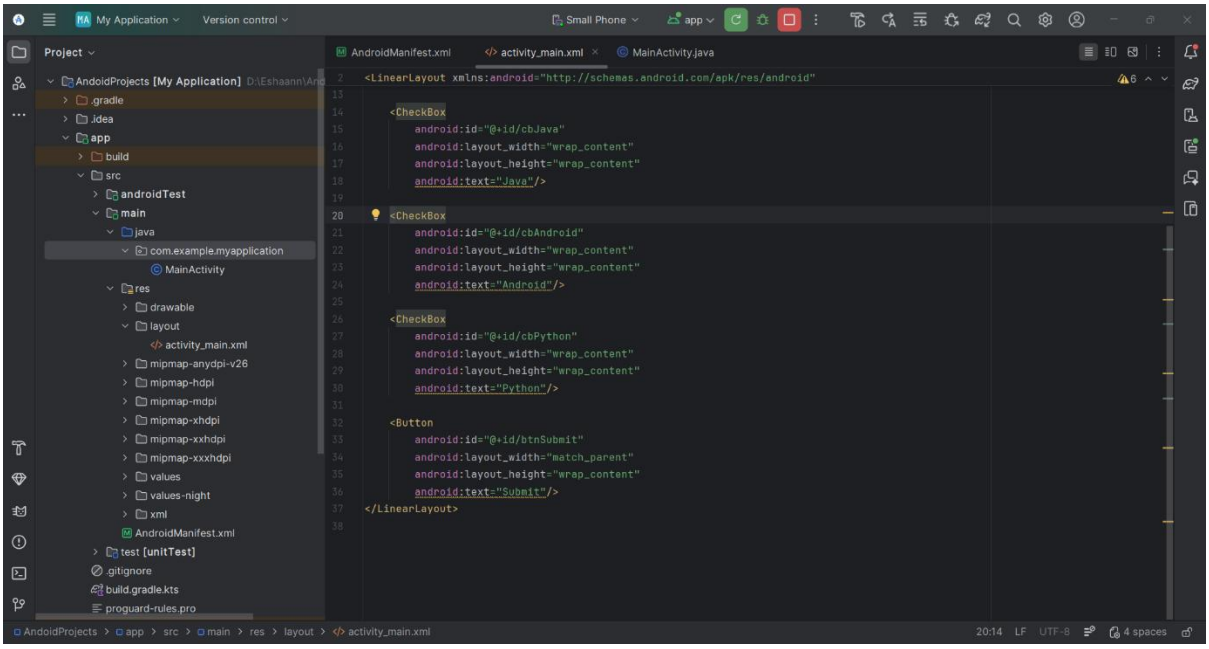
```
        });
    }
}
```

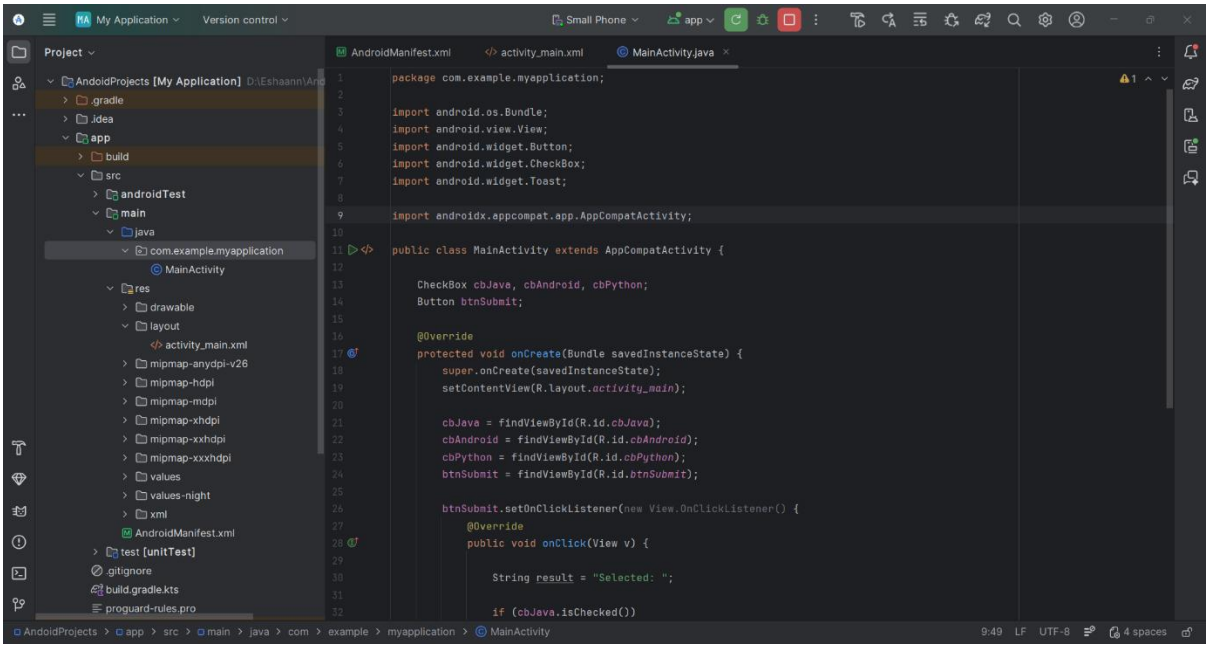
STEP-6: Run the Application

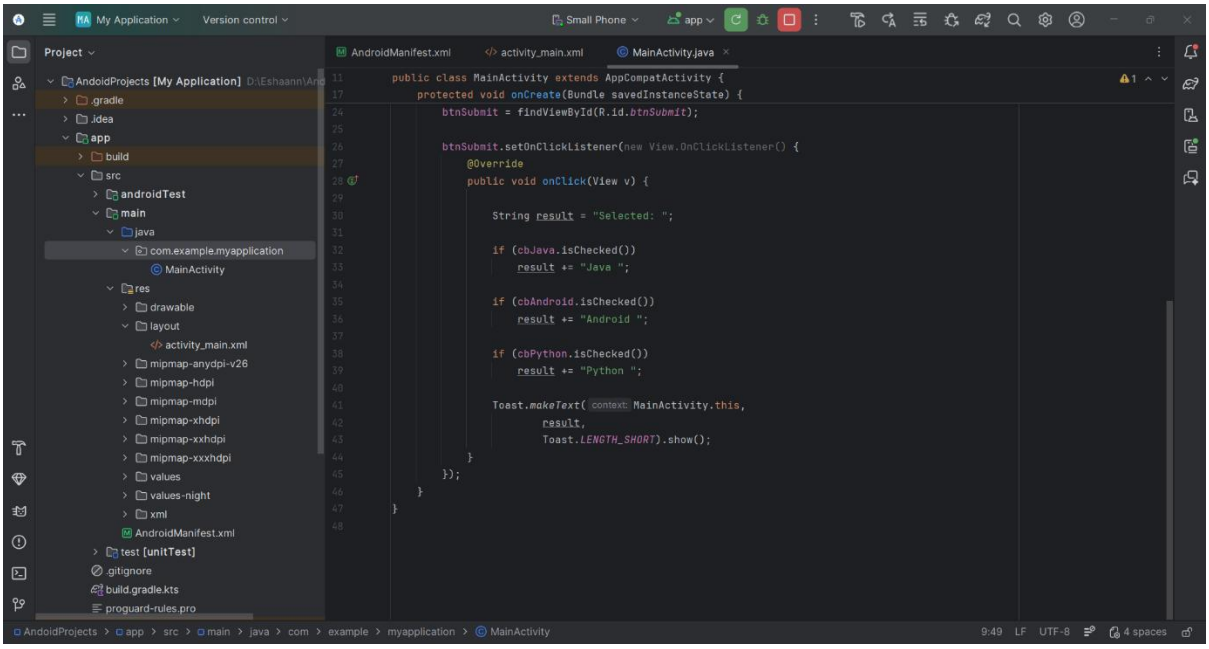
1. Start Emulator
2. Click ▶ **Run**
3. Select multiple checkboxes
4. Click **Submit**

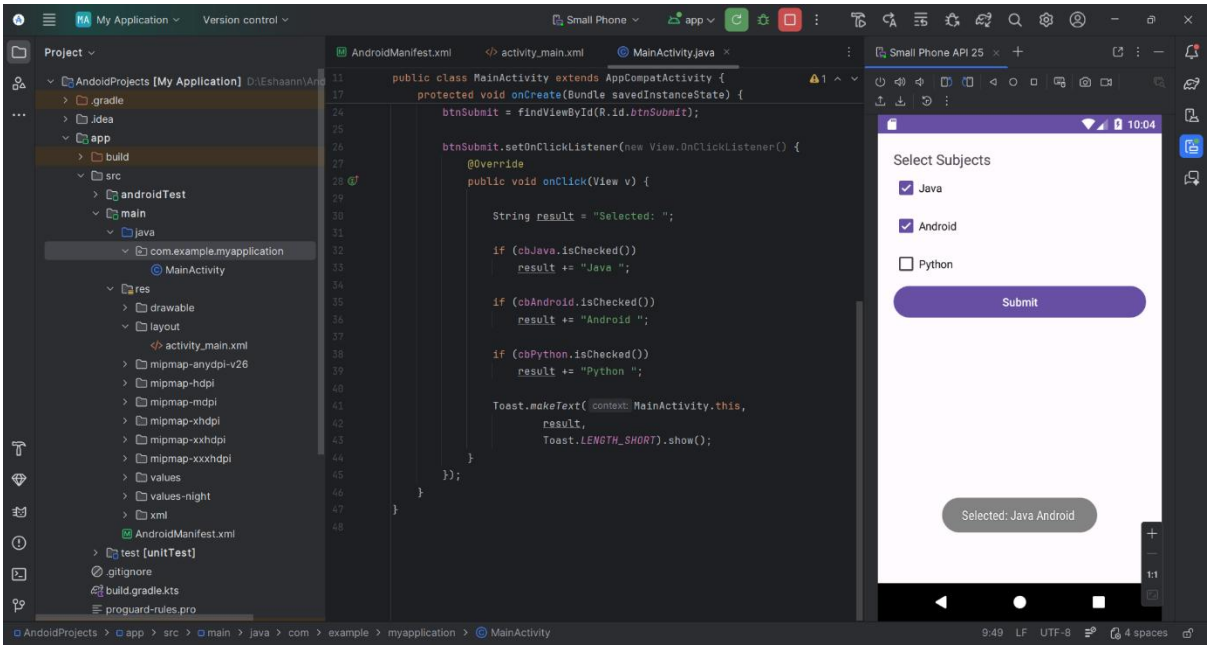
OUTPUT




My Application
Version control
Small Phone
app
Project
AndoidProjects [My Application]
.gradle
.idea
app
build
src
androidTest
main
java
com.example.myapplication
MainActivity
res
drawable
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
AndroidManifest.xml
activity_main.xml
MainActivity.java
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
<CheckBox
android:id="@+id/cbJava"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Java"/>
<CheckBox
android:id="@+id/cbAndroid"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Android"/>
<CheckBox
android:id="@+id/cbPython"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Python"/>
<Button
android:id="@+id/btnSubmit"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Submit"/>
</LinearLayout>
AndoidProjects > app > src > main > res > layout > activity_main.xml
20:14 LF UTF-8 4 spaces


My Application
Version control
Small Phone
app
Project
AndoidProjects [My Application]
.gradle
.idea
app
build
src
androidTest
main
java
com.example.myapplication
MainActivity
res
drawable
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
AndroidManifest.xml
activity_main.xml
MainActivity.java
package com.example.myapplication;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
CheckBox cbJava, cbAndroid, cbPython;
Button btnSubmit;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
cbJava = findViewById(R.id.cbJava);
cbAndroid = findViewById(R.id.cbAndroid);
cbPython = findViewById(R.id.cbPython);
btnSubmit = findViewById(R.id.btnSubmit);
btnSubmit.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String result = "Selected: ";
if (cbJava.isChecked())
AndoidProjects > app > src > main > java > com > example > myapplication > MainActivity
9:49 LF UTF-8 4 spaces


My Application
Version control
Small Phone
app
Project
AndroidManifest.xml
activity_main.xml
MainActivity.java
AndoidProjects [My Application]
.gradle
.idea
app
build
src
androidTest
main
java
com.example.myapplication
MainActivity
res
drawable
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
public class MainActivity extends AppCompatActivity {
protected void onCreate(Bundle savedInstanceState) {
btnSubmit = findViewById(R.id.btnSubmit);
btnSubmit.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String result = "Selected: ";
if (cbJava.isChecked())
result += "Java ";
if (cbAndroid.isChecked())
result += "Android ";
if (cbPython.isChecked())
result += "Python ";
Toast.makeText(context: MainActivity.this,
result,
Toast.LENGTH_SHORT).show();
});
AndoidProjects > app > src > main > java > com > example > myapplication > MainActivity
9:49 LF UTF-8 4 spaces


My Application
Version control
Small Phone
app
Project
AndroidManifest.xml
activity_main.xml
MainActivity.java
Small Phone API 25
AndoidProjects [My Application]
.gradle
.idea
app
build
src
androidTest
main
java
com.example.myapplication
MainActivity
res
drawable
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
public class MainActivity extends AppCompatActivity {
protected void onCreate(Bundle savedInstanceState) {
btnSubmit = findViewById(R.id.btnSubmit);
btnSubmit.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String result = "Selected: ";
if (cbJava.isChecked())
result += "Java ";
if (cbAndroid.isChecked())
result += "Android ";
if (cbPython.isChecked())
result += "Python ";
Toast.makeText(context: MainActivity.this,
result,
Toast.LENGTH_SHORT).show();
10:04
Select Subjects
Java
Android
Python
Submit
Selected: Java Android
1:1
AndoidProjects > app > src > main > java > com > example > myapplication > MainActivity
9:49 LF UTF-8 4 spaces

PRACTICAL 8

8. Applications based on Radio Buttons.

Step 1: Create New Project

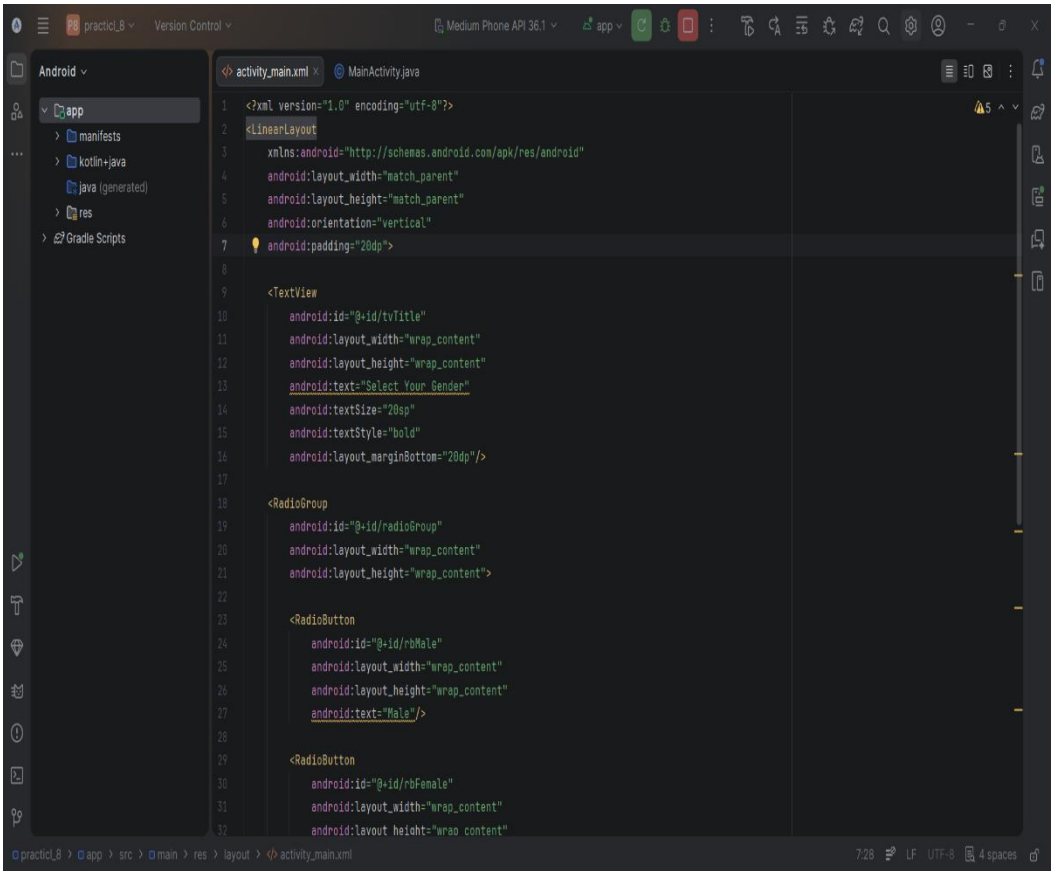
- Open **Android Studio**
- Click **New Project**
- Select **Empty Activity**
- Language: **Java**
- Finish

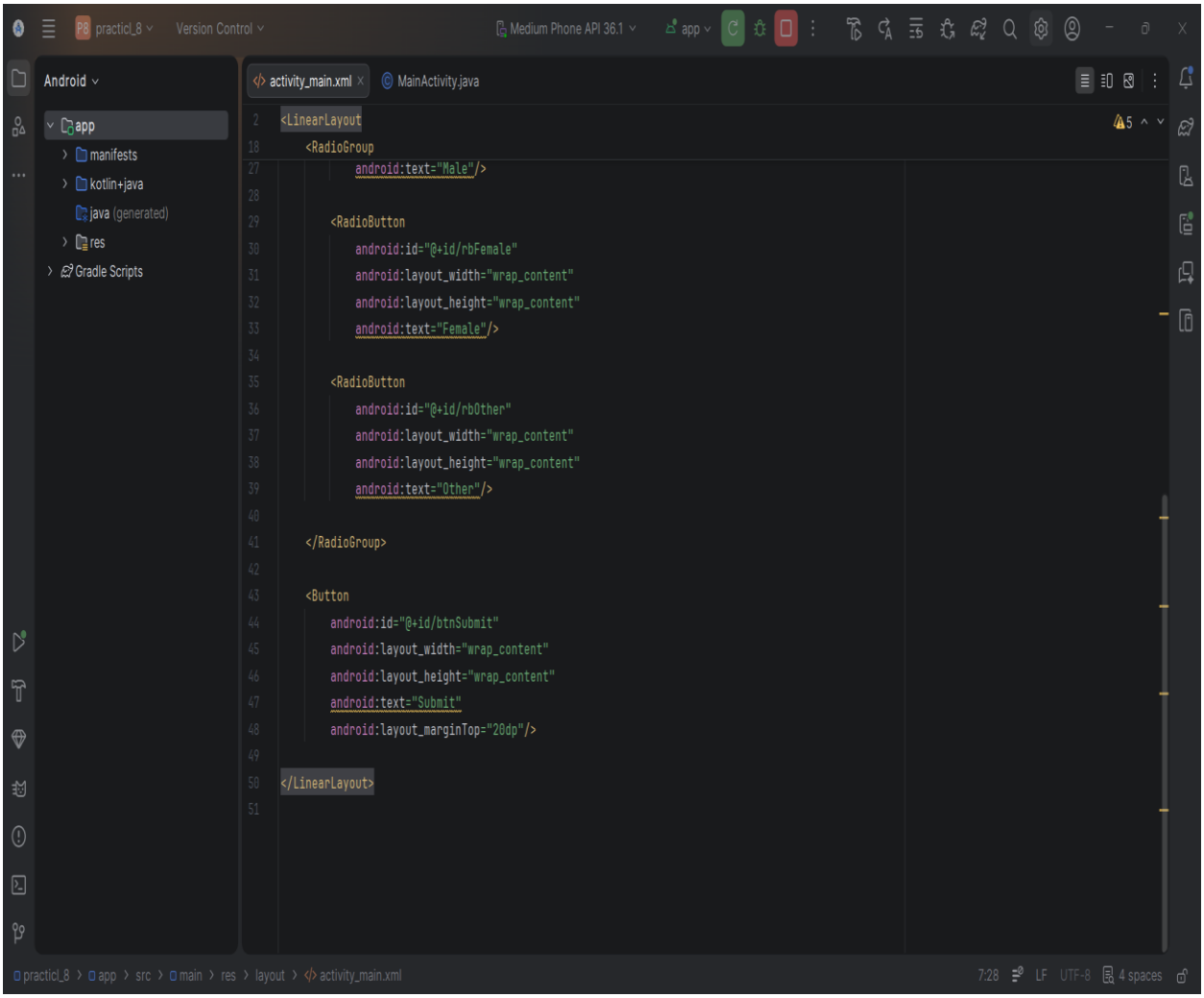
Step 2: Design XML Layout

- Open:

  res → layout → activity_main.xml

- **activity_main.xml**



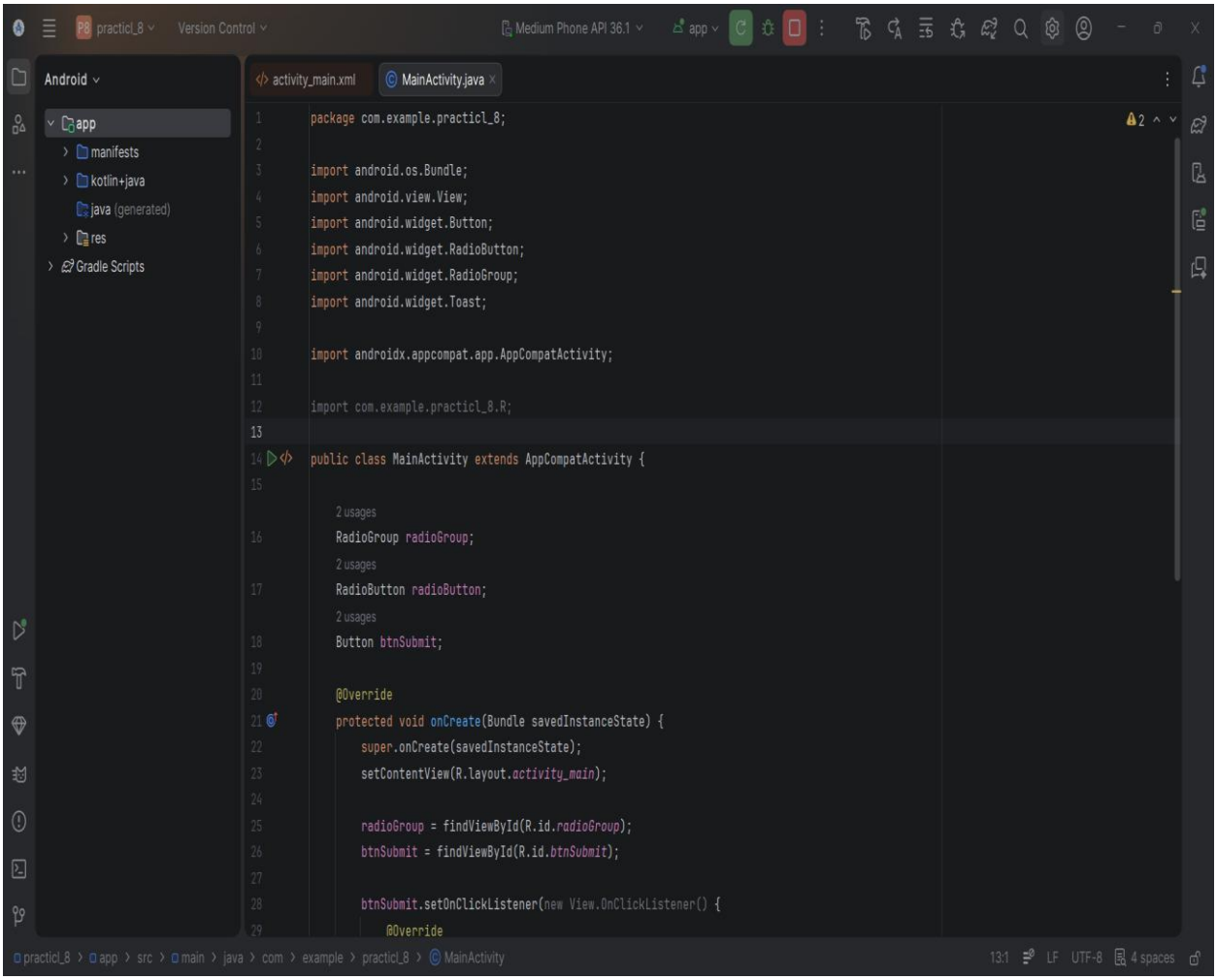


Step 3: Write Java Code

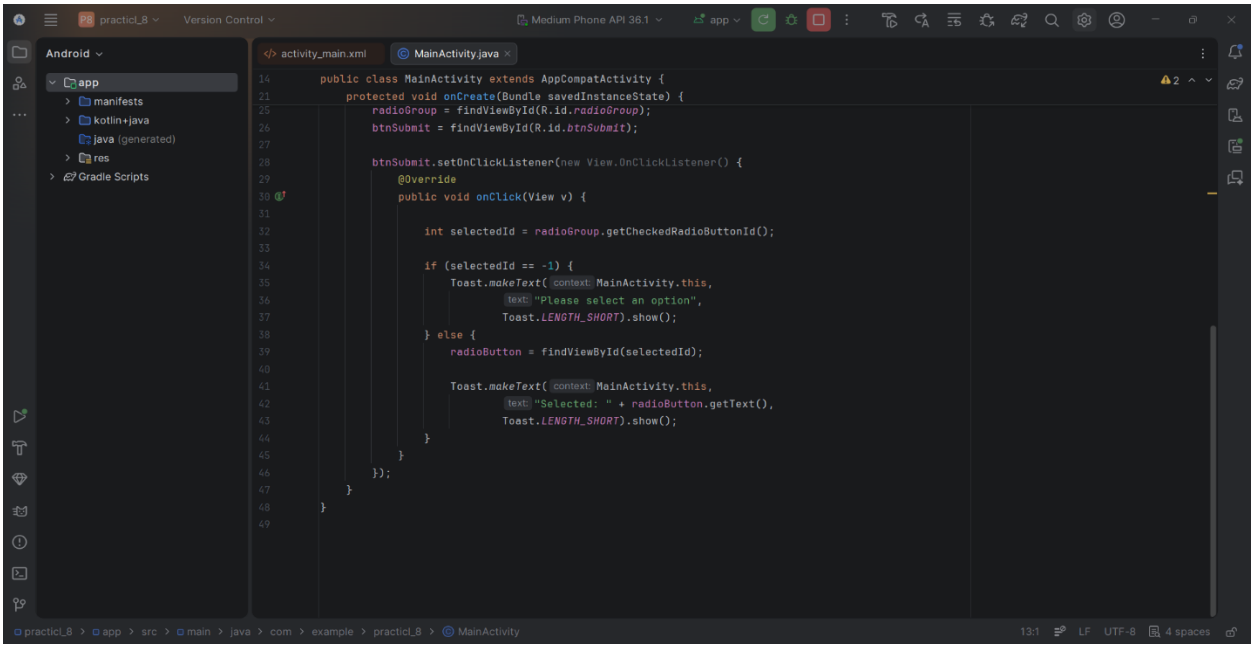
Open:

java → your package name → MainActivity.java

MainActivity.java

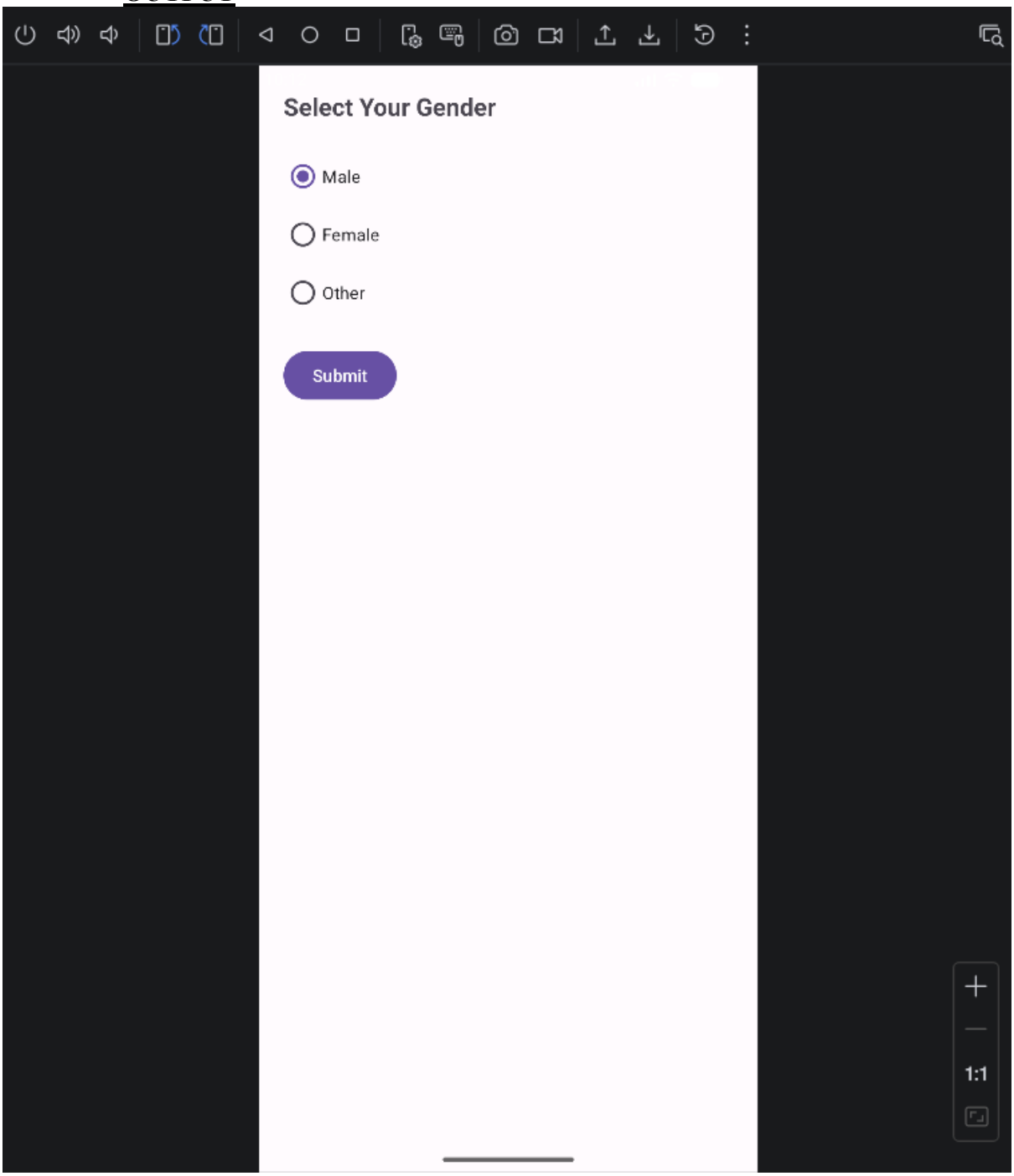


```java
package com.example.practicl_8;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.practicl_8.R;

public class MainActivity extends AppCompatActivity {

    RadioGroup radioGroup;
    RadioButton radioButton;
    Button btnSubmit;

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

        radioGroup = findViewById(R.id.radioGroup);
        btnSubmit = findViewById(R.id.btnSubmit);

        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
```



```java
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        radioGroup = findViewById(R.id.radioGroup);
        btnSubmit = findViewById(R.id.btnSubmit);

        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                int selectedId = radioGroup.getCheckedRadioButtonId();

                if (selectedId == -1) {
                    Toast.makeText( context: MainActivity.this,
                            text: "Please select an option",
                            Toast.LENGTH_SHORT).show();
                } else {
                    radioButton = findViewById(selectedId);

                    Toast.makeText( context: MainActivity.this,
                            text: "Selected: " + radioButton.getText(),
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

OUTPUT



PRACTICAL 9

9. Applications based on Intents and Intent Filters.

To create an Android application demonstrating:

- Explicit Intent
- Implicit Intent
- Custom Intent Filter

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project**
3. Select **Empty Activity**
4. Project Name: IntentPractical9
5. Language: **Java**
6. Click Finish

Step 2: Design activity_main.xml

Go to:

app → res → layout → activity_main.xml

Replace code with:

```
<?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="20dp">

   <Button
      android:id="@+id/btnExplicit"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Open Second Activity (Explicit Intent)" />

   <Button
```

```
    android:id="@+id/btnImplicit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Open Browser (Implicit Intent)"
    android:layout_marginTop="20dp"/>

  <Button
    android:id="@+id/btnCustom"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Custom Intent Filter"
    android:layout_marginTop="20dp"/>

</LinearLayout>
```

Step 3: Create Second Activity

Right Click:

```
java → your package name → New → Activity → Empty Activity
```

Name it:

```
SecondActivity
```

Step 4: Design activity_second.xml

Go to:

```
res → layout → activity_second.xml
```

Replace with:

```
<?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:gravity="center">

  <TextView
    android:id="@+id/txtMessage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:text="Message Here"/>
</LinearLayout>
```

Step 5: Write MainActivity.java Code

Open:

MainActivity.java

Replace with:

```
package com.example.intentpractical9;

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

public class MainActivity extends AppCompatActivity {

    Button btnExplicit, btnImplicit, btnCustom;

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

        btnExplicit = findViewById(R.id.btnExplicit);
        btnImplicit = findViewById(R.id.btnImplicit);
        btnCustom = findViewById(R.id.btnCustom);

        // Explicit Intent
        btnExplicit.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtra("msg", "Hello from Main Activity");
            startActivity(intent);
        });

        // Implicit Intent
        btnImplicit.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_VIEW);
            intent.setData(Uri.parse("https://www.google.com"));
            startActivity(intent);
        });

        // Custom Intent Filter
        btnCustom.setOnClickListener(v -> {
            Intent intent = new Intent("com.example.CUSTOM_ACTION");
            startActivity(intent);
```

```
        });
    }
}
```

Step 6: Write SecondActivity.java Code

Open:

```
SecondActivity.java
```

Replace with:

```
package com.example.intentpractical9;

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

public class SecondActivity extends AppCompatActivity {

    TextView txtMessage;

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

        txtMessage = findViewById(R.id.txtMessage);

        String message = getIntent().getStringExtra("msg");

        if (message != null) {
            txtMessage.setText(message);
        } else {
            txtMessage.setText("Opened using Custom Intent Filter");
        }
    }
}
```

Step 7: Configure AndroidManifest.xml

Open:

```
AndroidManifest.xml
```

Ensure this configuration:

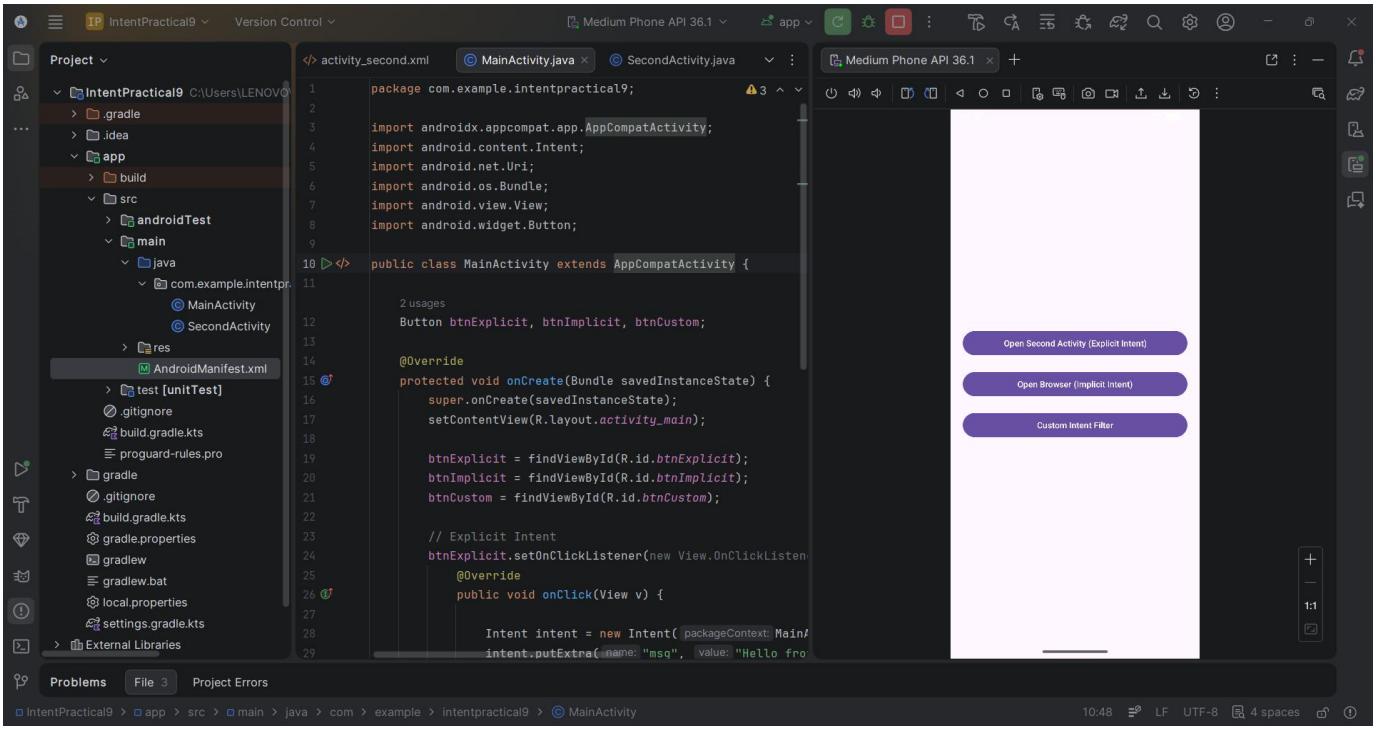
```
<application
    android:allowBackup="true"
```

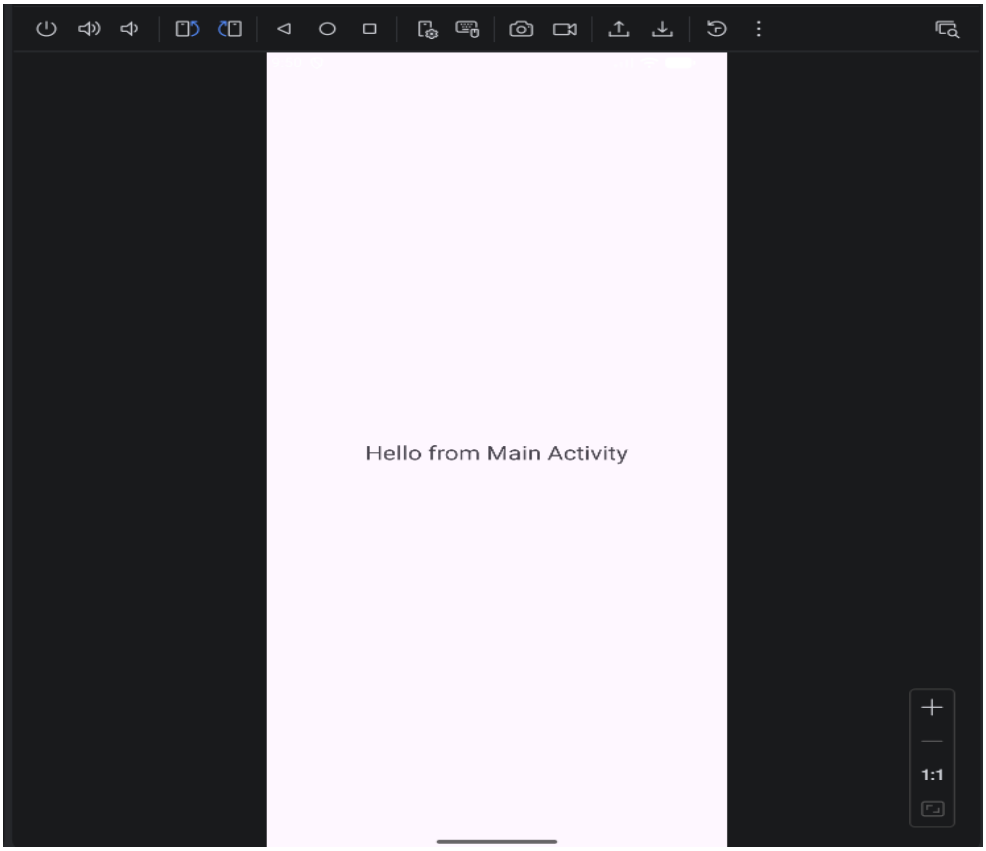
```
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.IntentPractical9">

    <activity
        android:name=".MainActivity"
        android:exported="true">

        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity
        android:name=".SecondActivity"
        android:exported="true">

        <intent-filter>
            <action android:name="com.example.CUSTOM_ACTION"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>

    </activity>

</application>
```

Step 8: Run the Application

1. Click ▶ Run
2. Select Emulator
3. App launches successfully

OUTPUT


IntentPractical9
Version Control
Medium Phone API 36.1
app
Project
activity_second.xml
MainActivity.java
SecondActivity.java
Medium Phone API 36.1
package com.example.intentpractical9;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
2 usages
Button btnExplicit, btnImplicit, btnCustom;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btnExplicit = findViewById(R.id.btnExplicit);
btnImplicit = findViewById(R.id.btnImplicit);
btnCustom = findViewById(R.id.btnCustom);
// Explicit Intent
btnExplicit.setOnClickListener(new View.OnClickListen
@Override
public void onClick(View v) {
Intent intent = new Intent(packageContext: MainA
intent.putExtra(name: "msg", value: "Hello fro
Open Second Activity (Explicit Intent)
Open Browser (Implicit Intent)
Custom Intent Filter
Problems
File 3
Project Errors


Hello from Main Activity
1:1

PRACTICAL 10

10.Applications based on Activities and services.

To create an Android application demonstrating **Activities and Services**.

Activities handle the **user interface**, while Services run **background tasks**.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project**
3. Select **Empty Activity**
4. Name: ActivityServiceDemo
5. Language: **Java**
6. Click **Finish**

Step 2: Design activity_main.xml

Go to:

res → layout → activity_main.xml

Add two buttons.

```
<?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="20dp">

    <Button
        android:id="@+id/btnStart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Service"/>
```

```
    <Button
        android:id="@+id/btnStop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Stop Service"
        android:layout_marginTop="20dp"/>

</LinearLayout>
```

Step 3: Create Service Class

Right click:

java → package name → New → Java Class

Name it:

MyService

Step 4: Write MyService.java Code

```
package com.example.activityservicedemo;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {

    @Override
    public void onCreate() {
        super.onCreate();
        Toast.makeText(this, "Service Created", Toast.LENGTH_SHORT).show();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {

        Toast.makeText(this, "Service Started", Toast.LENGTH_SHORT).show();
```

```
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, "Service Stopped", Toast.LENGTH_SHORT).show();
        super.onDestroy();
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Step 5: Write MainActivity.java

```
package com.example.activityservicedemo;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    Button btnStart, btnStop;

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

        btnStart = findViewById(R.id.btnStart);
        btnStop = findViewById(R.id.btnStop);

        btnStart.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, MyService.class);
            startService(intent);
        });

        btnStop.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, MyService.class);
            stopService(intent);
        });
```

```
    }
}
```

Step 6: Register Service in AndroidManifest.xml

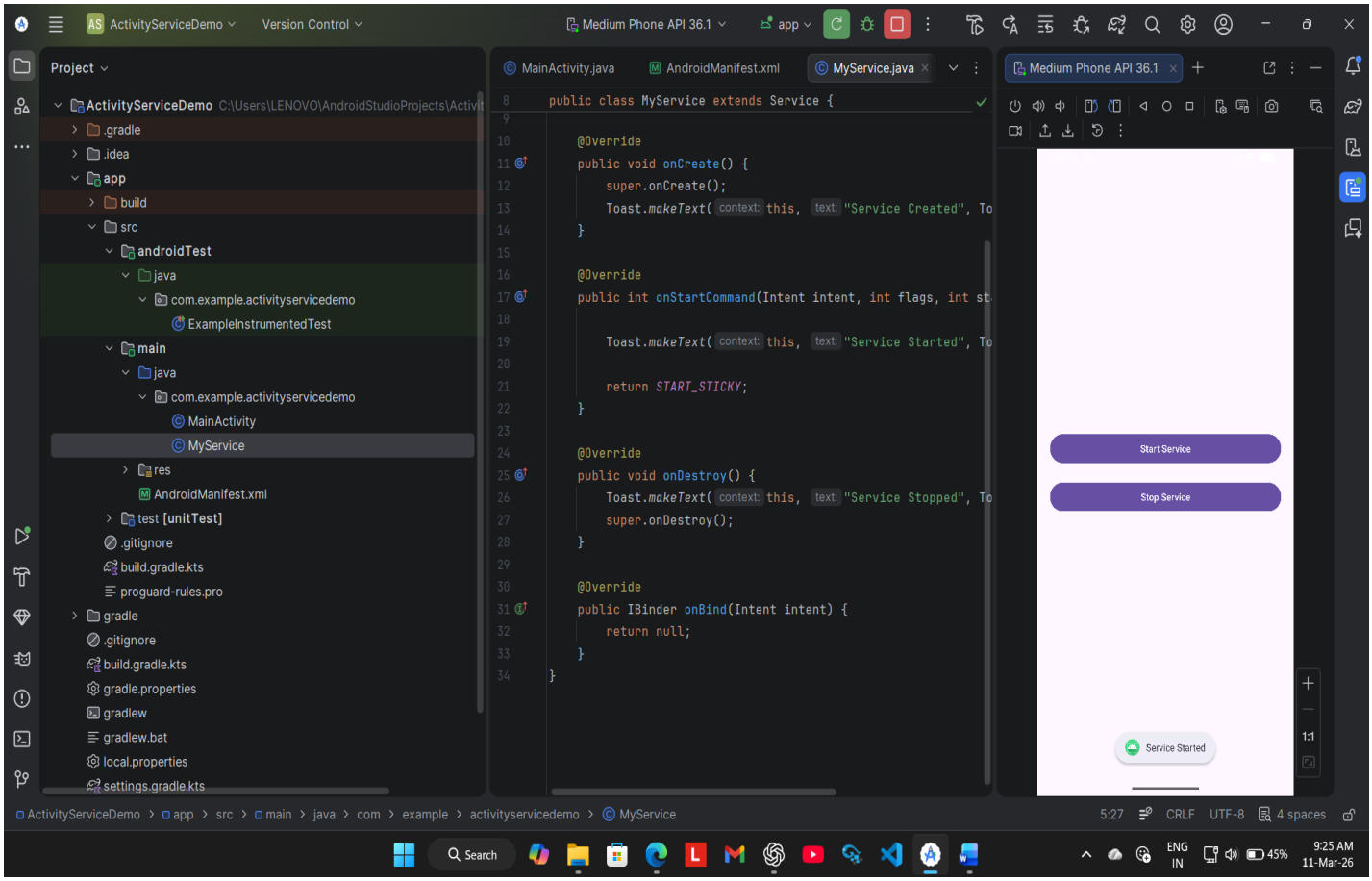
Inside <application> add:

```
<service
    android:name=".MyService"
    android:exported="false"/>
```

Output

- Clicking **Start Service** → Service starts in background
- Clicking **Stop Service** → Service stops

Toast messages appear.



PRACTICAL 11

11.Applications based on Action Bar.

To create an Android application demonstrating **Action Bar with menu options**.

Step 1: Create New Project

1. Open Android Studio
2. New Project → Empty Activity
3. Project Name: ActionBarDemo
4. Language: Java

Step 2: Create Menu Resource

Right Click:

res → New → Android Resource Directory

Select:

Resource Type → menu

Create file:

menu_main.xml

Step 3: Write Menu XML

res → menu → menu_main.xml

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

<menu xmlns:android="http://schemas.android.com/apk/res/android">

   <item
      android:id="@+id/menu_settings"
      android:title="Settings"/>

   <item
      android:id="@+id/menu_about"
      android:title="About"/>

</menu>
```

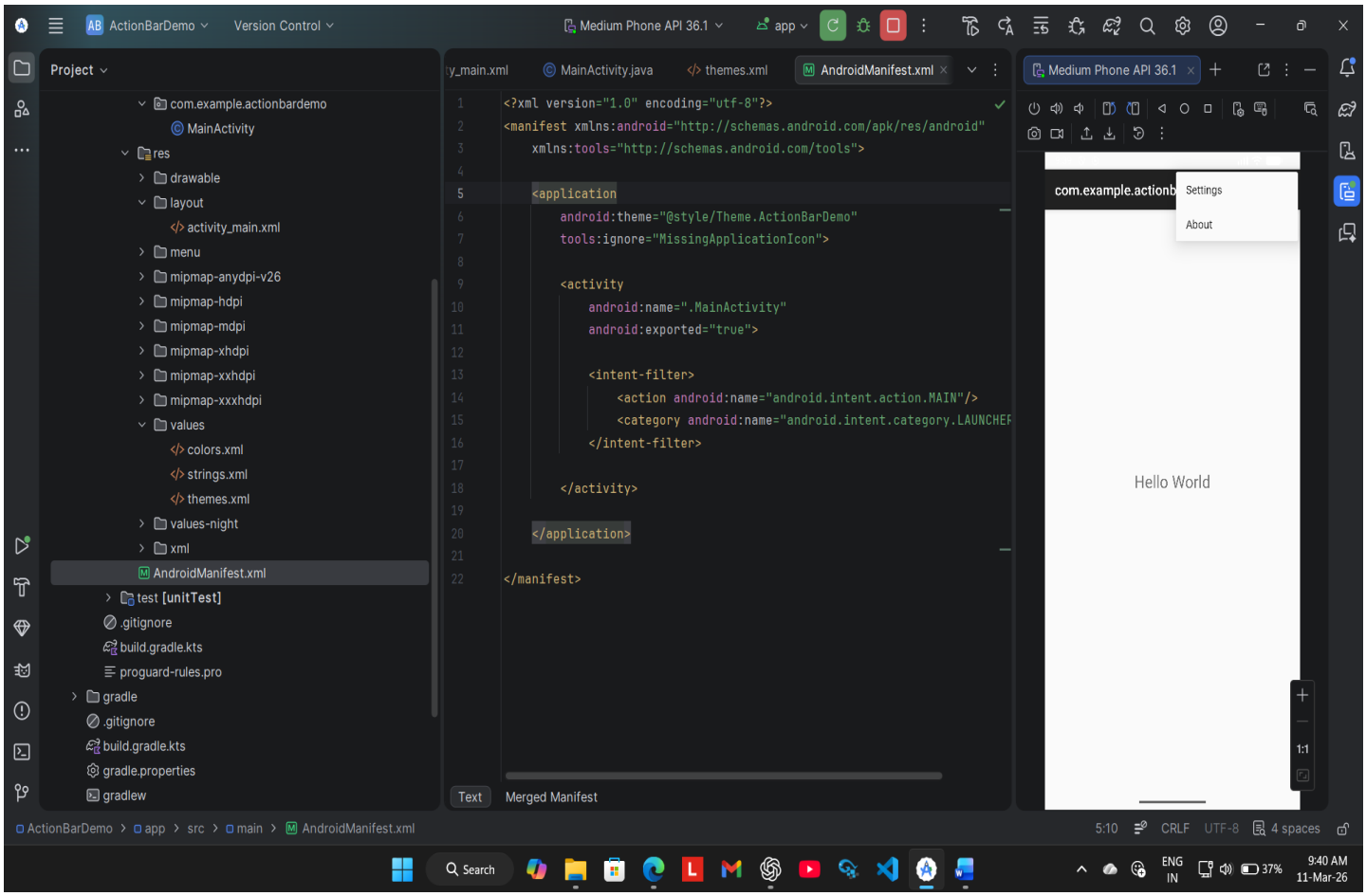
Step 4: Modify MainActivity.java

```
package com.example.actionbardemo;
```

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

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {

      getMenuInflater().inflate(R.menu.menu_main, menu);

      return true;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item) {

      int id = item.getItemId();

      if (id == R.id.menu_settings) {
         Toast.makeText(this, "Settings Selected", Toast.LENGTH_SHORT).show();
      }

      if (id == R.id.menu_about) {
         Toast.makeText(this, "About Selected", Toast.LENGTH_SHORT).show();
      }

      return true;
   }
}
```

Output

- Three dots menu appears in Action Bar
- Selecting **Settings** shows toast
- Selecting **About** shows toast


ActionBarDemo
Version Control
Medium Phone API 36.1
app
Project
com.example.actionbardemo
MainActivity
res
drawable
layout
activity_main.xml
menu
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
colors.xml
strings.xml
themes.xml
values-night
xml
AndroidManifest.xml
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro
gradle
.gitignore
build.gradle.kts
gradle.properties
gradlew
y_main.xml
MainActivity.java
themes.xml
AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">
<application
android:theme="@style/Theme.ActionBarDemo"
tools:ignore="MissingApplicationIcon">
<activity
android:name=".MainActivity"
android:exported="true">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER
</intent-filter>
</activity>
</application>
</manifest>
Text
Merged Manifest
Medium Phone API 36.1
com.example.actionb
Settings
About
Hello World
1:1
ActionBarDemo › app › src › main › AndroidManifest.xml
5:10
CRLF
UTF-8
4 spaces
Search
ENG
IN
37%
9:40 AM
11-Mar-26

PRACTICAL 12

12. Applications based on Option Menu.

To create an Android application demonstrating the **Options Menu (3 dots menu)**.

Steps

Step 1: Create Menu Folder

Right click res →
New → Android Resource Directory → select **menu**

Step 2: Create menu_main.xml

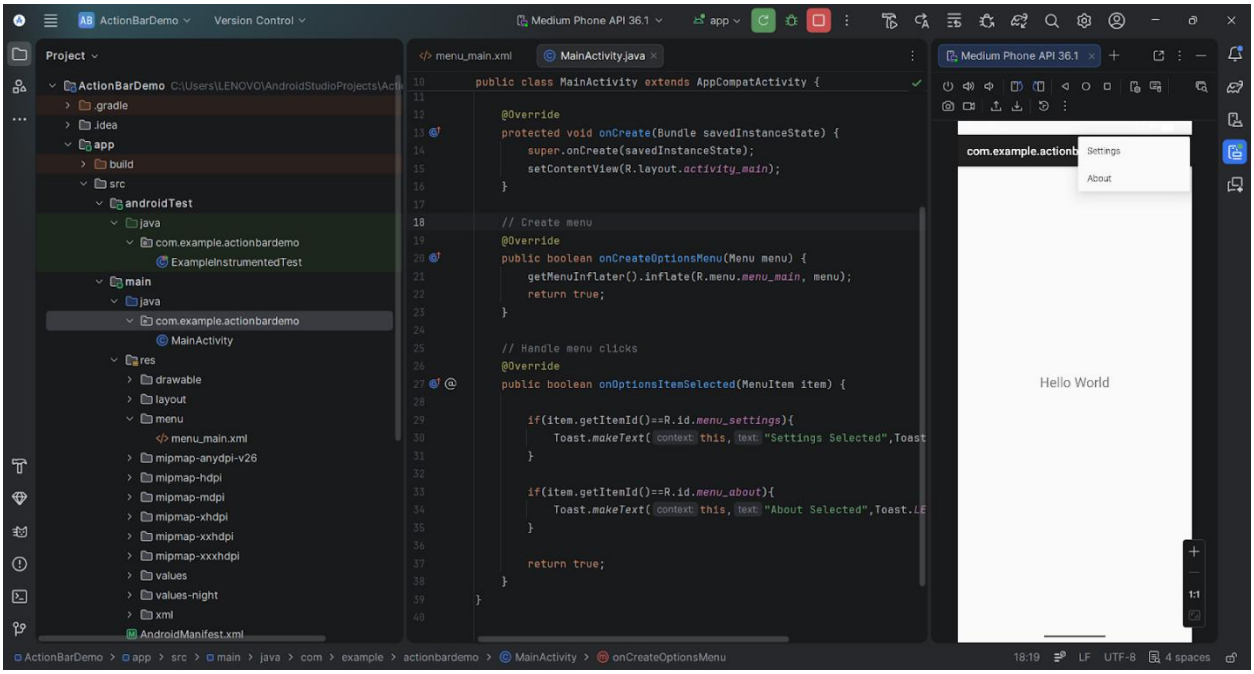
res/menu/menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item
     android:id="@+id/menu_settings"
     android:title="Settings"/>
  <item
     android:id="@+id/menu_about"
     android:title="About"/>
</menu>
```

Step 3: MainActivity.java

```
package com.example.actionbardemo;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
    }
    // Create menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    // Handle menu clicks
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId()==R.id.menu_settings){
            Toast.makeText(this,"Settings Selected",Toast.LENGTH_SHORT).show();
        }
        if(item.getItemId()==R.id.menu_about){
            Toast.makeText(this,"About Selected",Toast.LENGTH_SHORT).show();
        }
        return true;
    }
}
```


ActionBarDemo
Version Control
Medium Phone API 36.1
app
Project
menu_main.xml
MainActivity.java
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
// Create menu
@Override
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}
// Handle menu clicks
@Override
public boolean onOptionsItemSelected(MenuItem item) {
if(item.getItemId()==R.id.menu_settings){
Toast.makeText(context: this, text: "Settings Selected",Toast
}
if(item.getItemId()==R.id.menu_about){
Toast.makeText(context: this, text: "About Selected",Toast.LE
}
return true;
}
}
Settings
About
Hello World
ActionBarDemo > app > src > main > java > com > example > actionbardemo > MainActivity > onCreateOptionsMenu
18:19
LF
UTF-8
4 spaces

PRACTICAL 13

13.Applications based on Rating Bar.

To create an Android application that uses a **RatingBar** to collect user ratings.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project → Empty Activity**
3. Project Name: RatingBarDemo
4. Language: **Java**
5. Click **Finish**

Step 2: Design activity_main.xml

Go to:

res → layout → activity_main.xml

Replace with:

```
<?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:gravity="center"
  android:orientation="vertical"
  android:padding="20dp">
  <RatingBar
     android:id="@+id/ratingBar"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:numStars="5"
     android:stepSize="1.0"/>
  <Button
```

```
    android:id="@+id/btnSubmit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Submit Rating"
    android:layout_marginTop="20dp"/>
  <TextView
    android:id="@+id/txtResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Your Rating:"
    android:textSize="18sp"
    android:layout_marginTop="20dp"/>
</LinearLayout>
```

Step 3: Write MainActivity.java

Open:

MainActivity.java

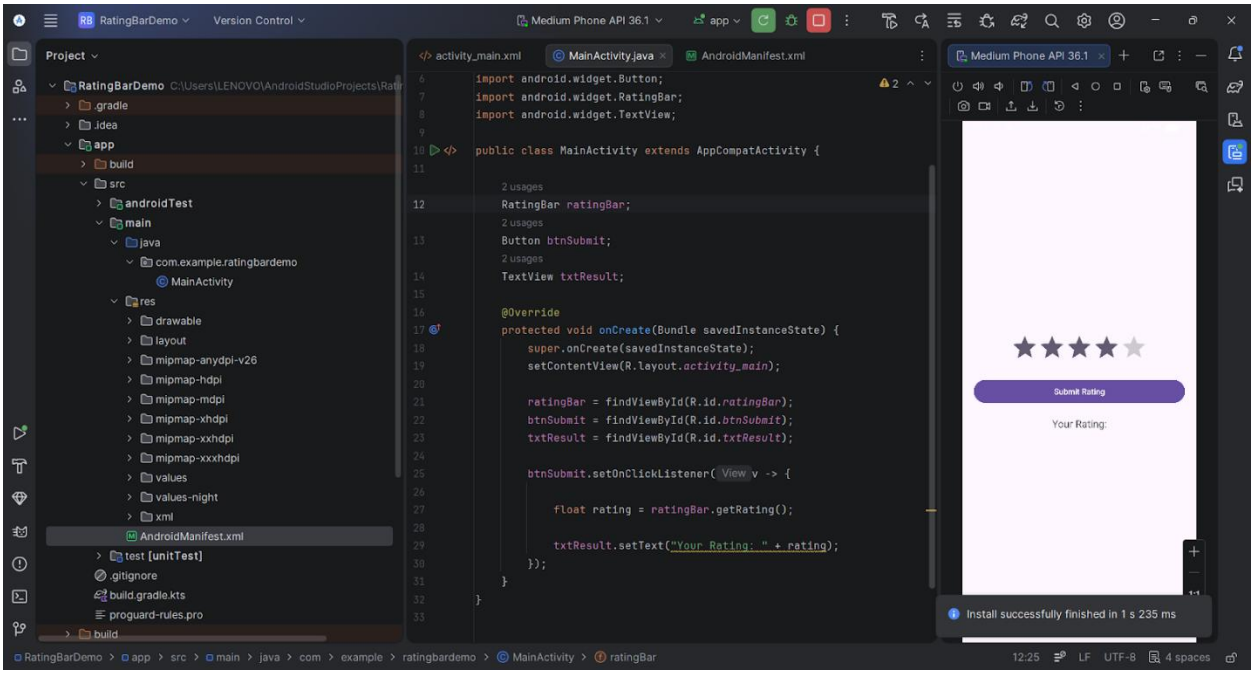
Replace with:

```
package com.example.ratingbardemo;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.RatingBar;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  RatingBar ratingBar;
  Button btnSubmit;
  TextView txtResult;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ratingBar = findViewById(R.id.ratingBar);
        btnSubmit = findViewById(R.id.btnSubmit);
        txtResult = findViewById(R.id.txtResult);
        btnSubmit.setOnClickListener(v -> {
            float rating = ratingBar.getRating();
            txtResult.setText("Your Rating: " + rating);
        });
    }
}
```

Step 4: AndroidManifest.xml

No special changes required.



PRACTICAL 14

14. Applications based on Media Player.

To create an Android application that **plays audio using MediaPlayer**.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project → Empty Activity**
3. Project Name: MediaPlayerDemo
4. Language: **Java**
5. Finish

Step 2: Add Audio File

1. Go to:

res → right click → New → Android Resource Directory

2. Select:

Resource type → raw

3. Click OK
4. Copy your audio file (e.g., music.mp3) into:

res/raw/music.mp3

Step 3: Design activity_main.xml

res/layout/activity_main.xml

```
<?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:gravity="center"
  android:orientation="vertical"
  android:padding="20dp">
  <Button
```

```
    android:id="@+id/btnPlay"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Play Music"/>
  <Button
    android:id="@+id/btnPause"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Pause Music"
    android:layout_marginTop="20dp"/>
  <Button
    android:id="@+id/btnStop"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Stop Music"
    android:layout_marginTop="20dp"/>
</LinearLayout>
```

Step 4: Write MainActivity.java

MainActivity.java

```
package com.example.mediaplayerdemo;
import androidx.appcompat.app.AppCompatActivity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
  Button btnPlay, btnPause, btnStop;
```

```
MediaPlayer mediaPlayer;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    btnPlay = findViewById(R.id.btnPlay);
    btnPause = findViewById(R.id.btnPause);
    btnStop = findViewById(R.id.btnStop);
    // Initialize MediaPlayer
    mediaPlayer = MediaPlayer.create(this, R.raw.music);
    // Play
    btnPlay.setOnClickListener(v -> {
        if (!mediaPlayer.isPlaying()) {
            mediaPlayer.start();
        }
    });
    // Pause
    btnPause.setOnClickListener(v -> {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.pause();
        }
    });
    // Stop
    btnStop.setOnClickListener(v -> {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
            mediaPlayer = MediaPlayer.create(this, R.raw.music); // reinitialize
```

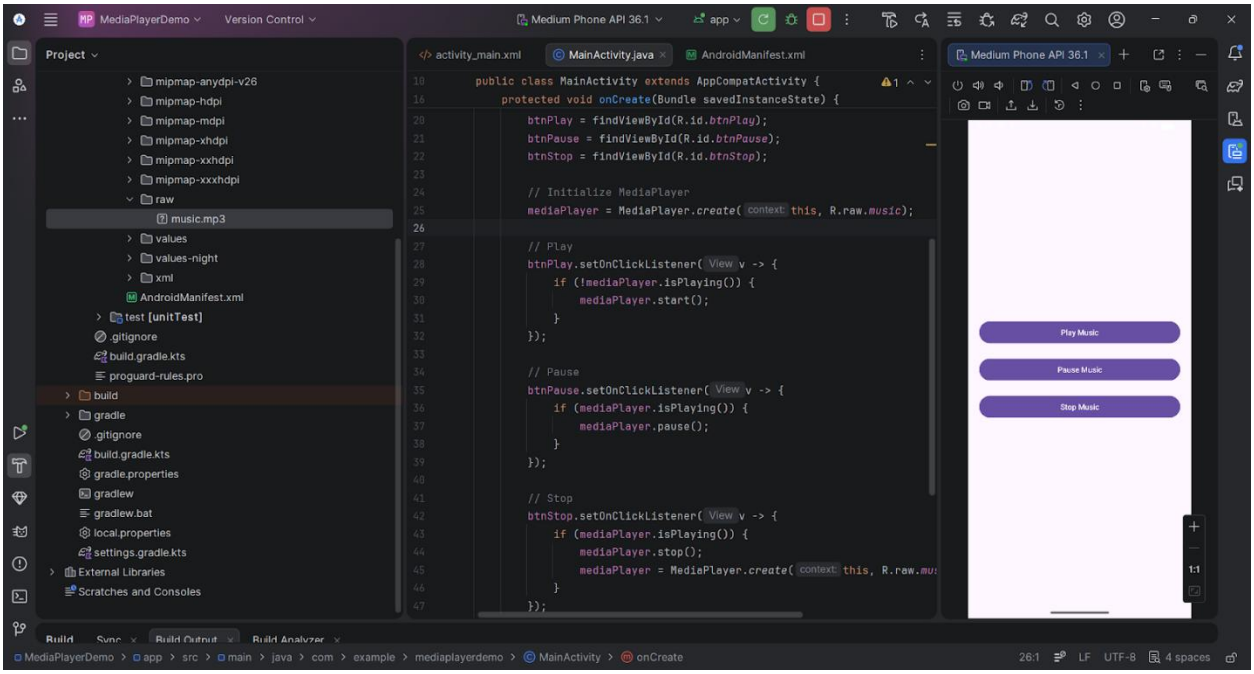
```
        }
    });
  }
  @Override
  protected void onDestroy() {
    super.onDestroy();
    mediaPlayer.release();
  }
}
```

Step 5: AndroidManifest.xml

No special permission required for **local audio**.

Just ensure:

```
<application
  android:theme="@style/Theme.MediaPlayerDemo">
```



PRACTICAL 15

15. Applications based on Content Providers.

Steps

Step 1: Create Project

- Open Android Studio
- New Project → Empty Activity
- Name: ContentProviderDemo

Step 2: Add Permission

AndroidManifest.xml

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

Step 3: Design Layout

activity_main.xml

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

  <Button
    android:id="@+id/btnLoad"
    android:text="Load Contacts"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

  <TextView
    android:id="@+id/txtContacts"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

Step 4: Write Code

MainActivity.java

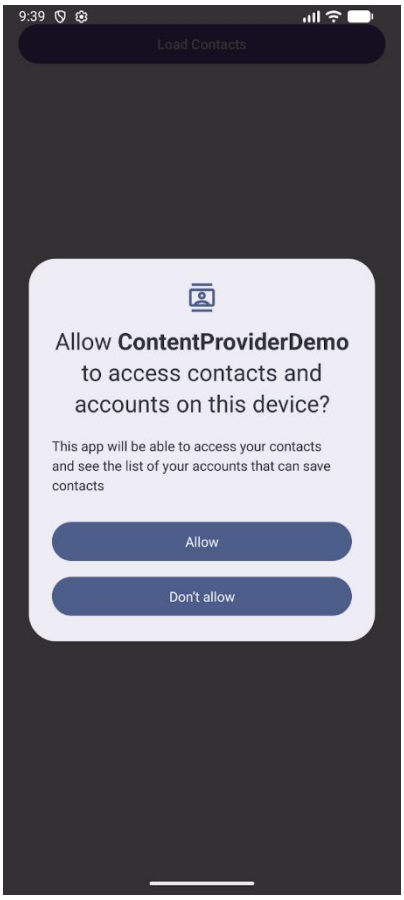
```
package com.example.contentproviderdemo;

import android.Manifest;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.Button;
import android.widget.TextView;

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

```
import androidx.core.app.ActivityCompat;

public class MainActivity extends AppCompatActivity {

    TextView txt;
    Button btn;

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

        txt = findViewById(R.id.txtContacts);
        btn = findViewById(R.id.btnLoad);

        btn.setOnClickListener(v -> loadContacts());
    }

    private void loadContacts() {

        // Check Permission
        if (ActivityCompat.checkSelfPermission(this,
                Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED) {

            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.READ_CONTACTS}, 1);
            return;
        }

        Cursor c = getContentResolver().query(
                ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                null, null, null, null);

        StringBuilder data = new StringBuilder();

        if (c != null) {
            while (c.moveToNext()) {
                String name = c.getString(
                        c.getColumnIndexOrThrow(
                                ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
                data.append(name).append("\n");
            }
            c.close();
        }
```

```
        txt.setText(data.toString());
    }
}
```

Step 5: Run App

- Click button → Contacts displayed



PRACTICAL 16

16. Application Based on Camera

Steps

Step 1: Create Project

Name: CameraDemo

Step 2: Design Layout

```
<Button
  android:id="@+id/btnCamera"
  android:text="Open Camera"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"/>
```

Step 3: Write Code

MainActivity.java

```
Intent i = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(i, 1);
```

Step 4: Run App

- Click button → Camera opens
- Capture image

PRACTICAL 17

17.Application Based on Accessing Location

Aim

To create an Android application that **retrieves and displays the current location (Latitude & Longitude)**.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project → Empty Activity**
3. Name: LocationDemo
4. Language: **Java**
5. Click **Finish**

Step 2: Add Permissions

Open AndroidManifest.xml and add:

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

Step 3: Design Layout

res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/main"
  android:orientation="vertical"
  android:gravity="center"
  android:padding="20dp"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
     android:id="@+id/btnLocation"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="Get Location"/>
  <TextView
     android:id="@+id/txtLocation"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Location will appear here"
     android:textSize="18sp"
     android:layout_marginTop="20dp"/>
</LinearLayout>
```

Step 4: Write MainActivity.java

Replace full code:

```
package com.example.locationdemo;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.content.pm.PackageManager;
```

```
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
  Button btnLocation;
  TextView txtLocation;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     btnLocation = findViewById(R.id.btnLocation);
     txtLocation = findViewById(R.id.txtLocation);
     btnLocation.setOnClickListener(v -> getLocation());
  }
  private void getLocation() {
     LocationManager lm = (LocationManager) getSystemService(LOCATION_SERVICE);
     // Permission Check
     if (ActivityCompat.checkSelfPermission(this,
           Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this,
              new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
        return;
     }

     Location location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
     if (location != null) {
        double lat = location.getLatitude();
        double lon = location.getLongitude();

        txtLocation.setText("Latitude: " + lat + "\nLongitude: " + lon);
     } else {
        txtLocation.setText("Location not available");
     }
  }
}
```

Step 5: Enable Location in Emulator

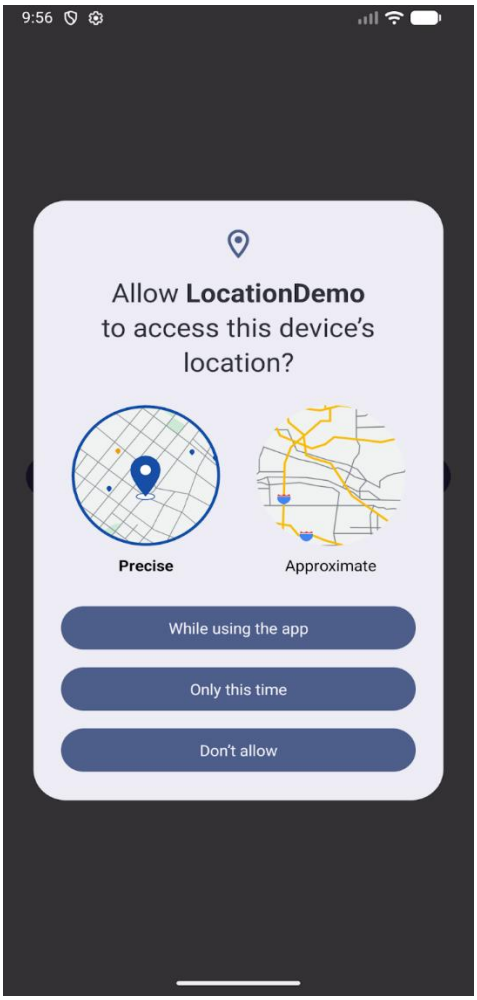
Very Important

- Open Emulator
- Go to **Settings → Location → ON**
- OR manually set location from emulator controls

Step 6: Run Application

1. Click ▶ Run

2. Click **Get Location**
3. Allow permission
•



PRACTICAL 18

18.Application Based on Sensors (Android – Java)

Aim

To create an Android application that **detects device motion using Accelerometer Sensor**.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project → Empty Activity**
3. Name: SensorDemo
4. Language: **Java**
5. Click **Finish**

Step 2: Design Layout

res/layout/activity_main.xml

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

  <TextView
    android:id="@+id/txtSensor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sensor Data"
    android:textSize="20sp"/>

</LinearLayout>
```

Step 3: Write MainActivity.java

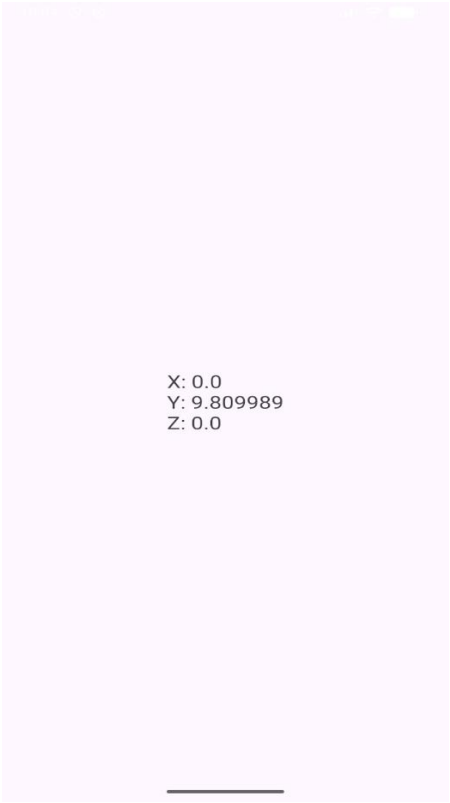
Replace full code:

```
package com.example.sensordemo;

import androidx.appcompat.app.AppCompatActivity;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements SensorEventListener {

  SensorManager sensorManager;
  Sensor accelerometer;
  TextView txtSensor;
```

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

        txtSensor = findViewById(R.id.txtSensor);

        // Initialize Sensor Manager
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        // Get Accelerometer Sensor
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        // Register Listener
        sensorManager.registerListener(this, accelerometer,
    SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {

        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        txtSensor.setText("X: " + x + "\nY: " + y + "\nZ: " + z);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
}
```

Step 4: Run Application

1. Click ▶ Run
2. Move your device / emulator


X: 0.0
Y: 9.809989
Z: 0.0

PRACTICAL 19

19.Application Based on Animations (Android – Java)

Aim

To create an Android application that **applies animation to a view (Button/Image)**.

Step 1: Create New Project

1. Open **Android Studio**
2. Click **New Project → Empty Activity**
3. Name: AnimationDemo
4. Language: **Java**
5. Click **Finish**

Step 2: Create Animation Folder

1. Right click res
2. Click **New → Android Resource Directory**
3. Select:

Resource Type → anim

4. Click OK

Step 3: Create Animation File

res/anim/rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
  android:duration="1000"
  android:fromDegrees="0"
  android:toDegrees="360"
  android:pivotX="50%"
  android:pivotY="50%"/>
```

Step 4: Design Layout

res/layout/activity_main.xml

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

  <Button
    android:id="@+id/btnAnim"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start Animation"/>
</LinearLayout>
```

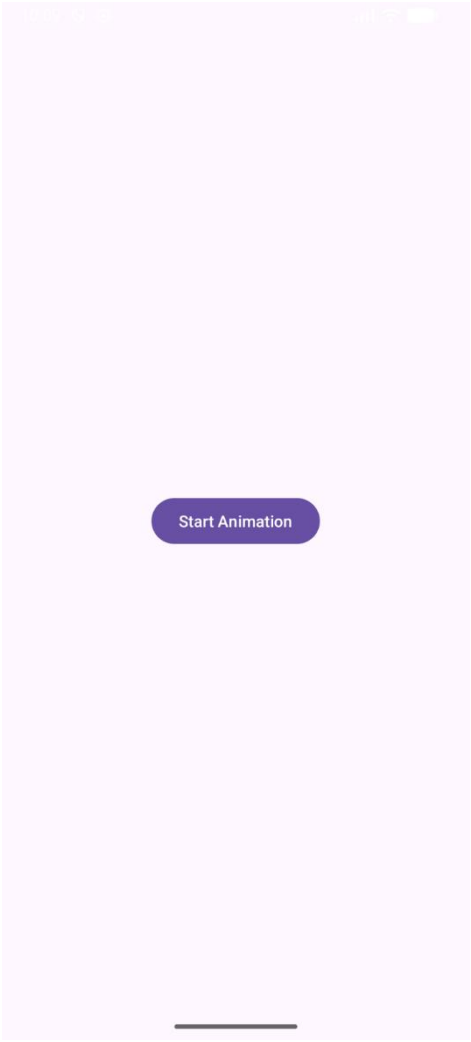
Step 5: Write MainActivity.java

Replace full code:

```
package com.example.animationdemo;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

  Button btnAnim;

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

     btnAnim = findViewById(R.id.btnAnim);

     btnAnim.setOnClickListener(v -> {

        Animation animation = AnimationUtils.loadAnimation(this, R.anim.rotate);
        btnAnim.startAnimation(animation);

     });
  }
}
```

Step 6: Run Application

1. Click ▶ Run
2. Click **Start Animation**


Start Animation