# **ASSIGNMENT NO.4**

# **GAURAV\_BODKHE\_2124UCEM1041**

## ***(Long Click App)***

## **1. Introduction**

## In this program, we will create an interactive Android app using an Activity class to handle various user events. The main screen will allow users to trigger events such as long presses, button clicks, and other interactions. Each event will display a corresponding message or action, demonstrating how user interactions can be captured and responded to in an Android application. This project is useful for learning about event handling and user engagement in mobile app development.

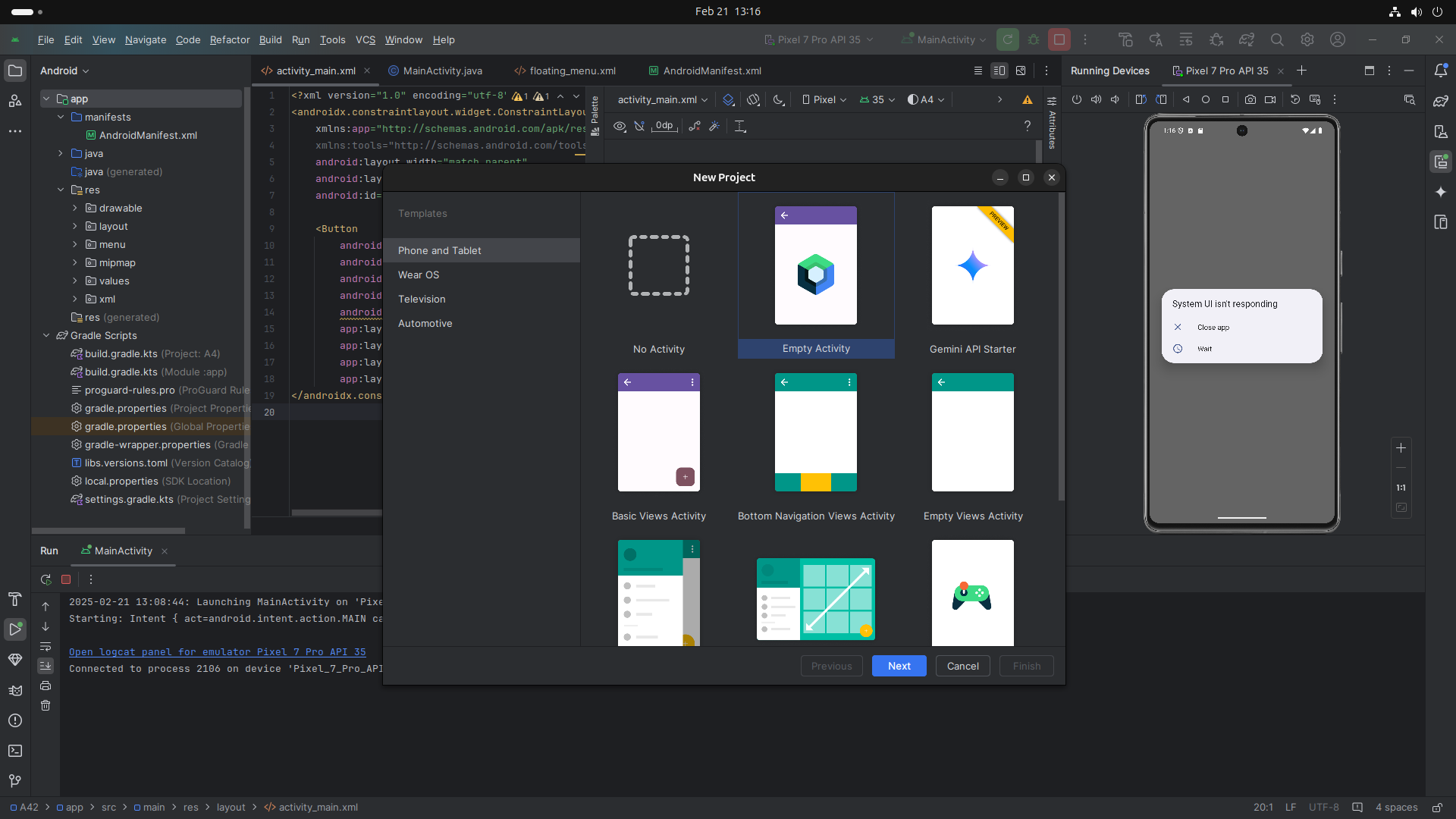
## **2. Tools & Technologies Used**

* Android Studio
* Java
* Emulator

## **3. Procedure & Steps**

### **Step 1: Create a New Project**

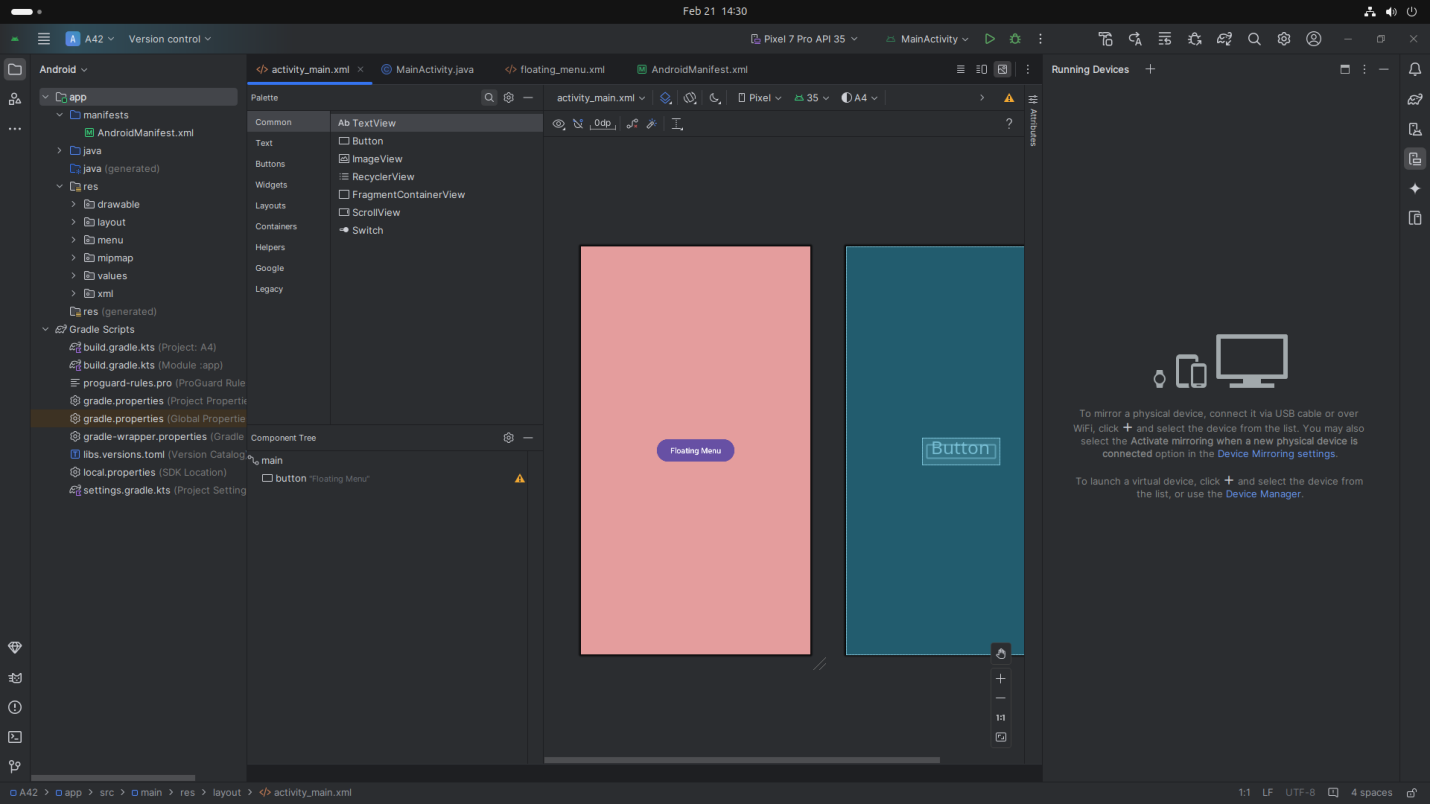
* Open Android Studio and create a new project.
* Choose an Empty Views Activity template.
* Set the project name and package name of your Application
* Select the programming language (Java).
* **Screenshot:**



### **Step 2: Designing the UI**

* Open activity\_main.xml and design the layout using XML.
* Add UI comwponents such as floating\_menu Button, etc.

**Screenshot:**

****

**Step 3: Writing the Code**

* Open Activity\_main.xml
* Implement functionality such as floating\_menu button,
* Use necessary Android components like Intents, RecyclerView, Fragments, etc
* **XML**

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

### *<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"*

### *xmlns:app="http://schemas.android.com/apk/res-auto"*

### *xmlns:tools="http://schemas.android.com/tools"*

### *android:id="@+id/main"*

### *android:layout\_width="match\_parent"*

### *android:layout\_height="match\_parent"*

### *android:background="#E49D9D">*

### *<Button*

### *android:id="@+id/button"*

### *android:layout\_width="wrap\_content"*

### *android:layout\_height="wrap\_content"*

### *android:layout\_centerInParent="true"*

### *android:text="Floating Menu"*

### *app:layout\_constraintBottom\_toBottomOf="parent"*

### *app:layout\_constraintEnd\_toEndOf="parent"*

### *app:layout\_constraintStart\_toStartOf="parent"*

### *app:layout\_constraintTop\_toTopOf="parent" />*

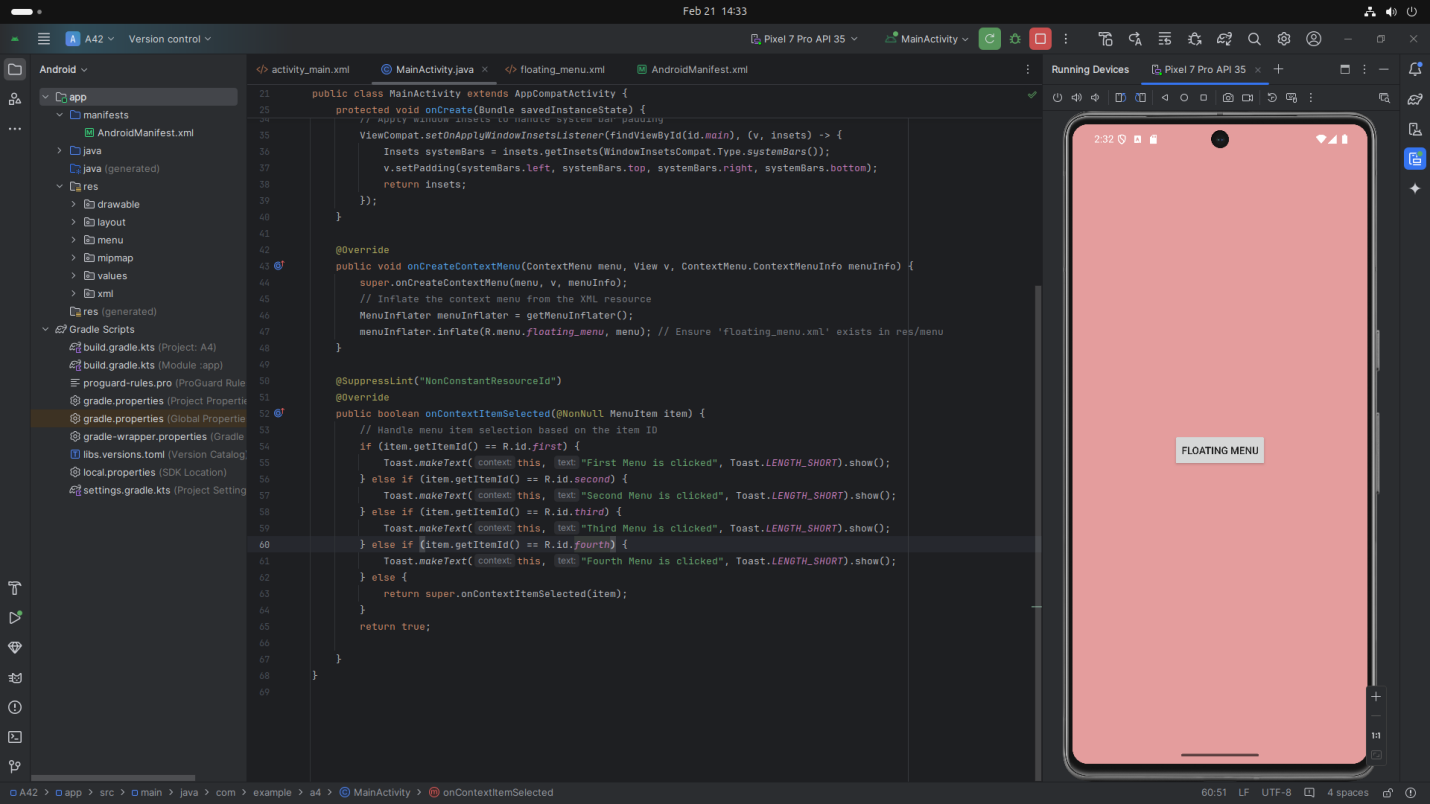
### *</androidx.constraintlayout.widget.ConstraintLayout>*

### **Step 4: Writing the Code (Backend code for long Press in JAVA)**

* Open MainActivity.java
* Implement functionality such as, when user **Long Press click** on button then display corresponding message(ex. First menu is clicked)

**Screenshot:**

## **Screenshot from 2025-02-21 14-32-58**



* **Code(** MainActivity.java**)**

### import static com.example.a4.R.\*;

### import android.annotation.SuppressLint;

### import android.os.Bundle;

### import android.view.ContextMenu;

### import android.view.Menu;

### import android.view.MenuInflater;

### import android.view.MenuItem;

### import android.view.View;

### import android.widget.Button;

### import android.widget.Toast;

### import androidx.annotation.NonNull;

### import androidx.appcompat.app.AppCompatActivity;

### import androidx.core.graphics.Insets;

### import androidx.core.view.ViewCompat;

### import androidx.core.view.WindowInsetsCompat;

### public class MainActivity extends AppCompatActivity {

### private Button btn;

### @Override

### protected void onCreate(Bundle savedInstanceState) {

### super.onCreate(savedInstanceState);

### // Set the layout for the activity

### setContentView(R.layout.activity\_main);

### // Initialize the button and register it for context menu

### btn = findViewById(id.button);

### registerForContextMenu(btn);

### // Apply window insets to handle system bar padding

### ViewCompat.setOnApplyWindowInsetsListener(findViewById(id.main), (v, insets) -> {

### Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

### v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

### return insets;

### });

### }

### @Override

### public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {

### super.onCreateContextMenu(menu, v, menuInfo);

### // Inflate the context menu from the XML resource

### MenuInflater menuInflater = getMenuInflater();

### menuInflater.inflate(R.menu.floating\_menu, menu); // Ensure 'floating\_menu.xml' exists in res/menu

### }

### @SuppressLint("NonConstantResourceId")

### @Override

### public boolean onContextItemSelected(@NonNull MenuItem item) {

### // Handle menu item selection based on the item ID

### if (item.getItemId() == R.id.first) {

### Toast.makeText(this, "First Menu is clicked", Toast.LENGTH\_SHORT).show();

### } else if (item.getItemId() == R.id.second) {

### Toast.makeText(this, "Second Menu is clicked", Toast.LENGTH\_SHORT).show();

### } else if (item.getItemId() == R.id.third) {

### Toast.makeText(this, "Third Menu is clicked", Toast.LENGTH\_SHORT).show();

### } else if (item.getItemId() == R.id.fourth) {

### Toast.makeText(this, "Fourth Menu is clicked", Toast.LENGTH\_SHORT).show();

### } else {

### return super.onContextItemSelected(item);

### }

### return true;

### }

### }

## **4. Procedure & Steps(For floating\_menu file)**

### **Step 1: Create a New Floating\_menu file**

* Select App section on Main Interface of App
* Choose RES (Right click)
* Clicked on New File and Select Android Resource File
* Give File Name And Resource Type
* **Screenshot:**



### **Step 2: Designing the UI**

* Open floating\_menu.xml and design the layout using XML.
* Add UI comwponents such as first, second, third,fourth floating\_menu Buttons, etc.
* **Screenshot:**

### **Screenshot from 2025-02-21 14-38-05**

**Code:(**floating\_menu.xml**)**

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

### *<item*

### *android:id="@+id/first"*

### *android:title="First" />*

### *<item*

### *android:id="@+id/second"*

### *android:title="Second" />*

### *<item*

### *android:id="@+id/third"*

### *android:title="Third" />*

### *<item*

### *android:id="@+id/fourth"*

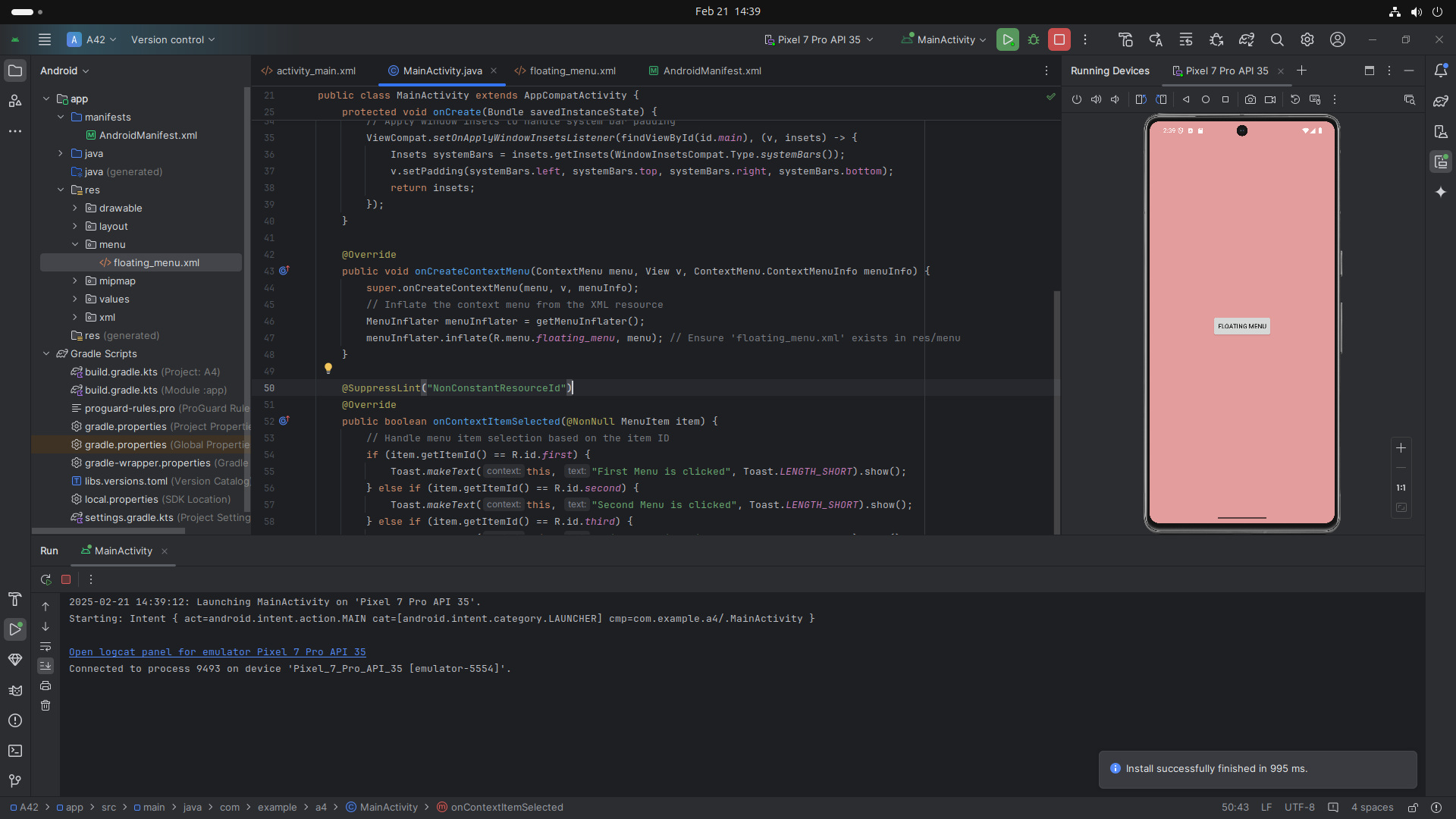
### *android:title="Fourth" />*

### *</menu>*

### **Step 5: Running the Application on Emulator**

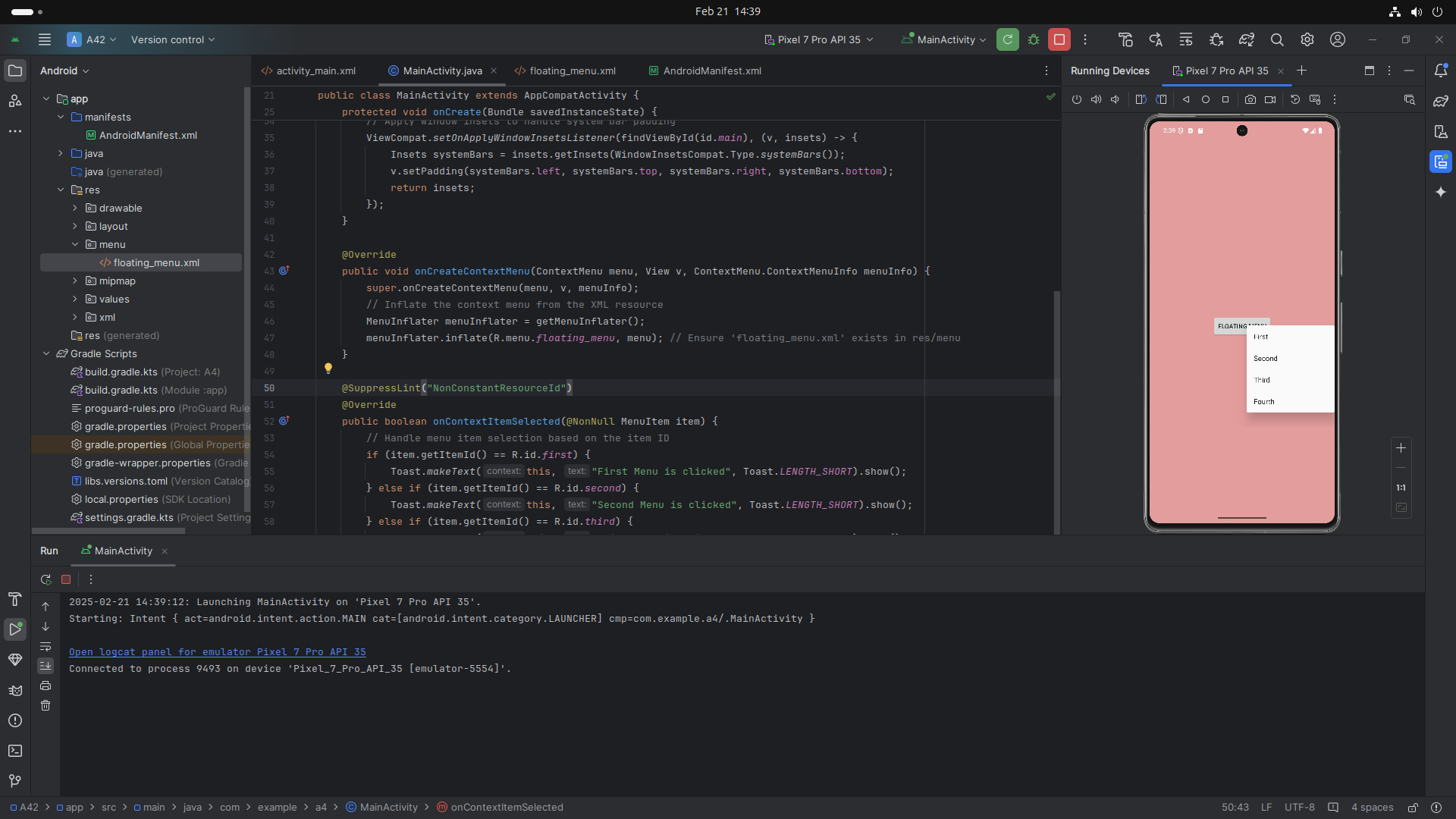
* Click on the **Run** button in Android Studio.
* Select the emulator and launch the app.

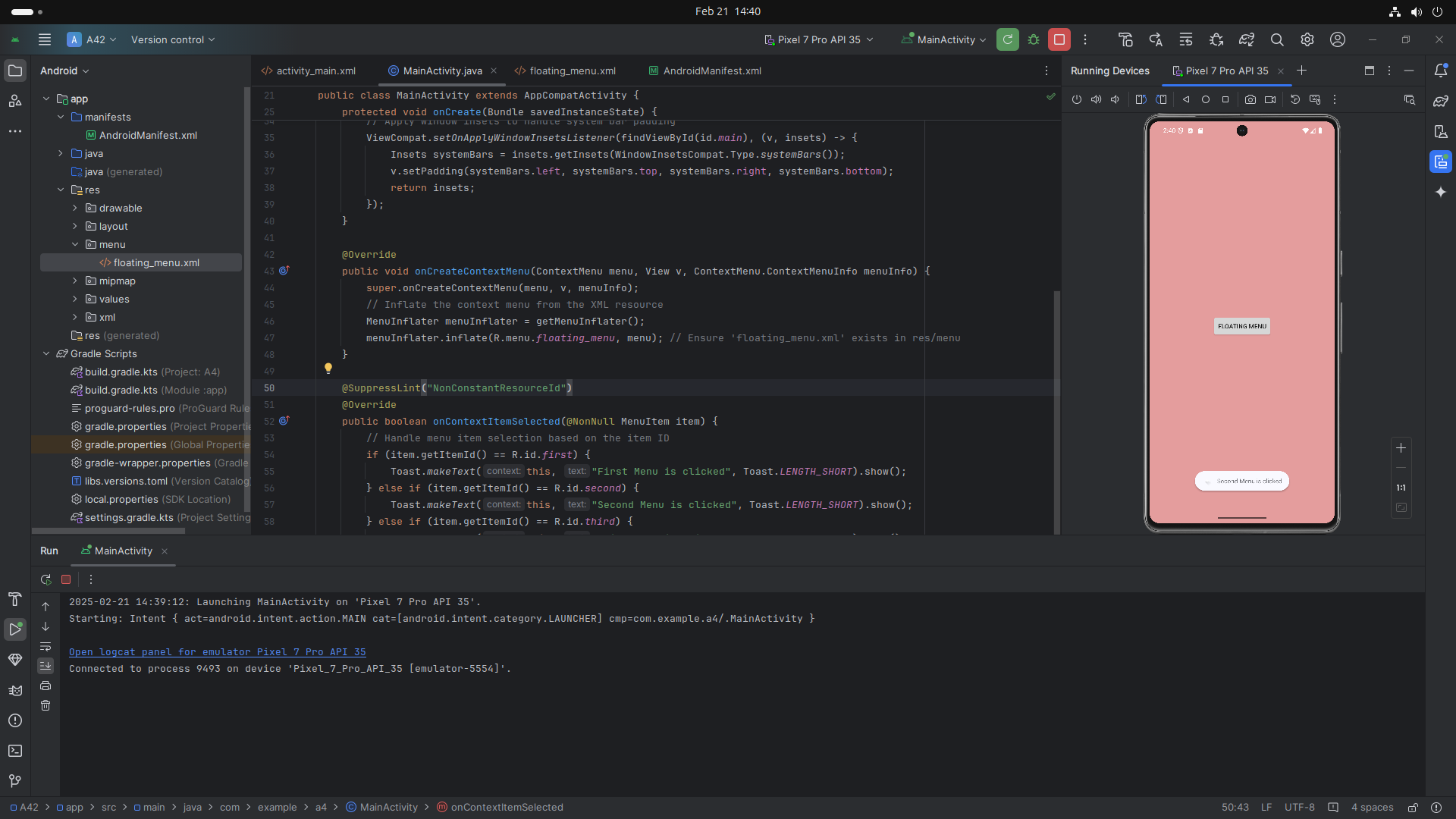
**Screenshot:**

****

### **Step 5: Testing & Output**

* Test different functionalities of the app.
* Capture the output results.
* **Screenshot:**

****

****

***(Swipe\_Refresh\_App)***

## **1. Introduction**

## In this program we create an Android app with a main screen that responds to different user actions, like swiping or refreshing. The app will use an Activity class to manage these actions. For example, when a user swipes down, the app will show a "Refreshing..." message or perform a refresh action.The goal is to demonstrate how the app reacts to events, such as swipes or gestures, with simple responses or actions, making the app interactive and user-friendly.

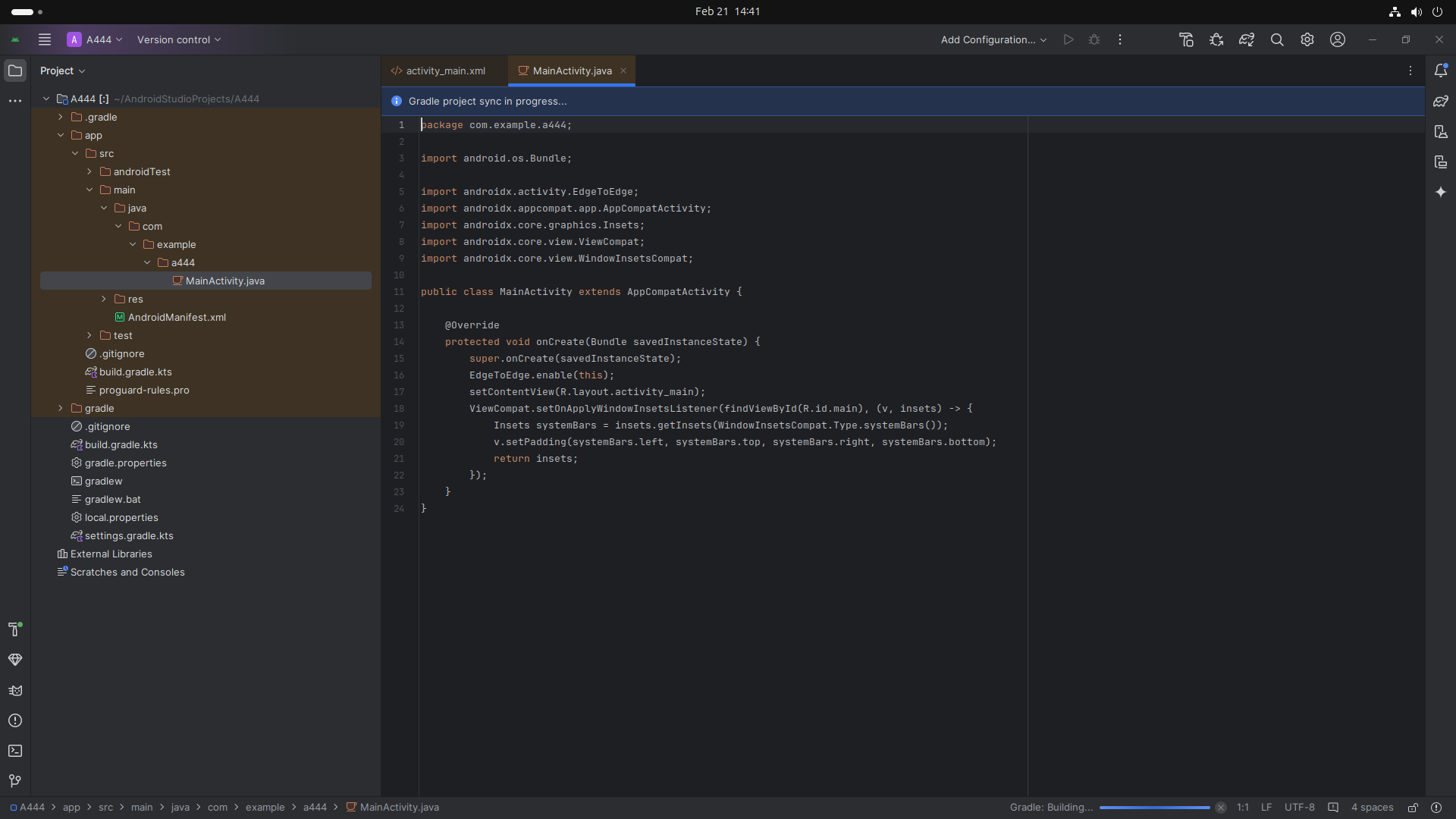
## **2. Tools & Technologies Used**

* Android Studio
* Java
* Emulator

## **3. Procedure & Steps**

### **Step 1: Create a New Project**

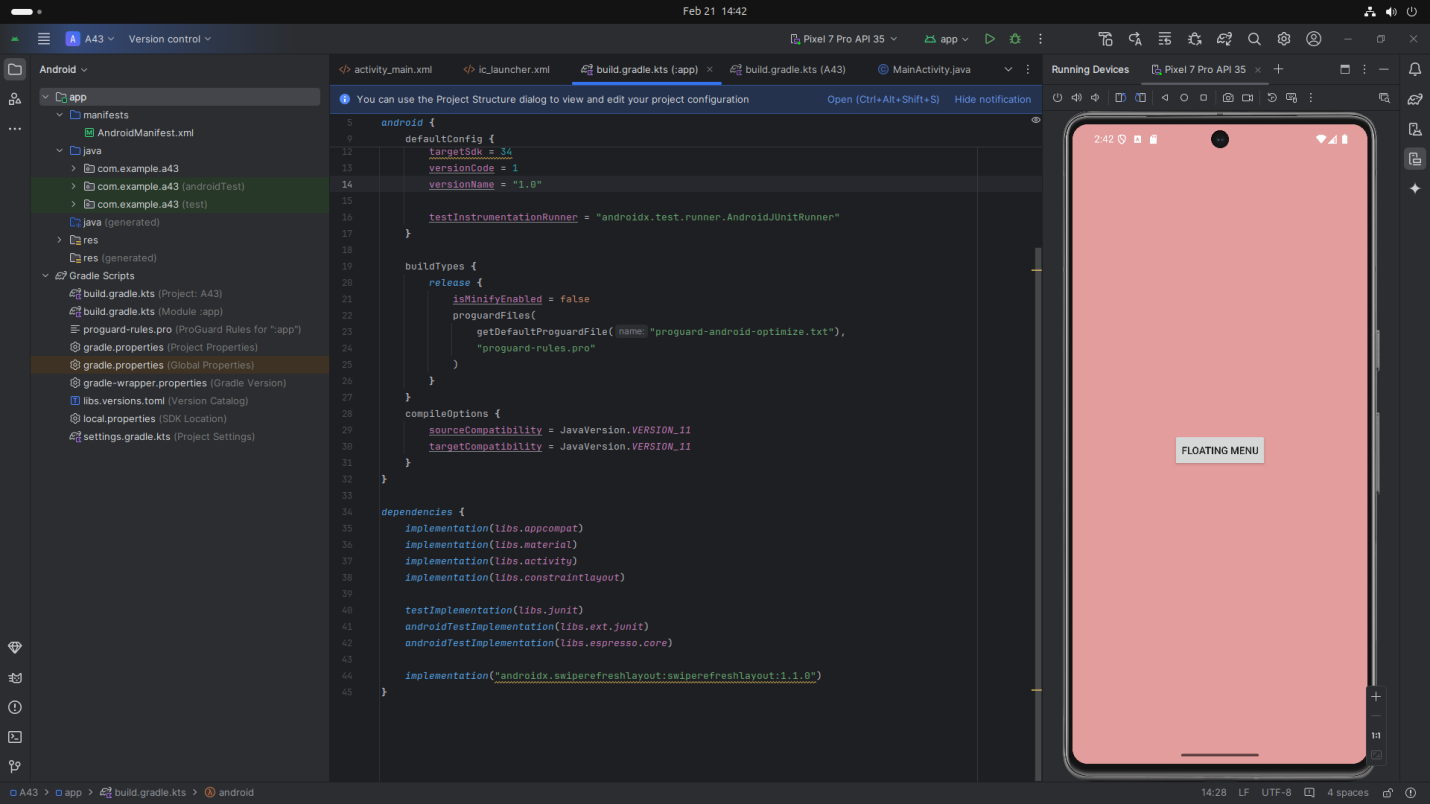
* Open Android Studio and create a new project.
* Choose an Empty Views Activity template.
* Set the project name and package name of your Application
* Select the programming language (Java).
* **Screenshot:**



### **Step 2: Implementing The Library (for swipe\_refresh\_layout)**

* Open The Gradle Scripts and select build.gradle.kts
* Implement the library for swipe\_refresh\_layout

**Screenshot:**

****

### **Code : (Kotlin)**

### plugins {

### alias(libs.plugins.android.application)

### }

### android {

### namespace = "com.example.swiperefreshapp"

### compileSdk = 35

### defaultConfig {

### applicationId = "com.example.swiperefreshapp"

### minSdk = 26// Changed to a reasonable minimum SDK version

### targetSdk = 34

### versionCode = 1

### versionName = "1.0"

### testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"

### }

### buildTypes {

### release {

### isMinifyEnabled = false

### proguardFiles(

### getDefaultProguardFile("proguard-android-optimize.txt"),

### "proguard-rules.pro"

### )

### }

### }

### compileOptions {

### sourceCompatibility = JavaVersion.VERSION\_11

### targetCompatibility = JavaVersion.VERSION\_11

### }

### }

### dependencies {

### implementation(libs.appcompat)

### implementation(libs.material)

### implementation(libs.activity)

### implementation(libs.constraintlayout)

### testImplementation(libs.junit)

### androidTestImplementation(libs.ext.junit)

### androidTestImplementation(libs.espresso.core)

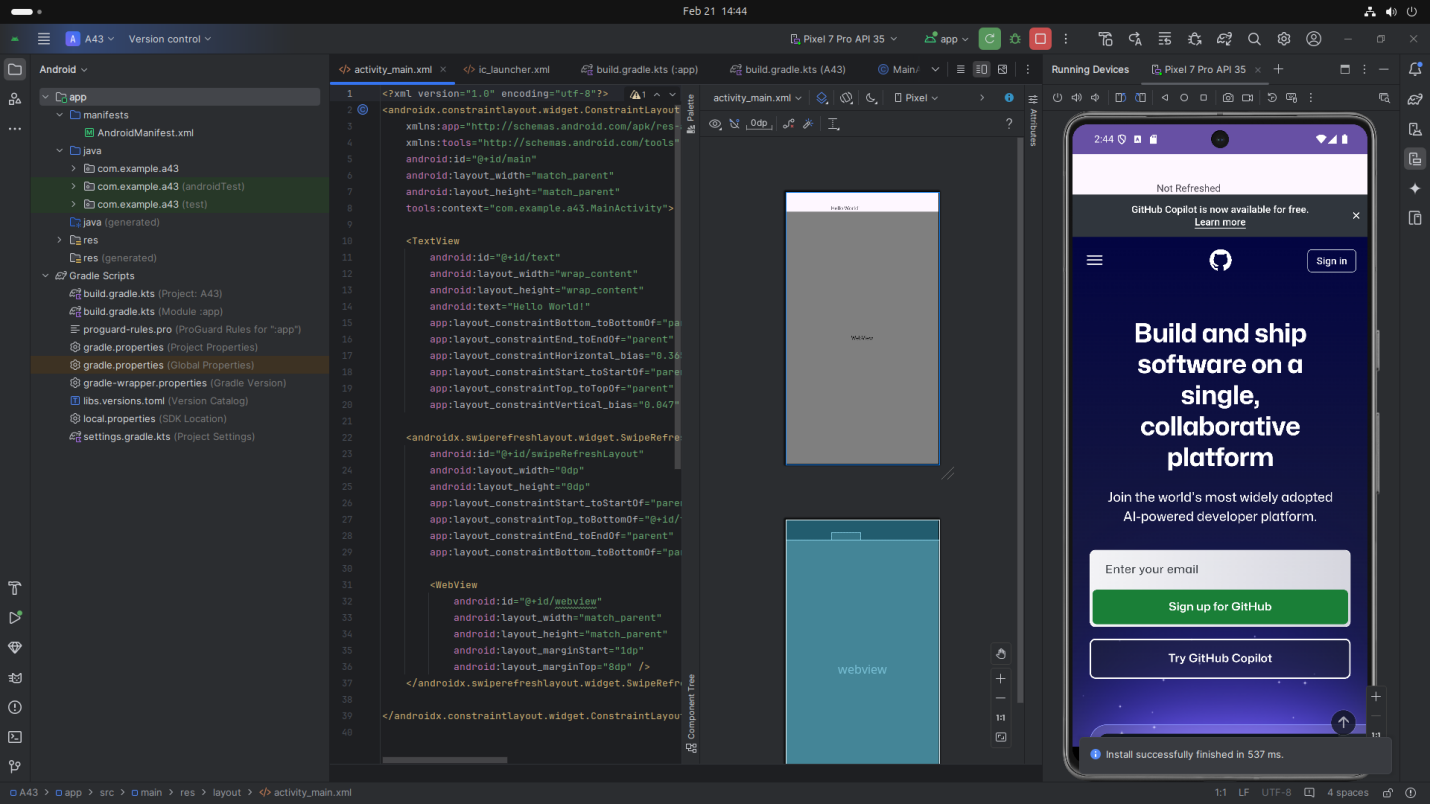
### **implementation("androidx.swiperefreshlayout:swiperefreshlayout:1.1.0")**

### }

### **Step 3: Designing the UI**

* Open activity\_main.xml and design the layout using XML.
* Add UI comwponents such as TextView, Webview etc.

**Screenshot:**

****

**Code:**

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

*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"*

*xmlns:app="http://schemas.android.com/apk/res-auto"*

*xmlns:tools="http://schemas.android.com/tools"*

*android:id="@+id/main"*

*android:layout\_width="match\_parent"*

*android:layout\_height="match\_parent"*

*tools:context="com.example.a43.MainActivity">*

*<TextView*

*android:id="@+id/text"*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content"*

*android:text="Hello World!"*

*app:layout\_constraintBottom\_toBottomOf="parent"*

*app:layout\_constraintEnd\_toEndOf="parent"*

*app:layout\_constraintHorizontal\_bias="0.365"*

*app:layout\_constraintStart\_toStartOf="parent"*

*app:layout\_constraintTop\_toTopOf="parent"*

*app:layout\_constraintVertical\_bias="0.047" />*

*<androidx.swiperefreshlayout.widget.SwipeRefreshLayout*

*android:id="@+id/swipeRefreshLayout"*

*android:layout\_width="0dp"*

*android:layout\_height="0dp"*

*app:layout\_constraintStart\_toStartOf="parent"*

*app:layout\_constraintTop\_toBottomOf="@+id/text"*

*app:layout\_constraintEnd\_toEndOf="parent"*

*app:layout\_constraintBottom\_toBottomOf="parent">*

*<WebView*

*android:id="@+id/webview"*

*android:layout\_width="match\_parent"*

*android:layout\_height="match\_parent"*

*android:layout\_marginStart="1dp"*

*android:layout\_marginTop="8dp" />*

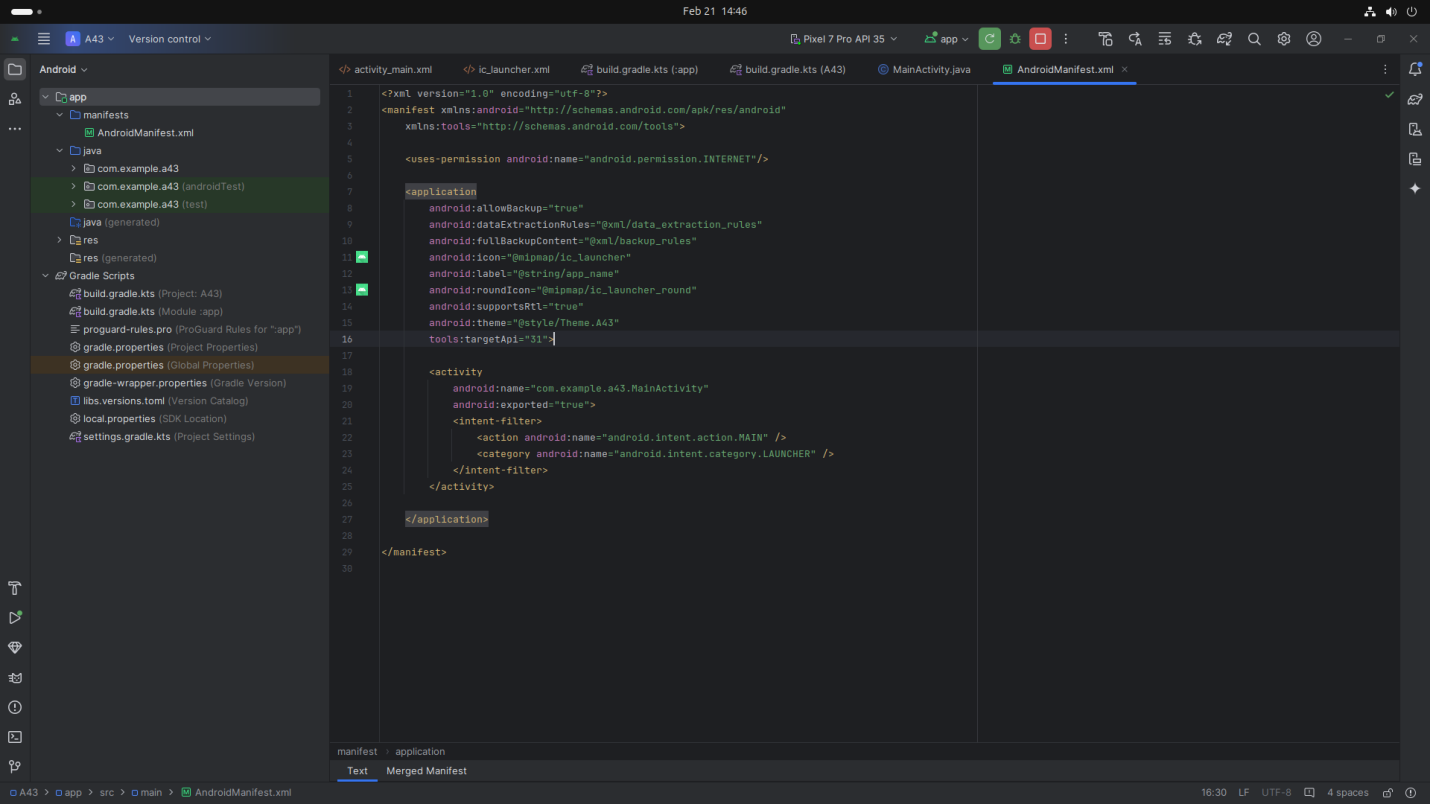
*</androidx.swiperefreshlayout.widget.SwipeRefreshLayout>*

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

**Step 4: Writing the Code (For Uses Permission)**

* Open AndroidManifest.xml
* Implement functionality get Uses Permission Android-Permission for Swipe Refresh layout

**Screenshot:**



* **Code(** 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">*

*<uses-permission android:name="android.permission.INTERNET"/>*

*<application*

*android:allowBackup="true"*

*android:dataExtractionRules="@xml/data\_extraction\_rules"*

*android:fullBackupContent="@xml/backup\_rules"*

*android:icon="@mipmap/ic\_launcher"*

*android:label="@string/app\_name"*

*android:roundIcon="@mipmap/ic\_launcher\_round"*

*android:supportsRtl="true"*

*android:theme="@style/Theme.A43"*

*tools:targetApi="31">*

*<activity*

*android:name="com.example.a43.MainActivity"*

*android:exported="true">*

*<intent-filter>*

*<action android:name="android.intent.action.MAIN" />*

*<category android:name="android.intent.category.LAUNCHER" />*

*</intent-filter>*

*</activity>*

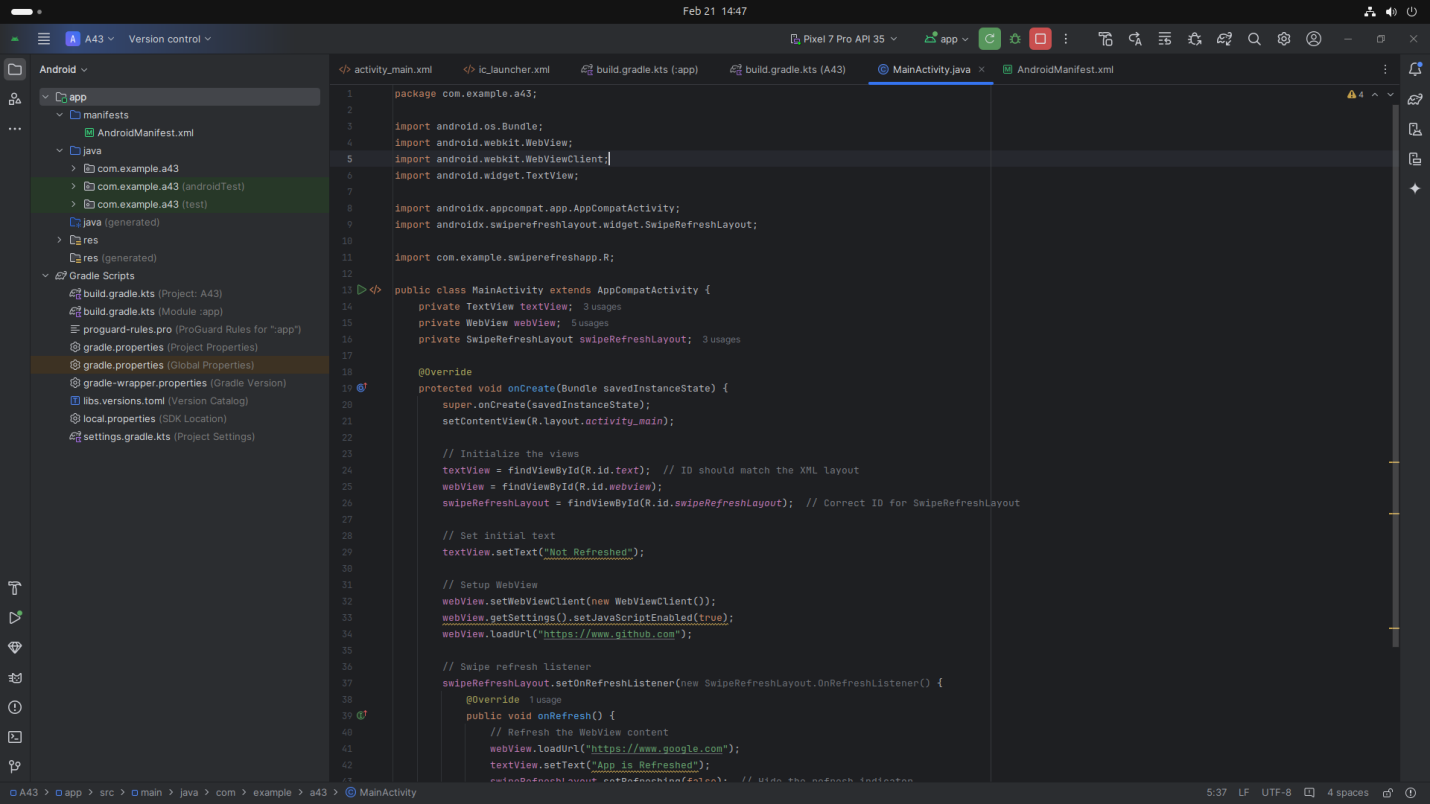
*</application>*

*</manifest>*

### **Step 5: Writing the Backend Code (java)**

* Open MainActivity.java
* Implement functionality such as By pressing up to down App Get Refreshed and Show Refresh Page

**Screenshot:**



## **Screenshot from 2025-02-21 14-47-39**

## **Code(java):**

package com.example.a43;

import android.os.Bundle;

import android.webkit.WebView;

import android.webkit.WebViewClient;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import com.example.swiperefreshapp.R;

public class MainActivity extends AppCompatActivity {

private TextView textView;

private WebView webView;

private SwipeRefreshLayout swipeRefreshLayout;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Initialize the views

textView = findViewById(R.id.text); // ID should match the XML layout

webView = findViewById(R.id.webview);

swipeRefreshLayout = findViewById(R.id.swipeRefreshLayout); // Correct ID for SwipeRefreshLayout

// Set initial text

textView.setText("Not Refreshed");

// Setup WebView

webView.setWebViewClient(new WebViewClient());

webView.getSettings().setJavaScriptEnabled(true);

webView.loadUrl("https://www.github.com");

// Swipe refresh listener

swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

@Override

public void onRefresh() {

// Refresh the WebView content

webView.loadUrl("https://www.google.com");

textView.setText("App is Refreshed");

swipeRefreshLayout.setRefreshing(false); // Hide the refresh indicator

}

});

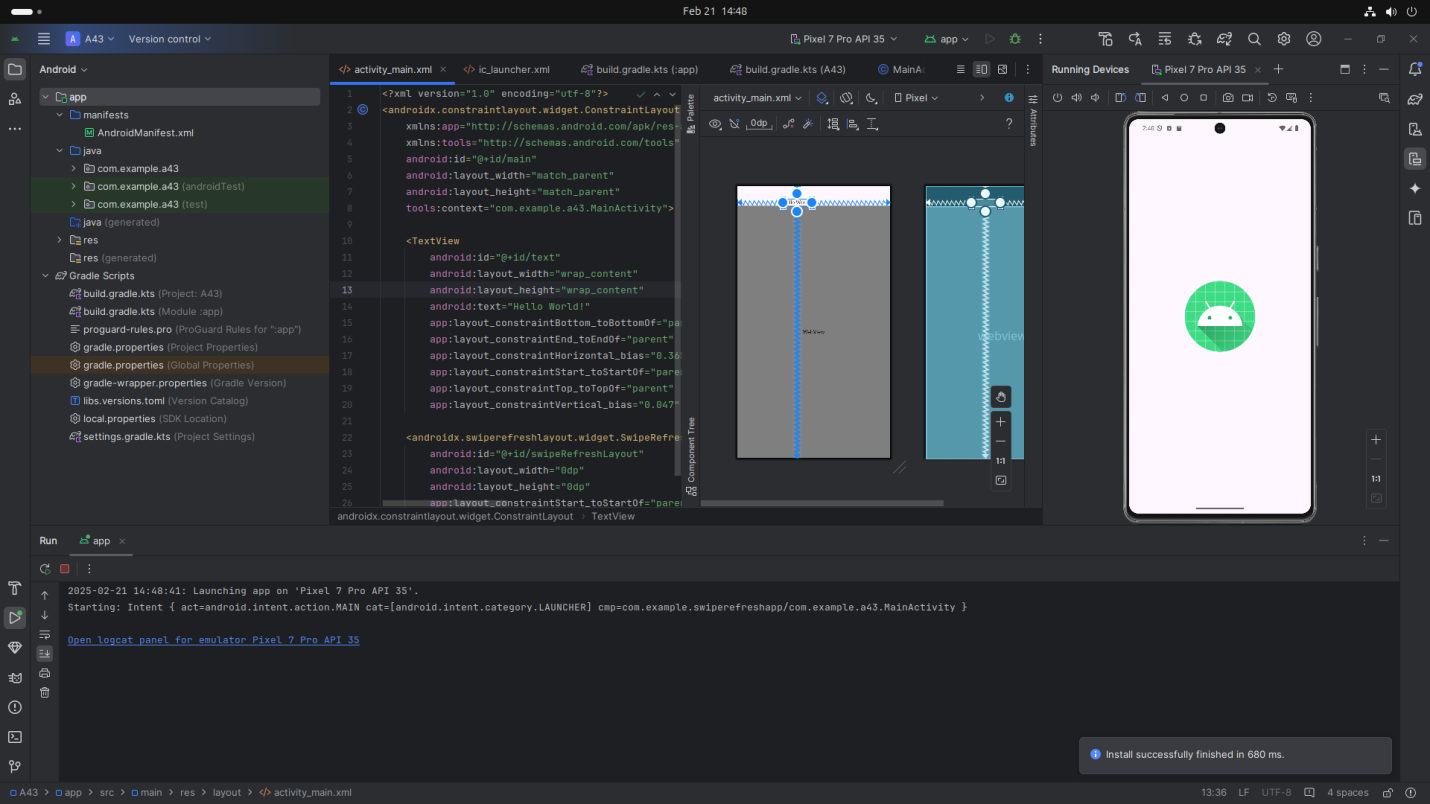
}

}

### **Step 4: Running the Application on Emulator**

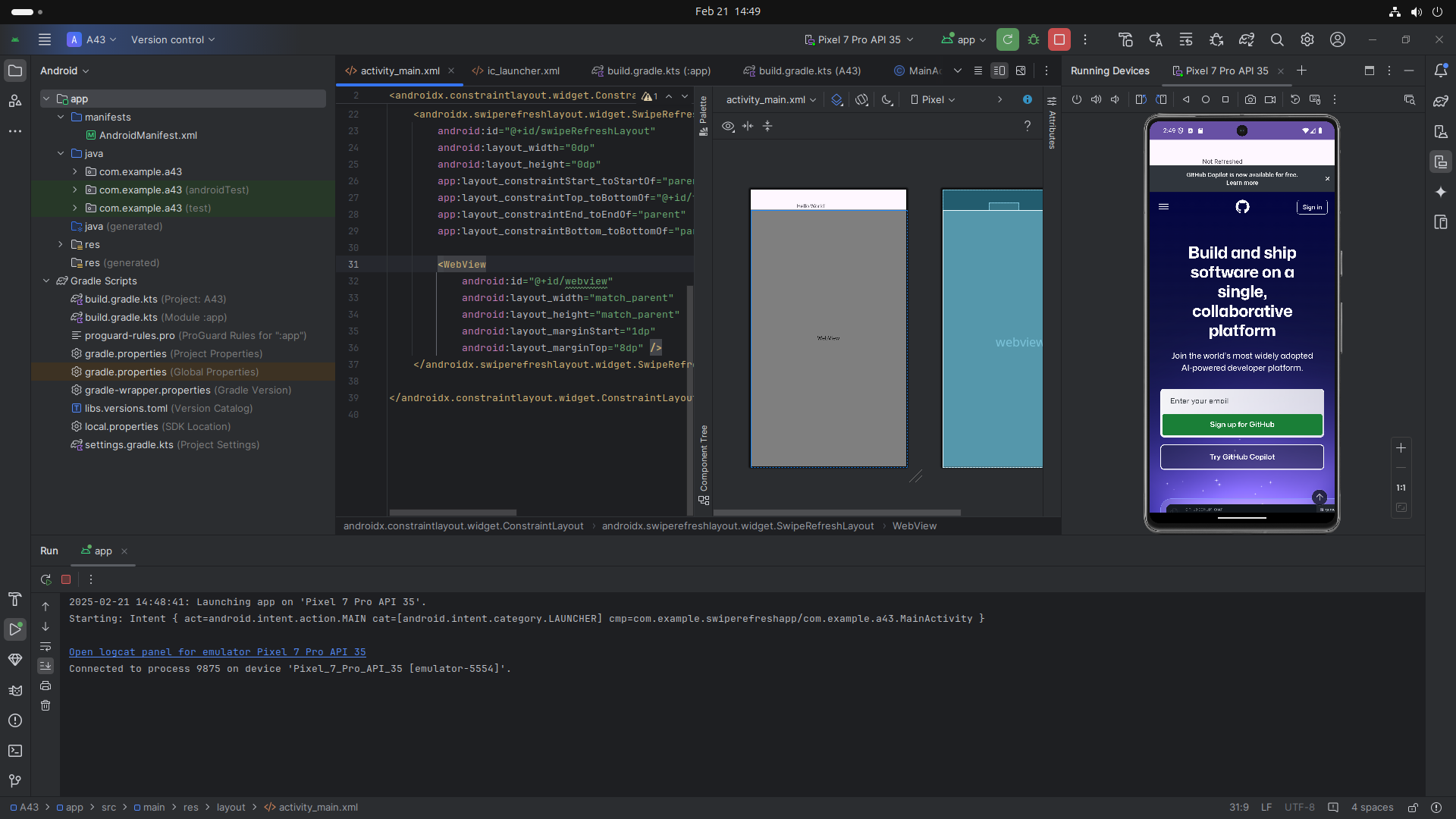
* Click on the **Run** button in Android Studio.
* Select the emulator and launch the app.

**Screenshot:**

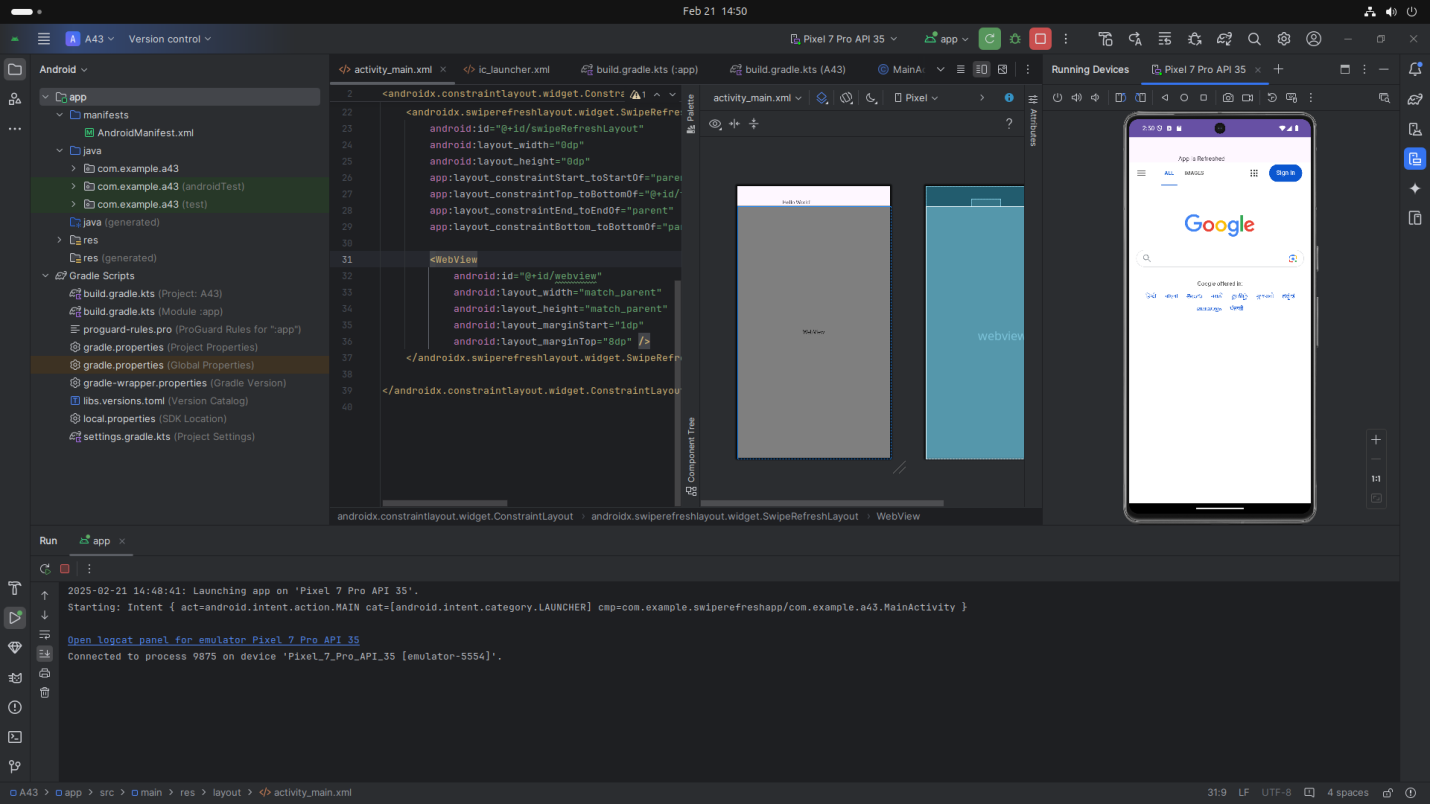
****

### **Step 5: Testing & Output**

* Test different functionalities of the app.
* Capture the output results.
* **Screenshot:**



**After Refreshing**

******

***(Button\_Click\_App)***

## **1. Introduction**

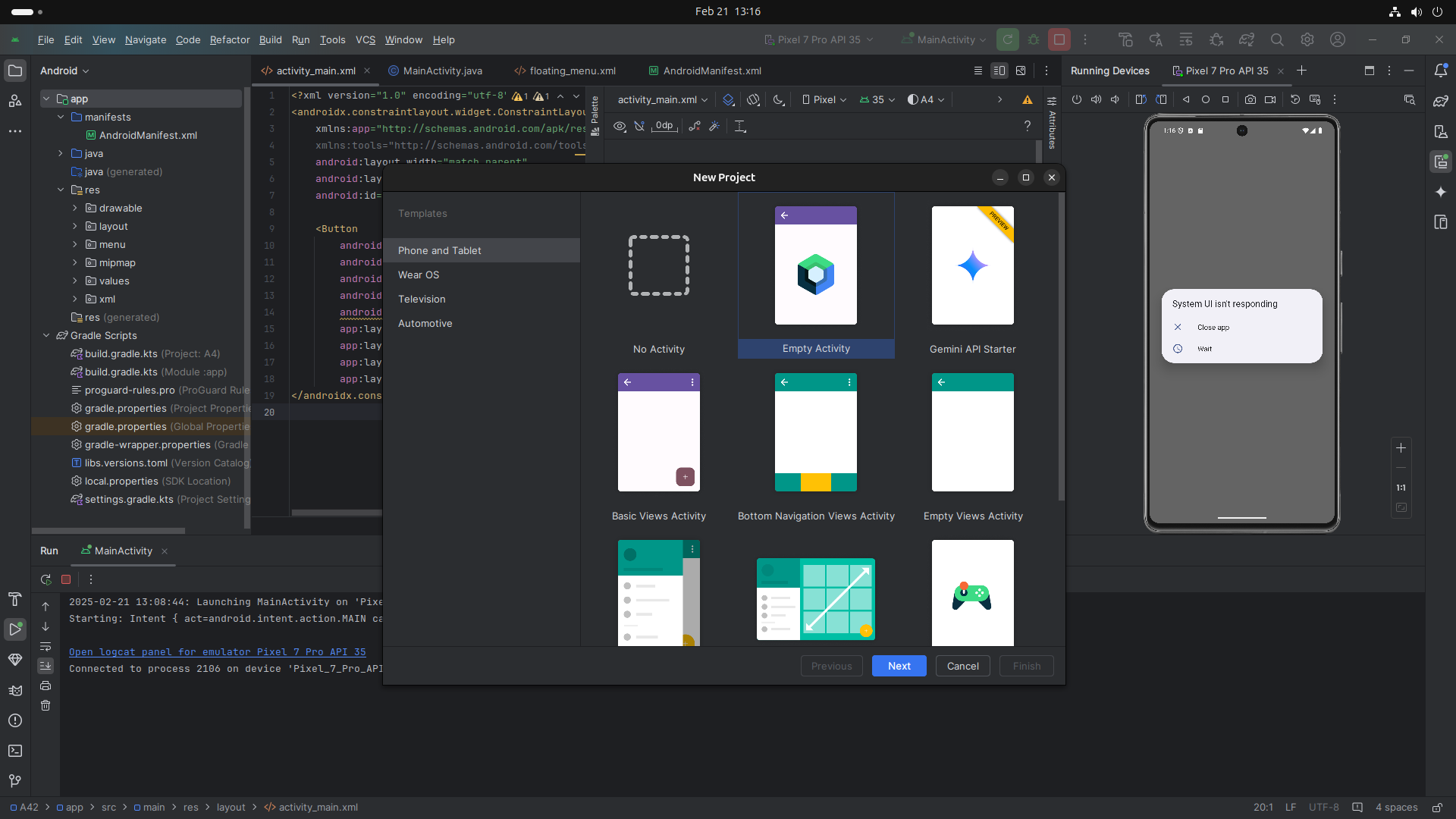
## In this app, we will create a main screen with buttons that users can click. Each time a button is clicked, the app will show a message or perform an action. This will demonstrate how the app reacts to user interactions, using simple event handling.

## **2. Tools & Technologies Used**

* Android Studio
* Kotlin
* Emulator

## **3. Procedure & Steps**

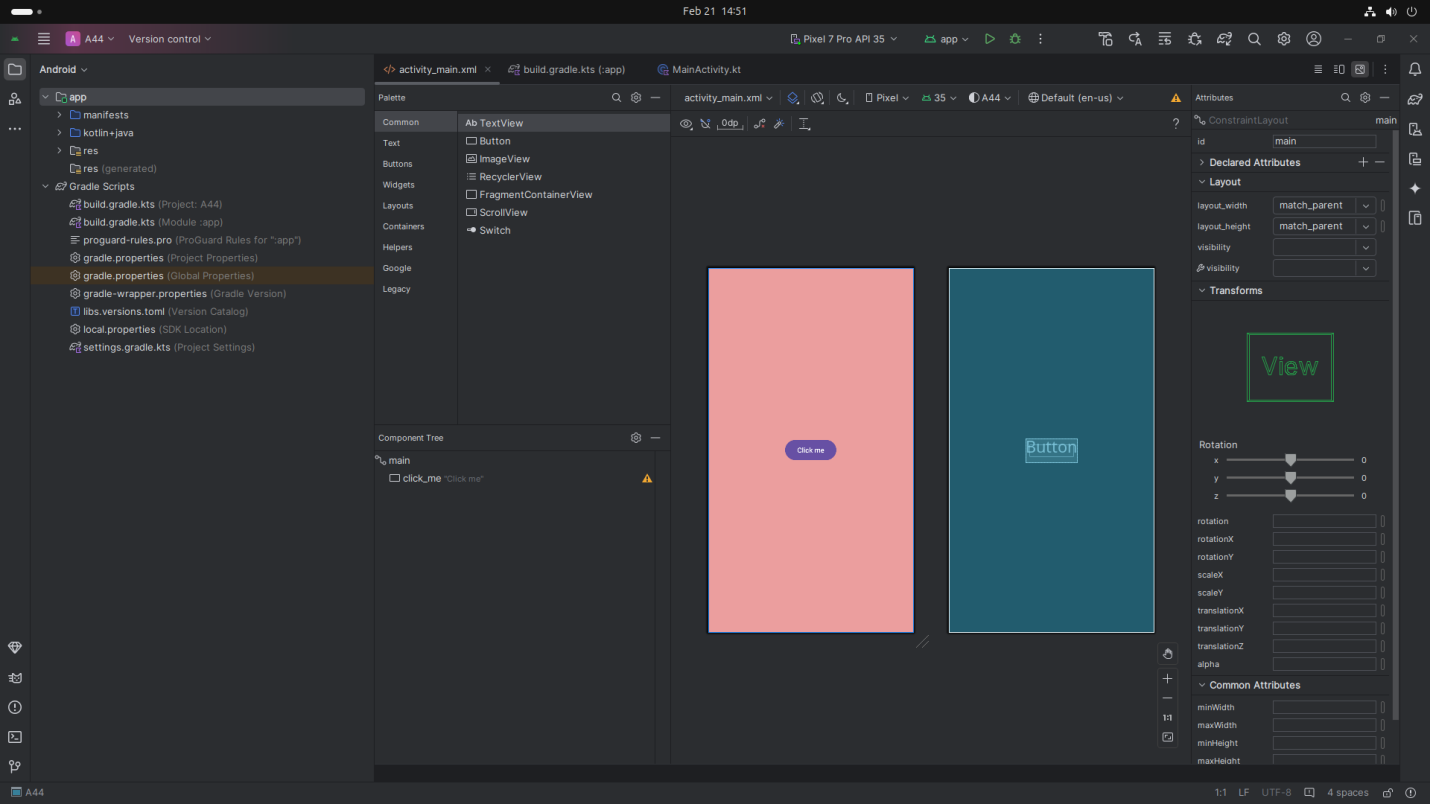
### **Step 1: Create a New Project**

* Open Android Studio and create a new project.
* Choose an Empty Views Activity template.
* Set the project name and package name of your Application
* Select the programming language (Kotlin).
* **Screenshot:**
* 

### **Step 2: Designing the UI**

* Open activity\_main.xml and design the layout using XML.
* Add UI comwponents such as TextView, Button etc.

**Screenshot:**

****

## **Code:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#EB9E9E"

tools:context=".MainActivity">

<Button

android:id="@+id/click\_me"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:text="Click me"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

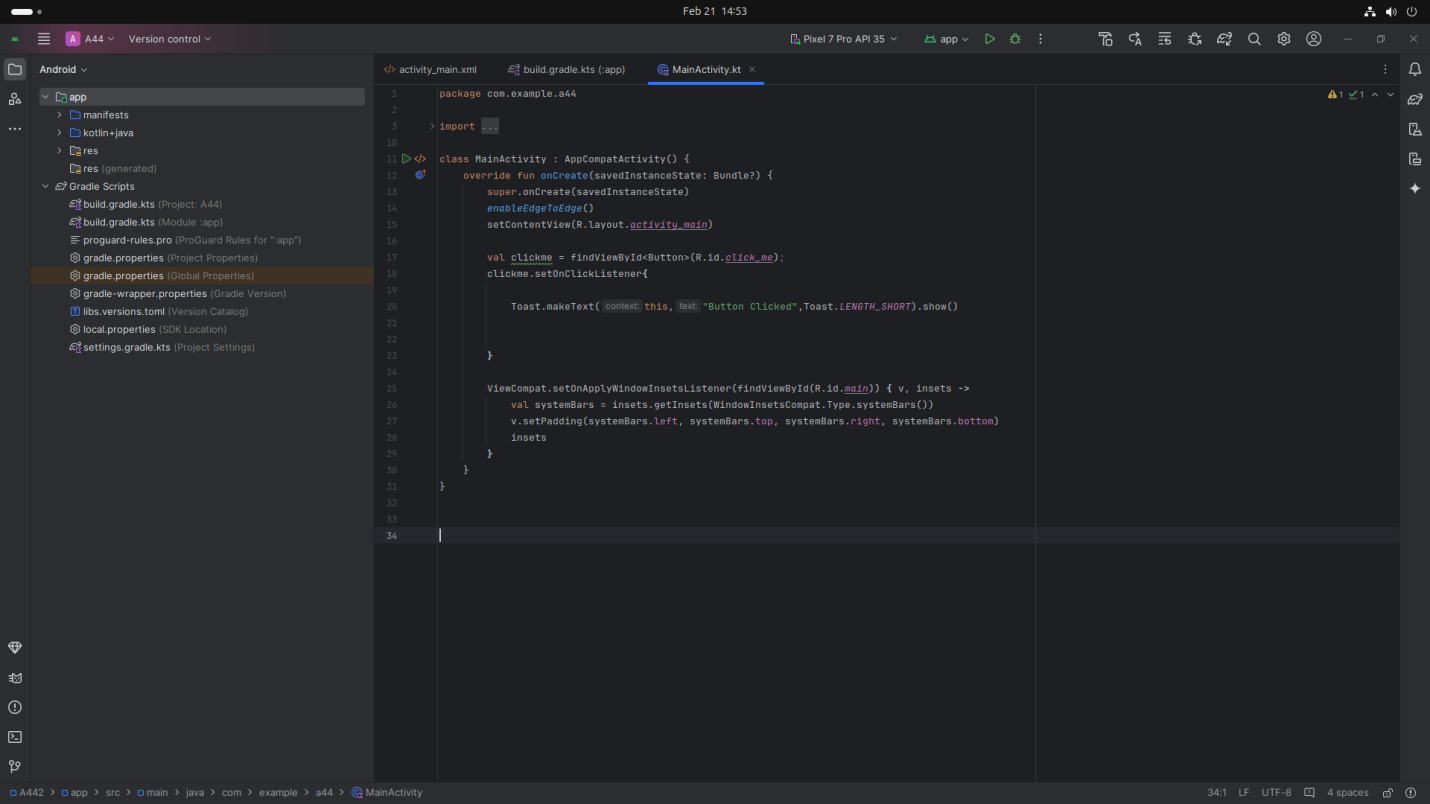
app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Step 3: Writing the Backend Code (Kotlin)**

* Open MainActivity.kt
* Implement functionality such as By Click On Button Display Message Button is Clicked
* **Screenshot:**

****

## **Code: (Kotlin)**

### package com.example.a44

### import android.os.Bundle

### import android.widget.Button

### import android.widget.Toast

### import androidx.activity.enableEdgeToEdge

### import androidx.appcompat.app.AppCompatActivity

### import androidx.core.view.ViewCompat

### import androidx.core.view.WindowInsetsCompat

### class MainActivity : AppCompatActivity() {

### override fun onCreate(savedInstanceState: Bundle?) {

### super.onCreate(savedInstanceState)

### enableEdgeToEdge()

### setContentView(R.layout.activity\_main)

### val clickme = findViewById<Button>(R.id.click\_me);

### clickme.setOnClickListener{

### Toast.makeText(this,"Button Clicked",Toast.LENGTH\_SHORT).show()

### }

### ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->

### val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())

### v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)

### insets

### }

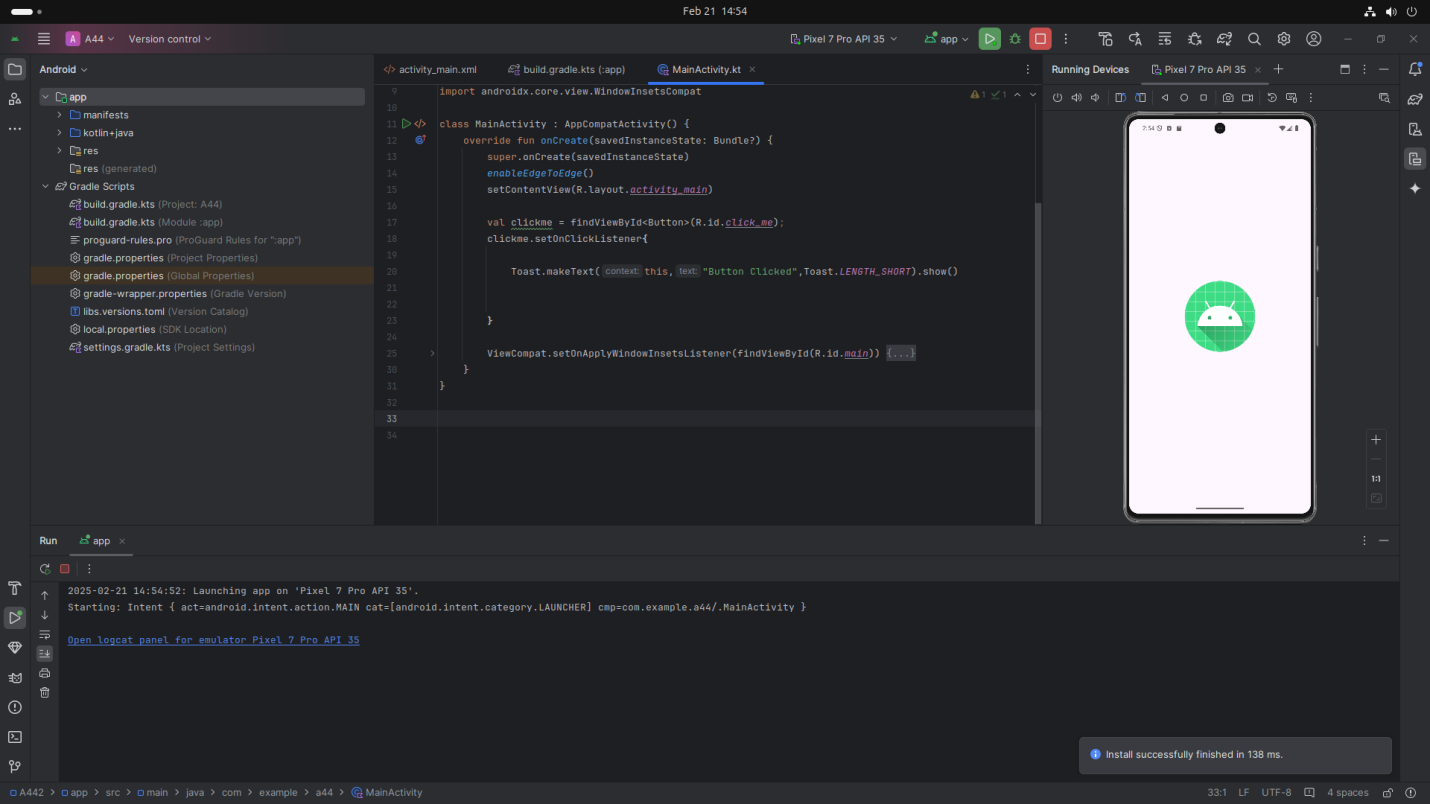
### }

### }

### **Step 4: Running the Application on Emulator**

* Click on the **Run** button in Android Studio.
* Select the emulator and launch the app.

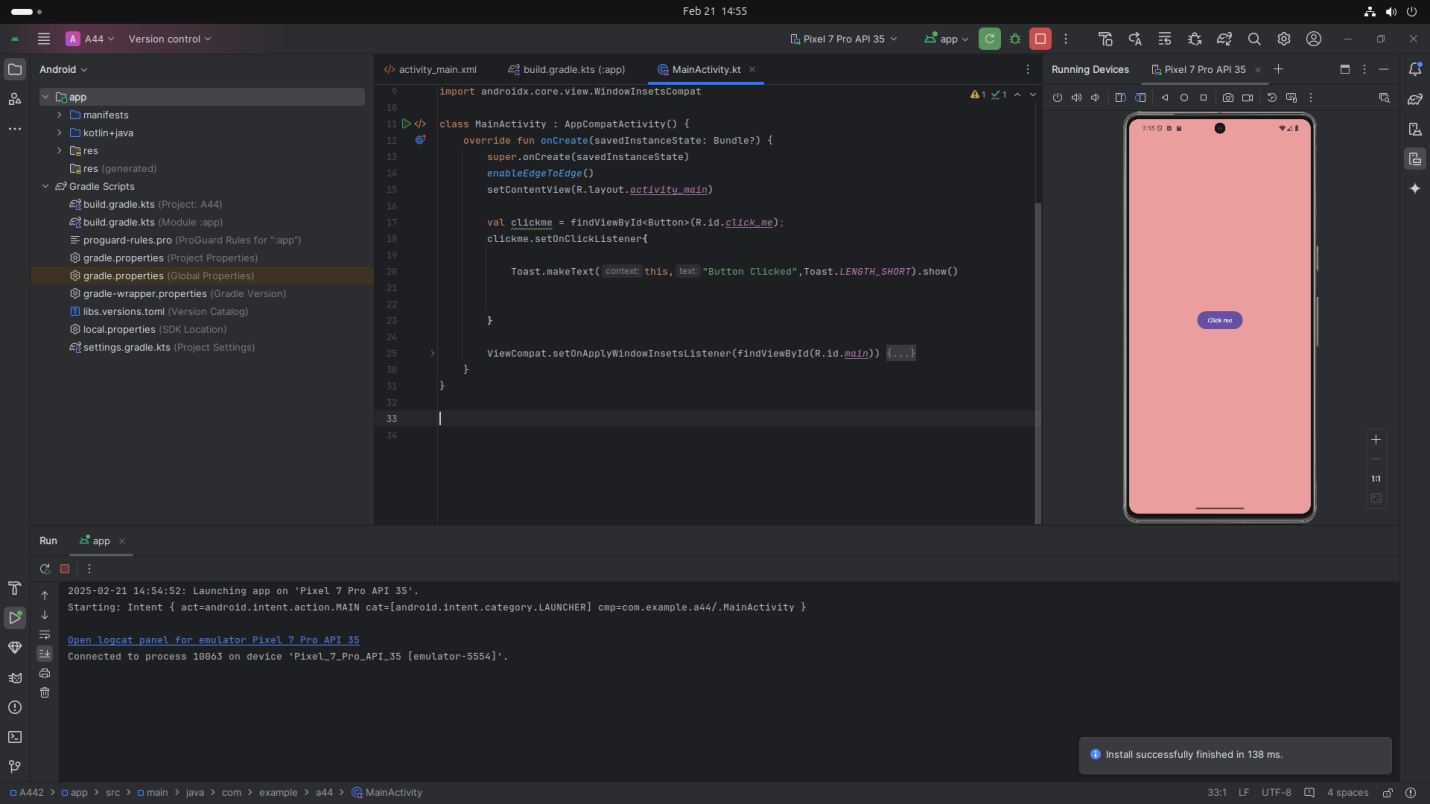
**Screenshot:**



### **Step 5: Testing & Output**

* Test different functionalities of the app.
* Capture the output results.

**Screenshot:**





## **4. Conclusion**

Through this project, I learned how to handle different user interactions in an Android app using Java/Kotlin. I gained useful experience in setting up event listeners for button clicks, long presses, and swipe gestures to improve user interactivity. One challenge I encountered was ensuring quick and accurate responses to events, but I tackled it with thorough testing and debugging. By the end, I became more confident in creating interactive applications and enhancing user engagement. This experience helped me sharpen my problem-solving abilities and deepened my understanding of Android event management.