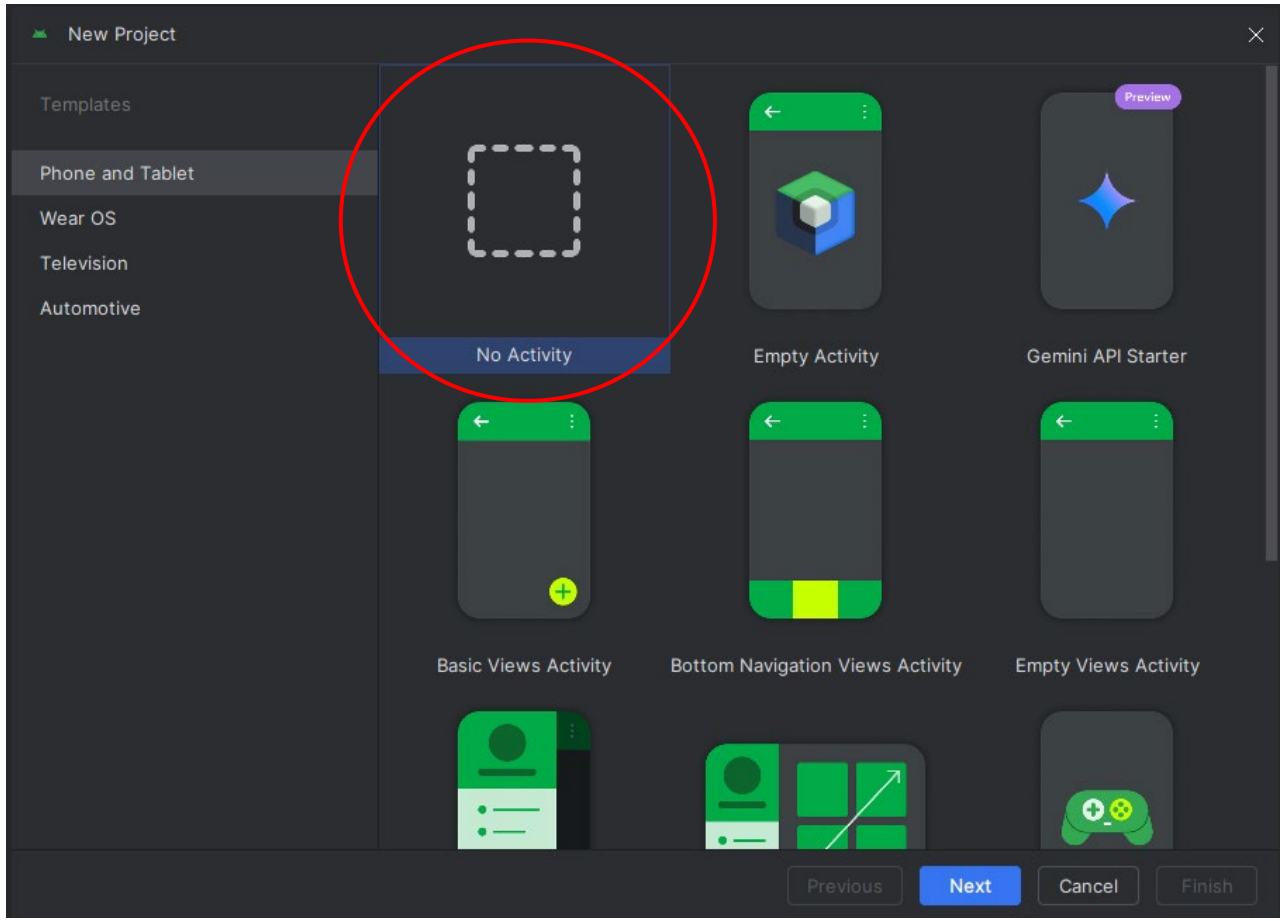
Android Tutorial

Outline

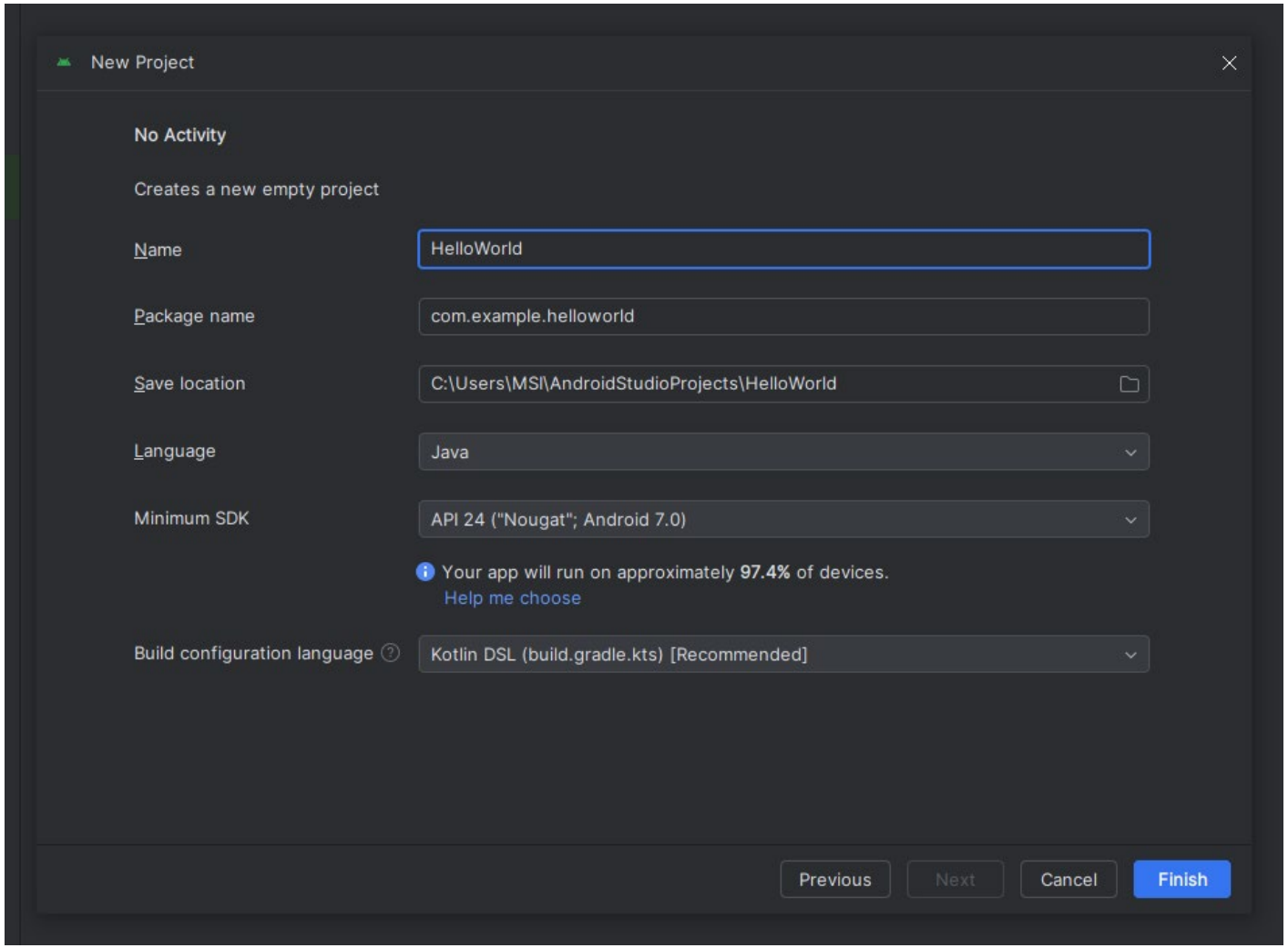
- Setting Up a New Project
- Understanding the Directory Layout
- Activity, Layout, and the Relationship
- Hello World Application
  - Creating a simple UI
  - Writing the logic
  - Running the app in the emulator
- Common Issues

Setting Up a New Project

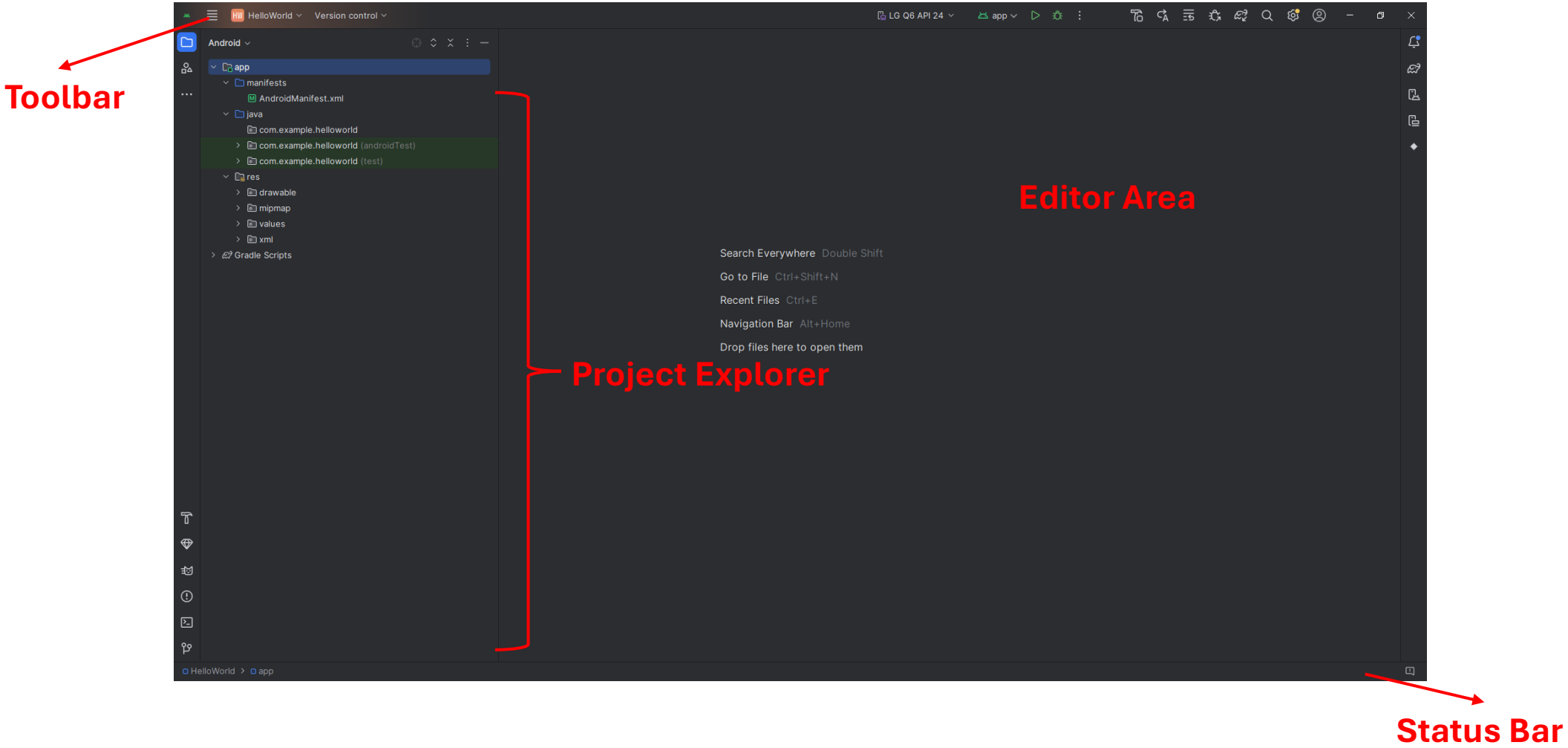
Select No Activity



Fill in the details → Finish

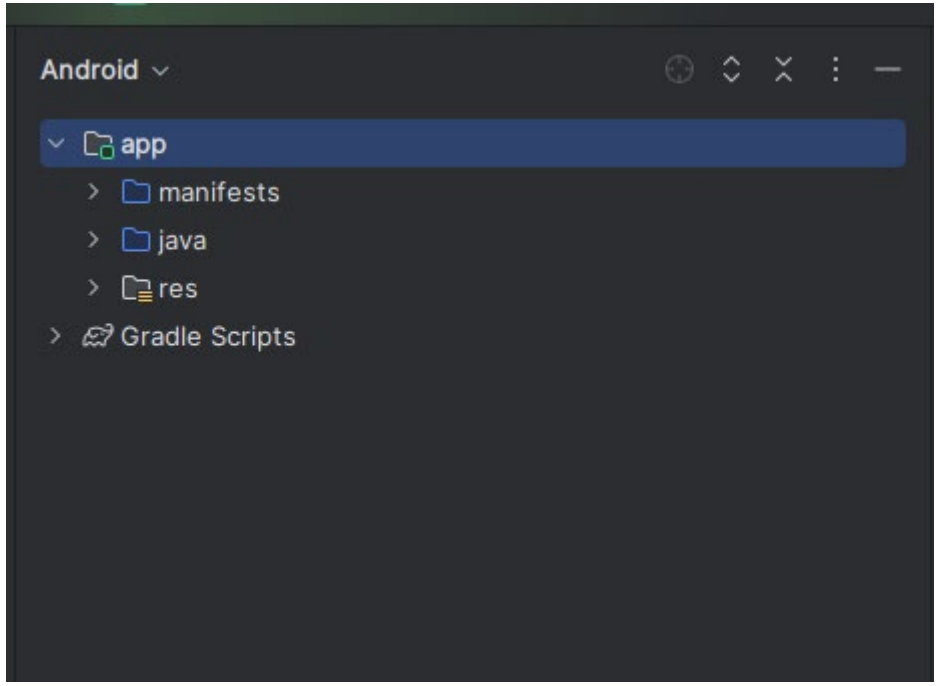


Initial Starting Point



Understanding the Directory Layout

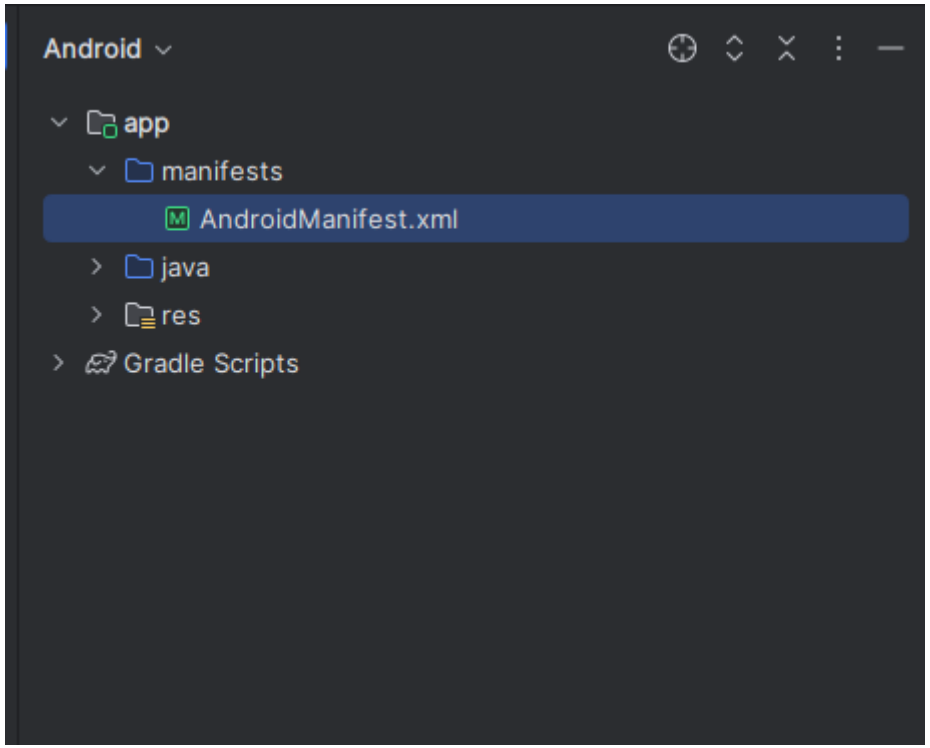
The High-level Directory Structure



- **app**: The main module of your Android application that contains all your code and resources.

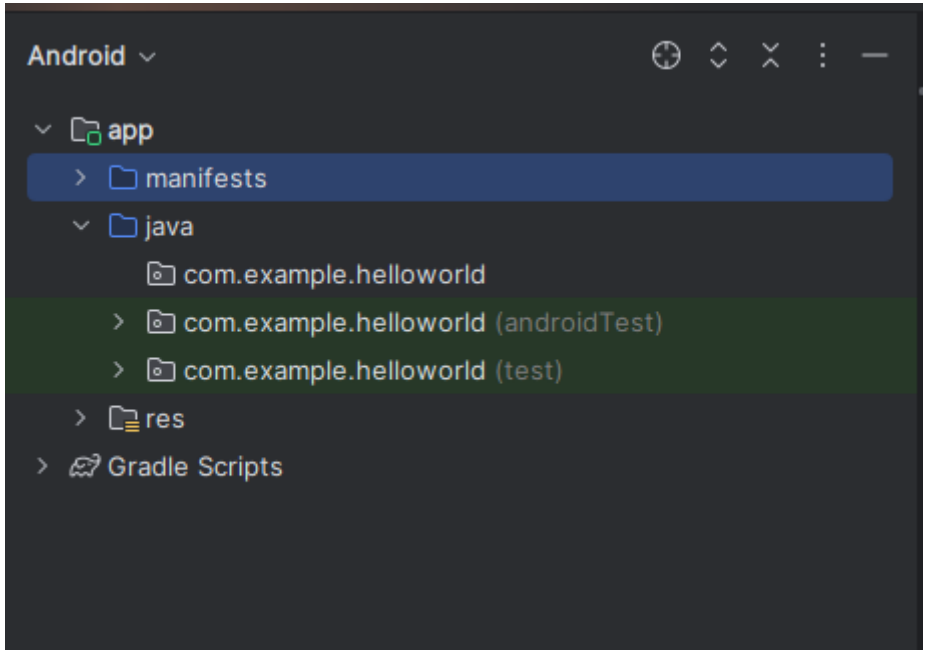
- **Gradle Scripts**: Contains build configuration files that define:
  - Dependencies and libraries
  - How your app is built
  - SDK versions
  - Build variants

What’s inside “manifest”



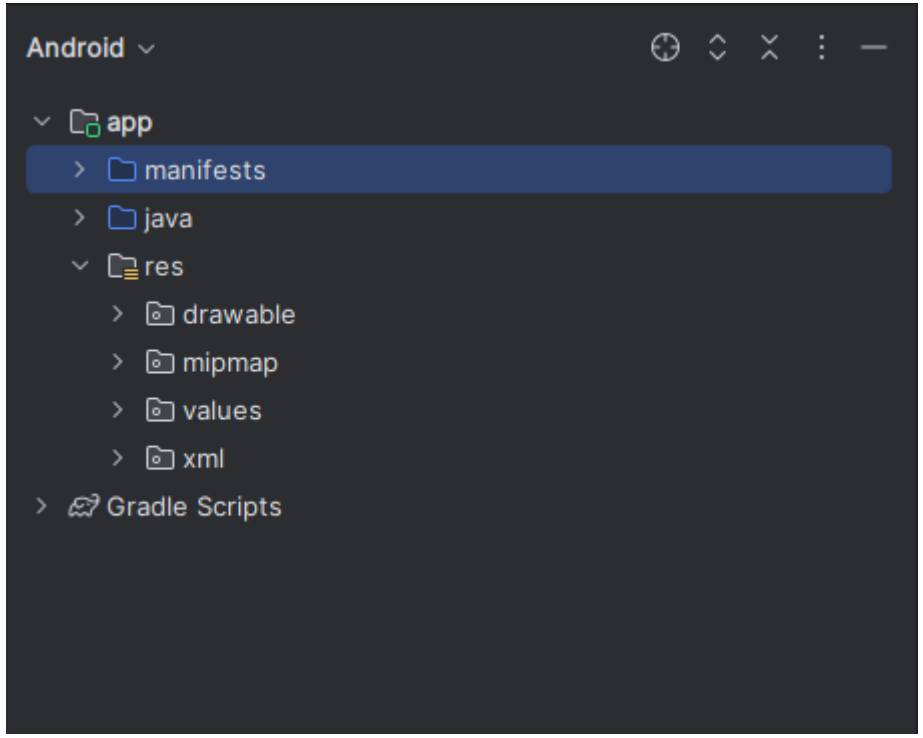
- Contains the **AndroidManifest.xml** file.
  - Defines app components (activities, services, etc.)
  - Lists required permissions
  - Specifies minimum Android version support

What's inside "java"



- Contains all Java/Kotlin source code files:
  - **com.example.helloworld**: Main package for application code
  - **com.example.helloworld (androidTest)**: Contains instrumented tests that run on Android devices
  - **com.example.helloworld (test)**: Contains unit tests that run on your local machine

What’s inside “res”



- Contains all non-code resources:
  - **drawable**: Images, shapes, and XML drawable definitions
  - **mipmap**: App icons in different densities
  - **values**: XML files with strings, colors, styles, dimensions
  - **xml**: Other XML configuration files

Activity, Layout, and the Relationship

An Android app comprises of Activity and Layout

Activity: The "Brains"

- **Purpose**: contain the logic and behavior of your app, usually written in Java or Kotlin
- **What they do**:
  - Control what happens when users interact with the app
  - Process data and perform calculations
  - Connect to networks and databases
  - Handle the app's lifecycle (what happens when the app starts, pauses, resumes)
  - Respond to user actions (like button clicks)

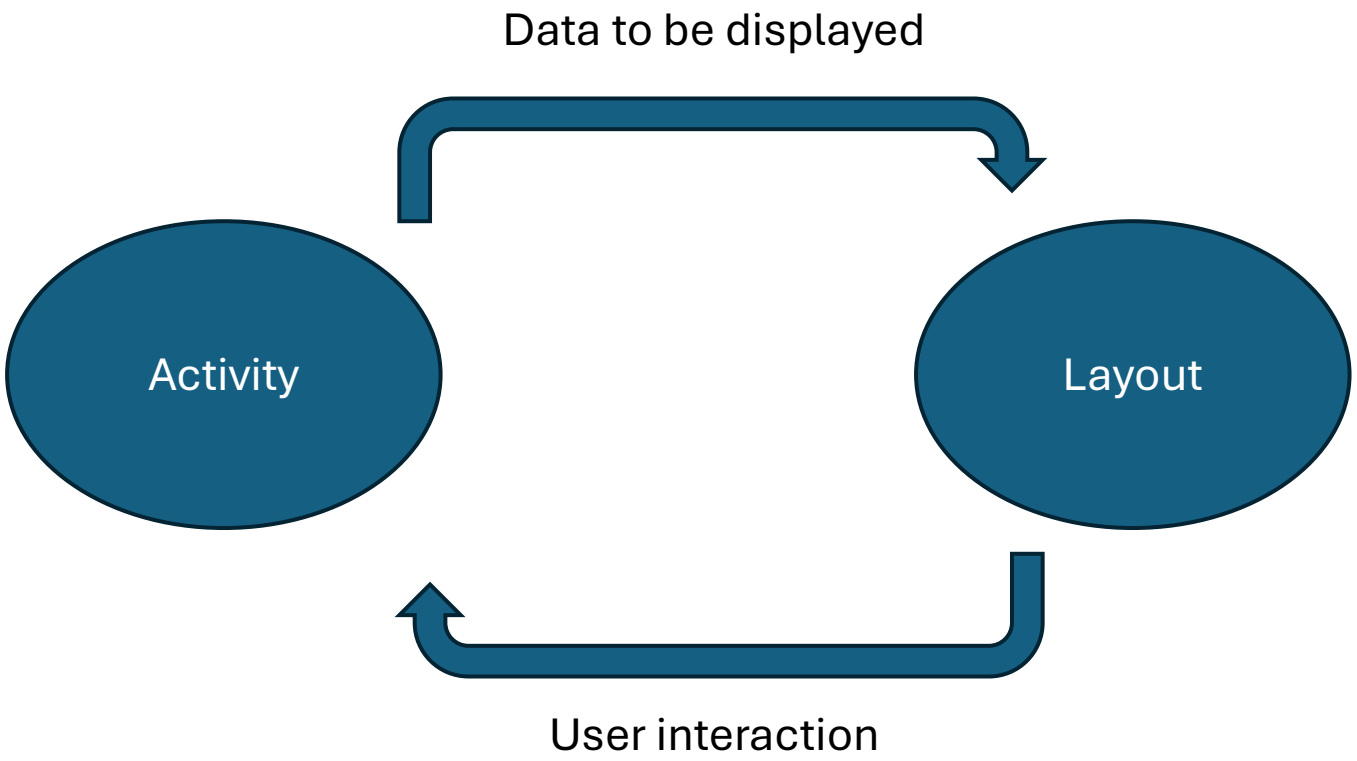
An Android app comprises of Activity and Layout

Layout: The "Body"

- **Purpose**: Layout define the structure, appearance, and resources of your app, usually written in XML files
- **Types of XML files**:
  - **Layout XML**: Defines the UI elements and their arrangement on screen
  - **Manifest XML**: Configures app permissions and components
  - **Resources XML**: Stores strings, colors, dimensions, and styles

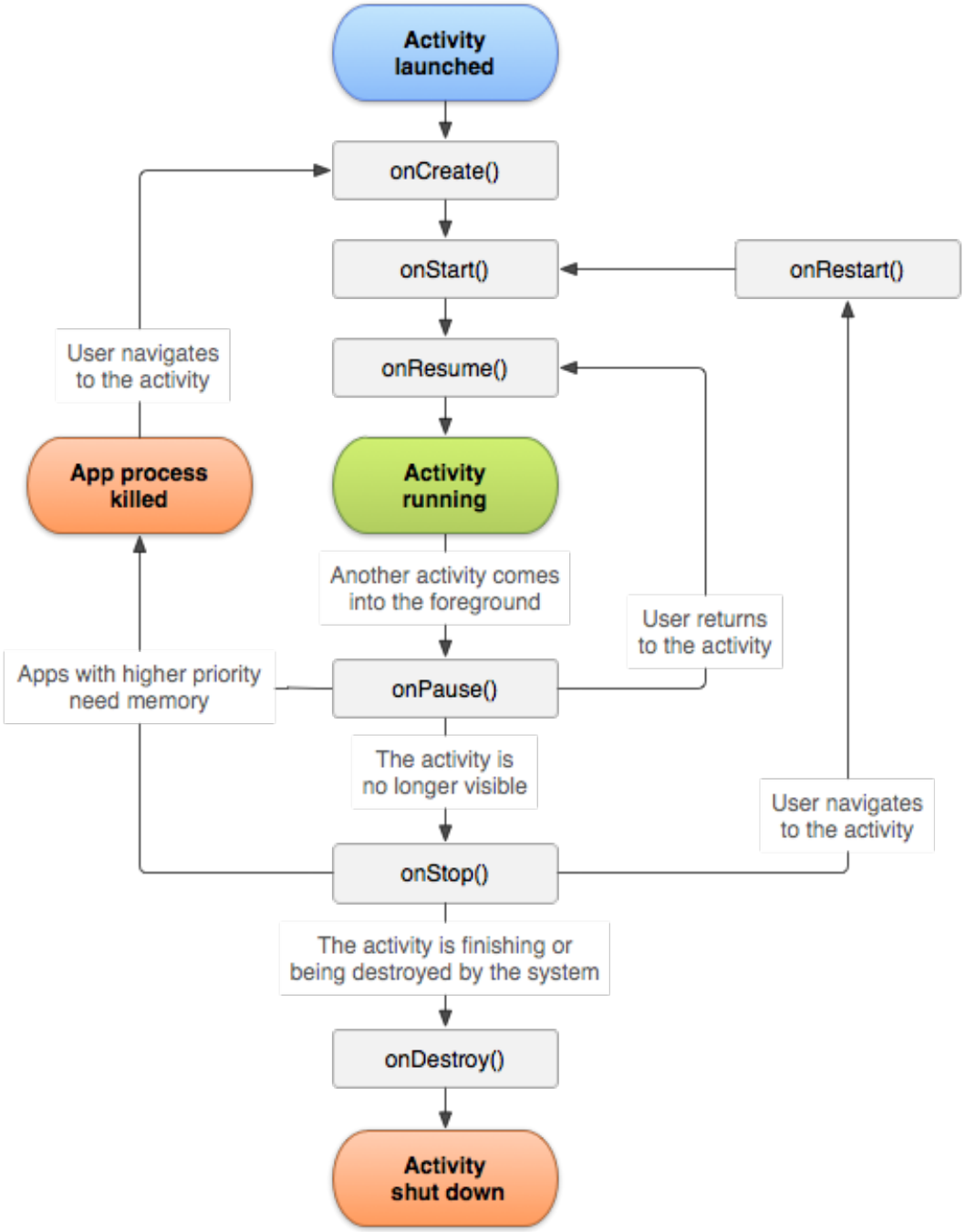
How activity and layout work together?

- Activity responds to user interactions with the layout
- Layout displays data managed by the Activity



App Lifecycle

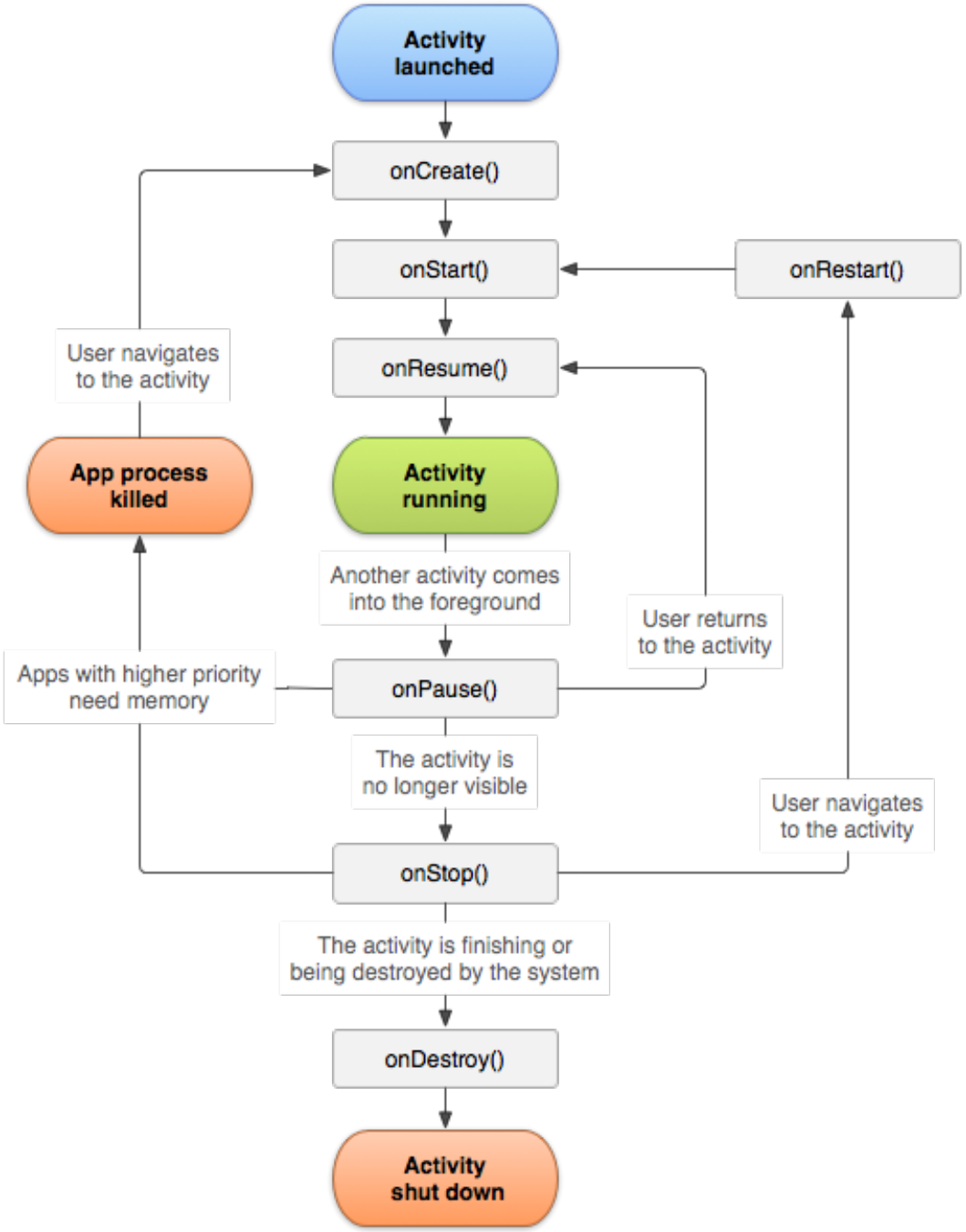
- The Android application lifecycle refers to the states an app goes through from launch to termination.



src: https://developer.android.com/guide/components/activities/activity-lifecycle

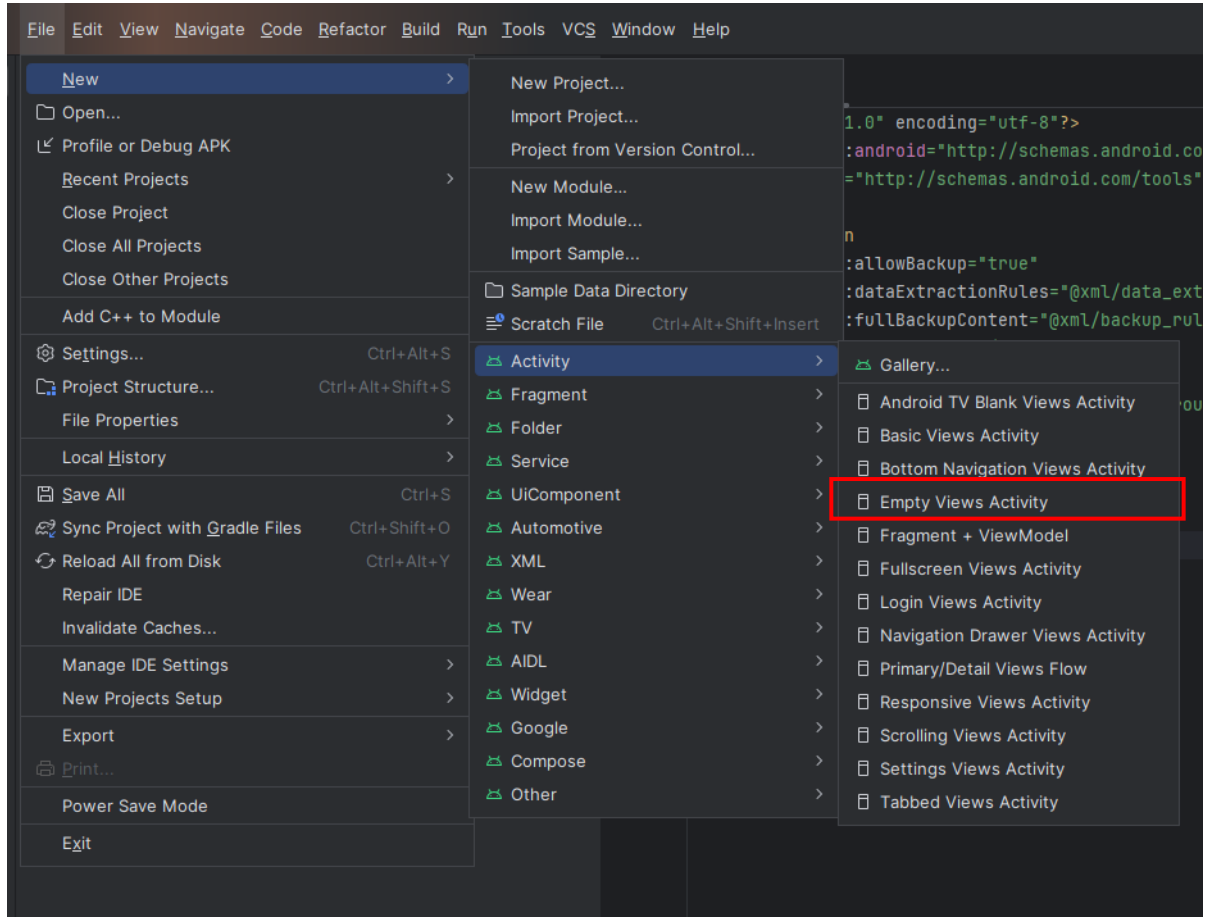
App Lifecycle

- onCreate(): Called when the activity is first created. Used for one-time initialization.
- onStart(): Called when the activity becomes visible to the user.
- onResume(): Called when the activity starts interacting with the user.
- onPause(): Called when the system is about to resume another activity.
- onStop(): Called when the activity is no longer visible to the user.
- onRestart(): Called when the activity restarts after being stopped.
- onDestroy(): Called before the activity is destroyed.

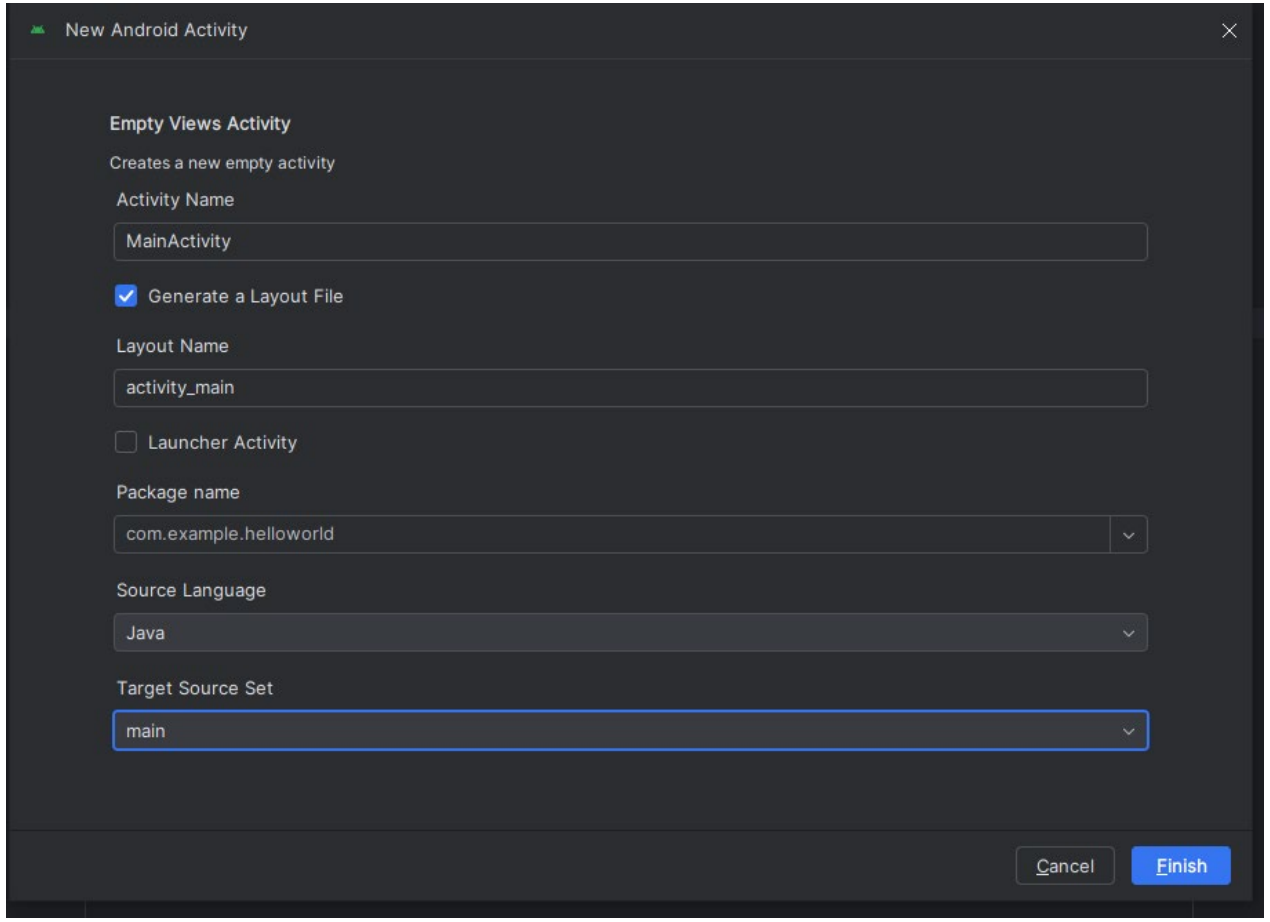


src: https://developer.android.com/guide/components/activities/activity-lifecycle

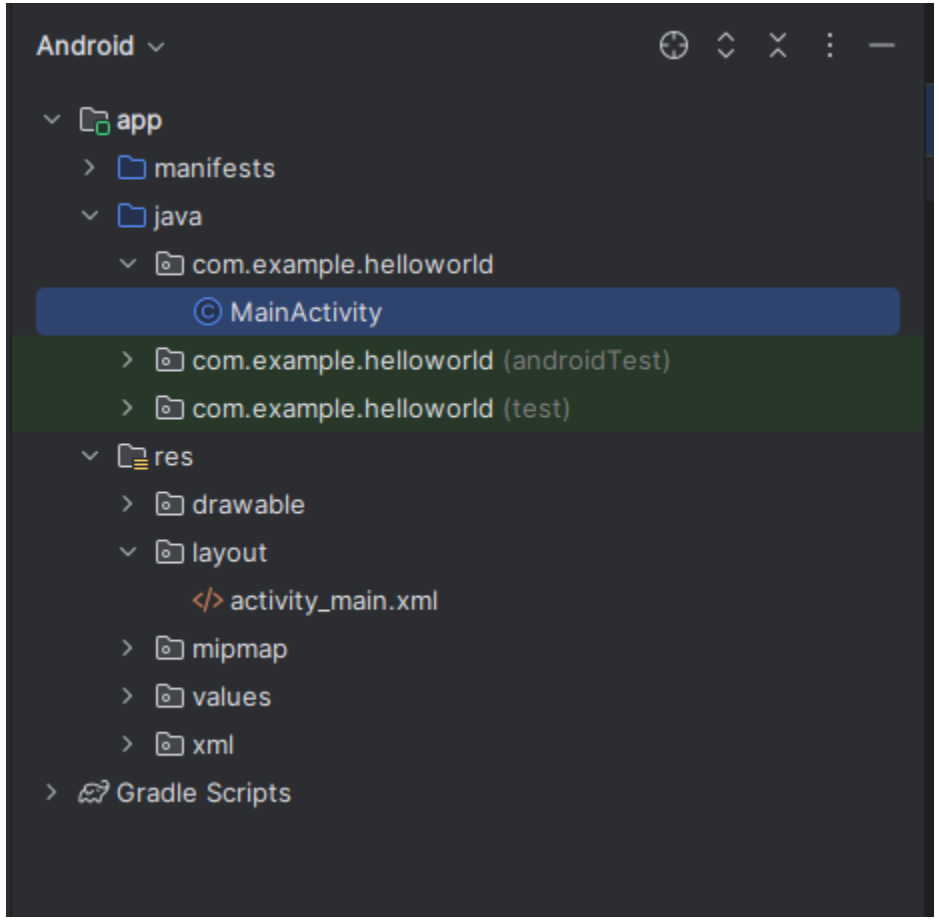
Create a new activity: File > New > Activity > Empty Views Activity



Fill the Details > Finish



Two new files will be created: **MainActivity.java** and **activity_main.xml**



- **MainActivity.java** -> defines the logic
- **activity_main.xml** → defines the layout

MainActivity.java

executed each time the activity is created

```java
package com.example.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```

loading the layout design and configuring display settings

activity_main.xml

AndroidManifest.xml | activity_main.xml | MainActivity.java

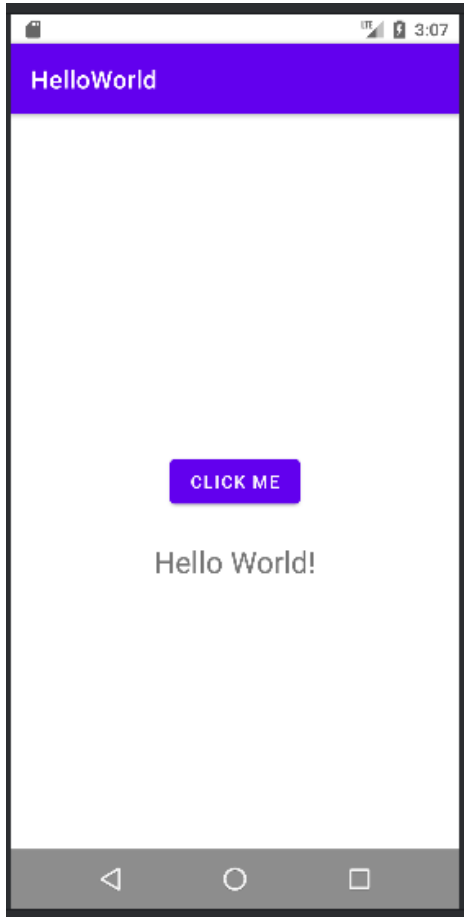
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"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

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

Created by default

Create a “Hello World” application

All codes that are used in this tutorial is available at https://github.com/imamnurby/cs708_tutorial/

Goal: Click the button, and the “Hello World” string will show up



Define the Button and the TextView

```
<Button
    android:id="@+id/helloButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/helloText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="24sp"
    android:visibility="gone"
    android:layout_marginTop="24dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/helloButton" />
```

- Add this code to the **activity_main.xml**
- Something to note:
  - **id** → we will use it to refer to this specific button in the java code later

Define the logic in the **MainActivity.java**

```java
public class MainActivity extends AppCompatActivity {

    // Define the button and text variable
    2 usages
    private Button button;
    2 usages
    private TextView text;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        // Find views by ID that we have specified in the activity_main.xml file
        button = findViewById(R.id.helloButton);
        text = findViewById(R.id.helloText);

        // Set up button click listener
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Show the "Hello World" text when the button is clicked
                text.setVisibility(View.VISIBLE);
            }
        });
    }
}
```

Define variable for the button and text

Connect this variable to the layout using the id in the xml file

Define the logic using button listener

Do not forget to include the necessary import

```java
package com.example.helloworld;

import android.os.Bundle;

// Define this
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
```

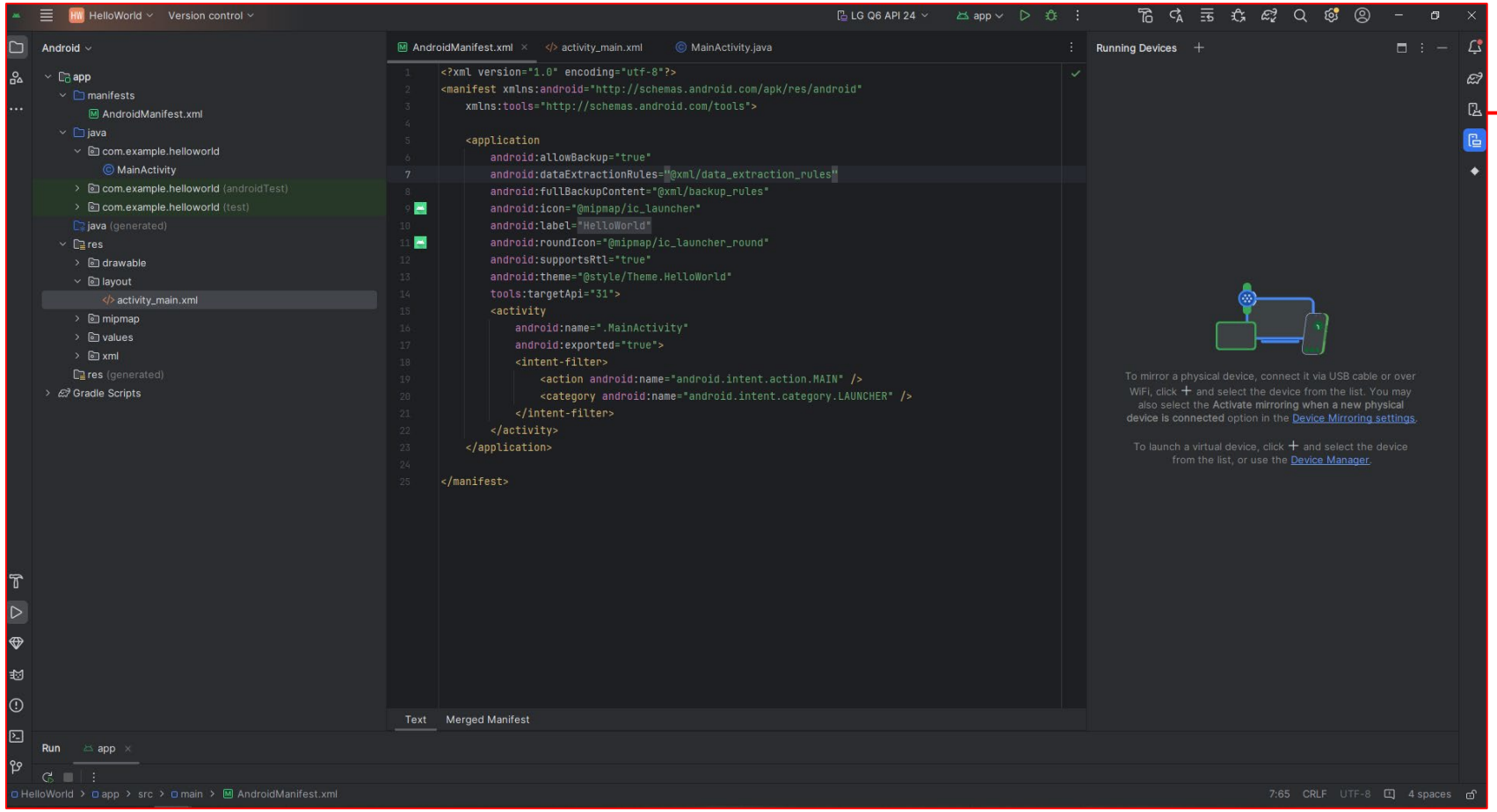
Define the application entry point

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="HelloWorld"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.HelloWorld"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

- This activity can be launched by the system (exported=true)
- This activity should be the entry point when the app is launched from the app drawer (MAIN/LAUNCHER)

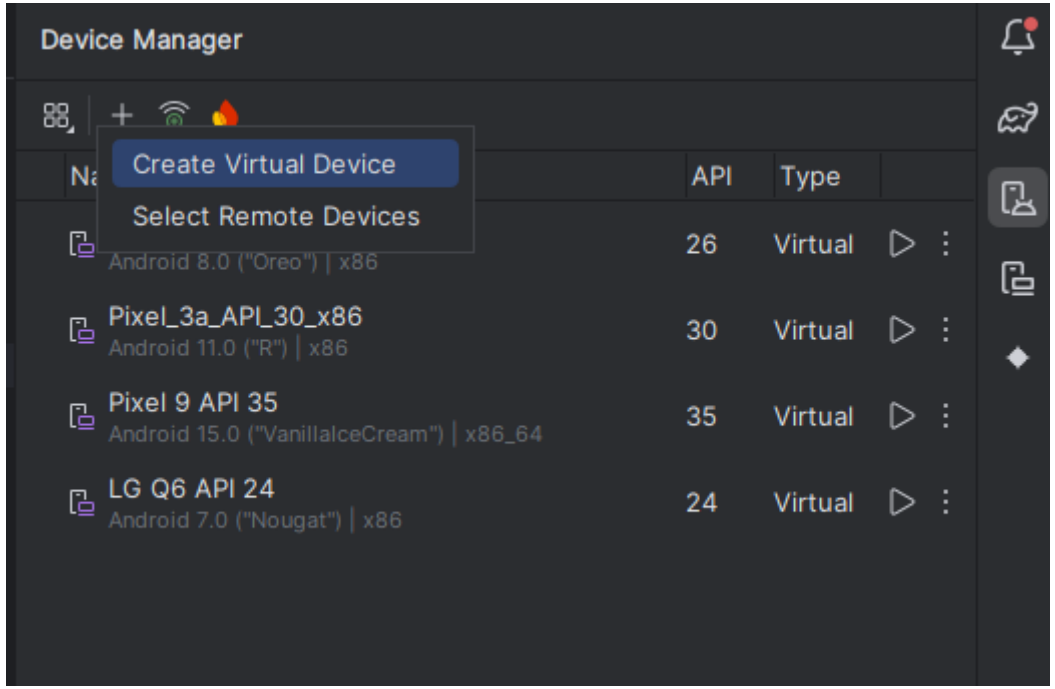
Running the App the Emulator

Instantiate a new device

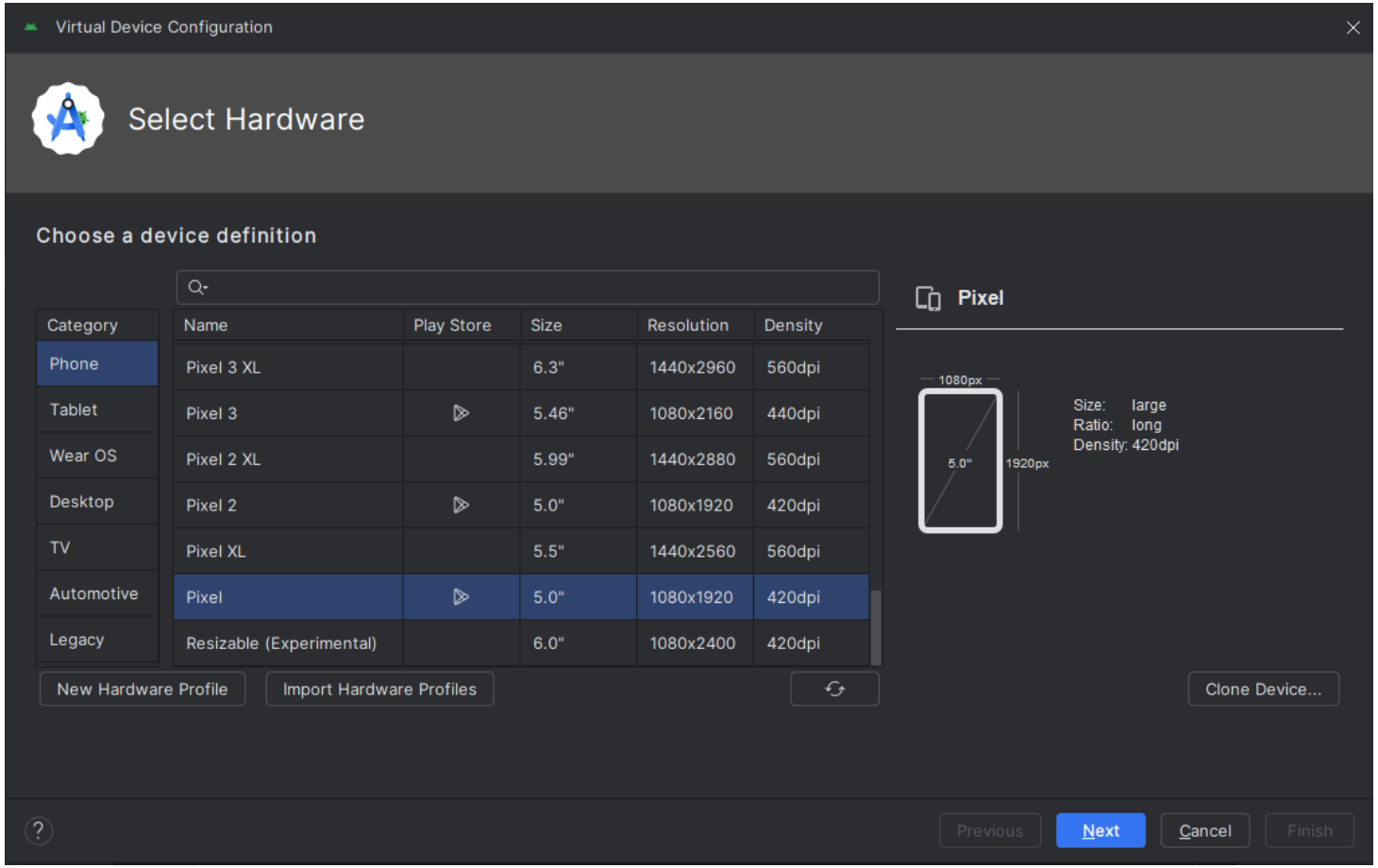


Click “Device Manager”

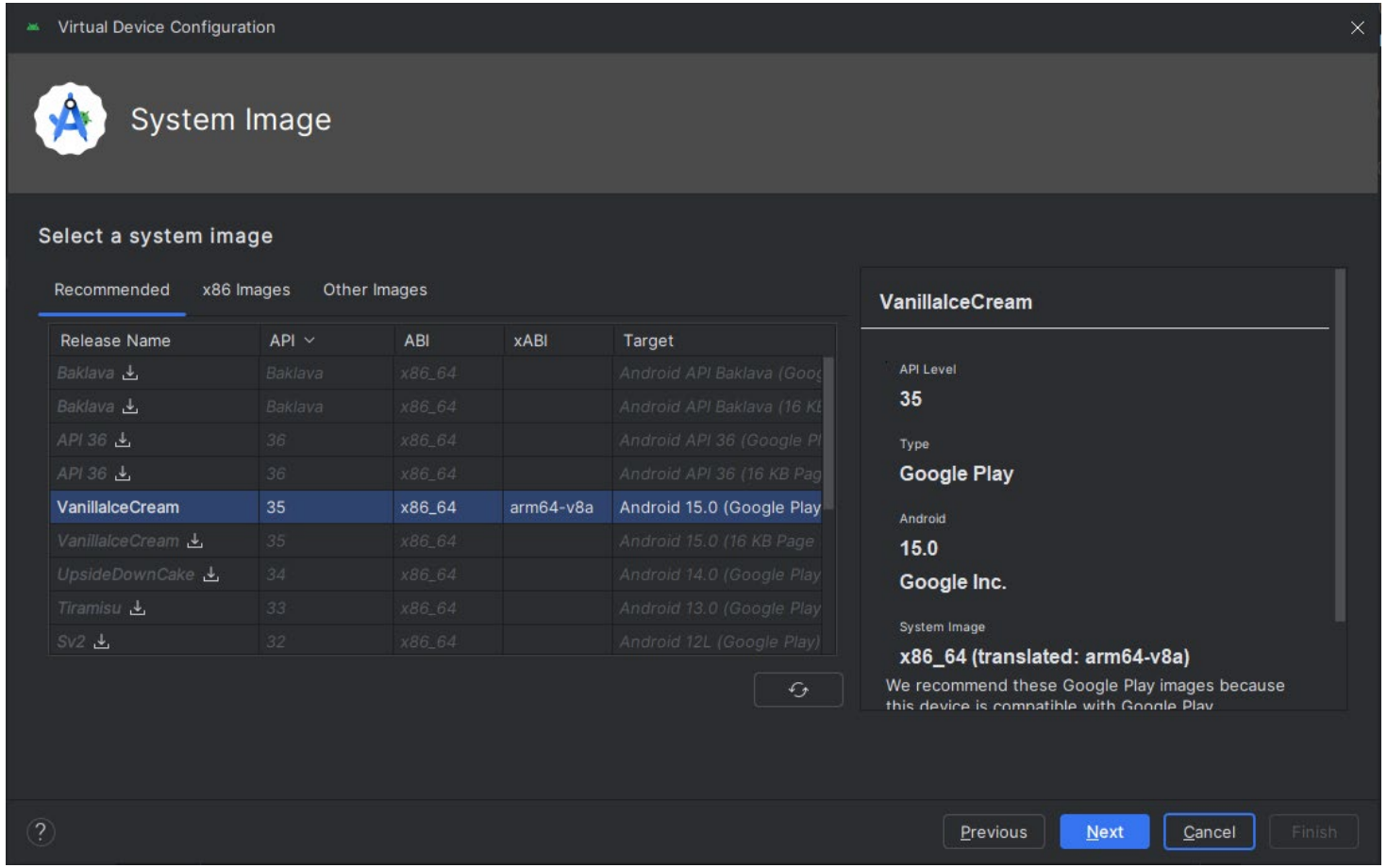
Click (+) → Create Virtual Device



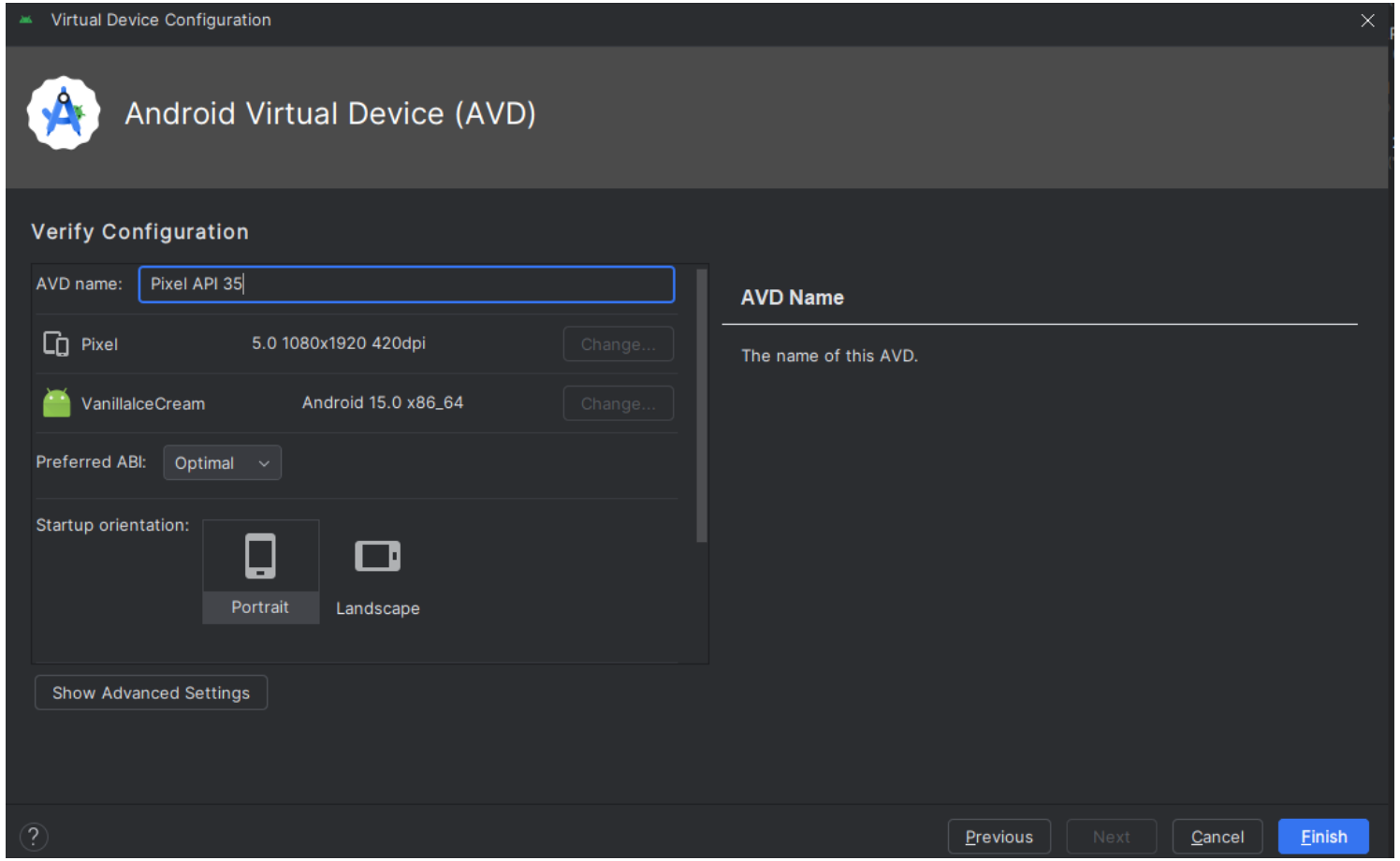
Select the hardware > Next



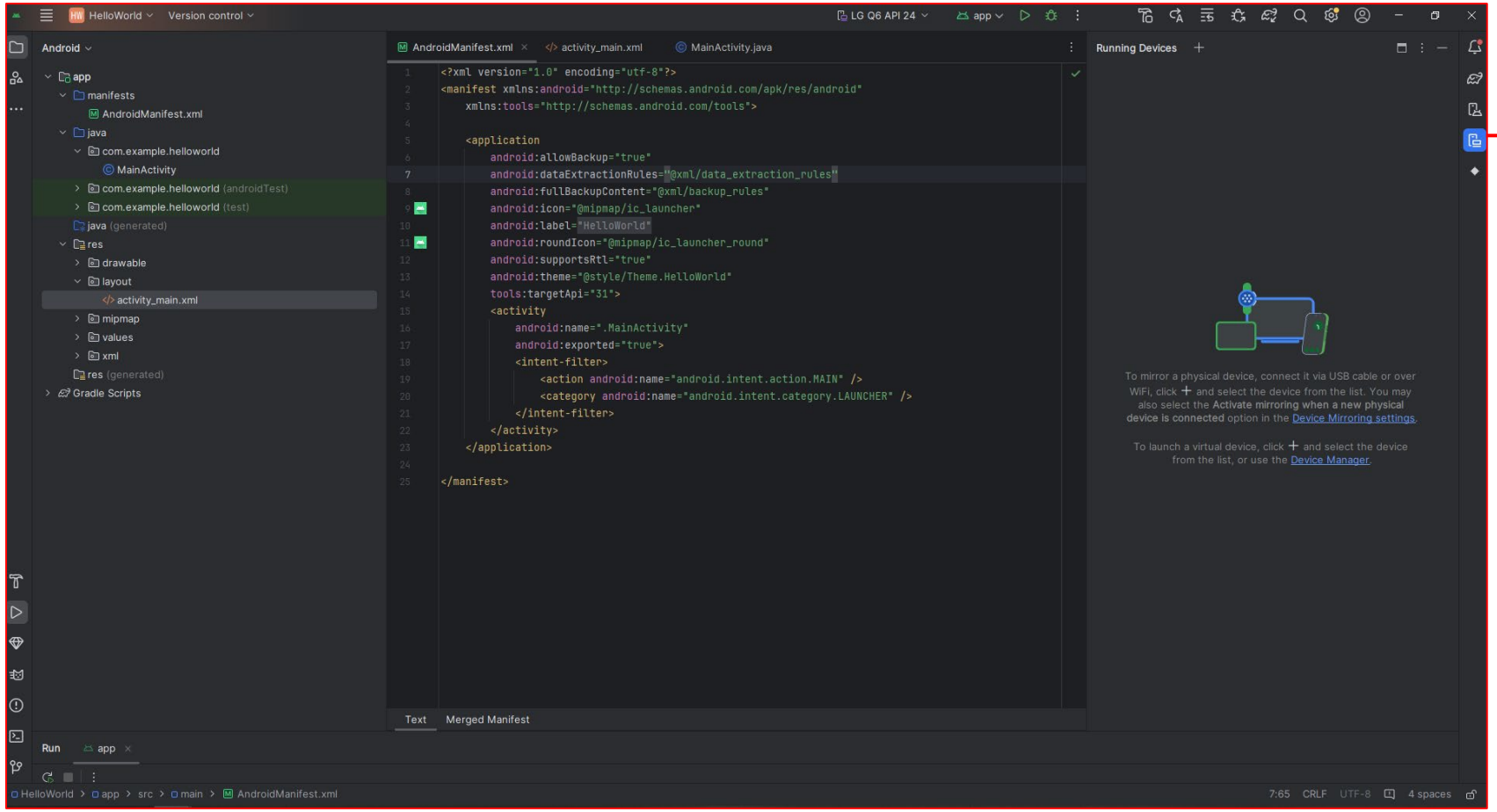
Choose the system image > Next



Finalize the setting > Finish

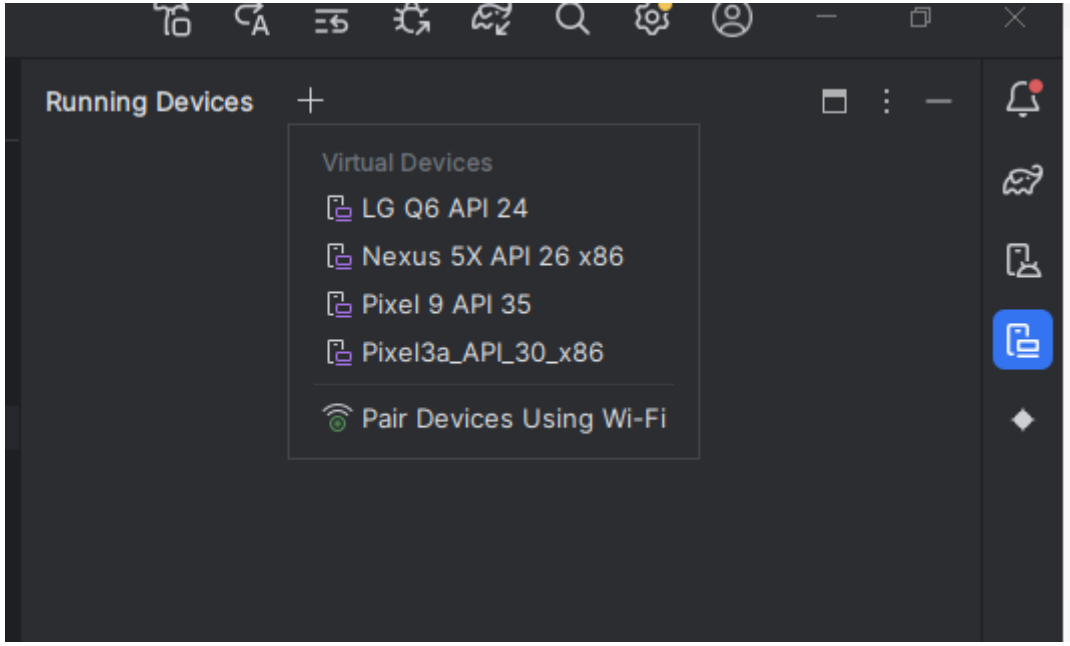


Start the emulator



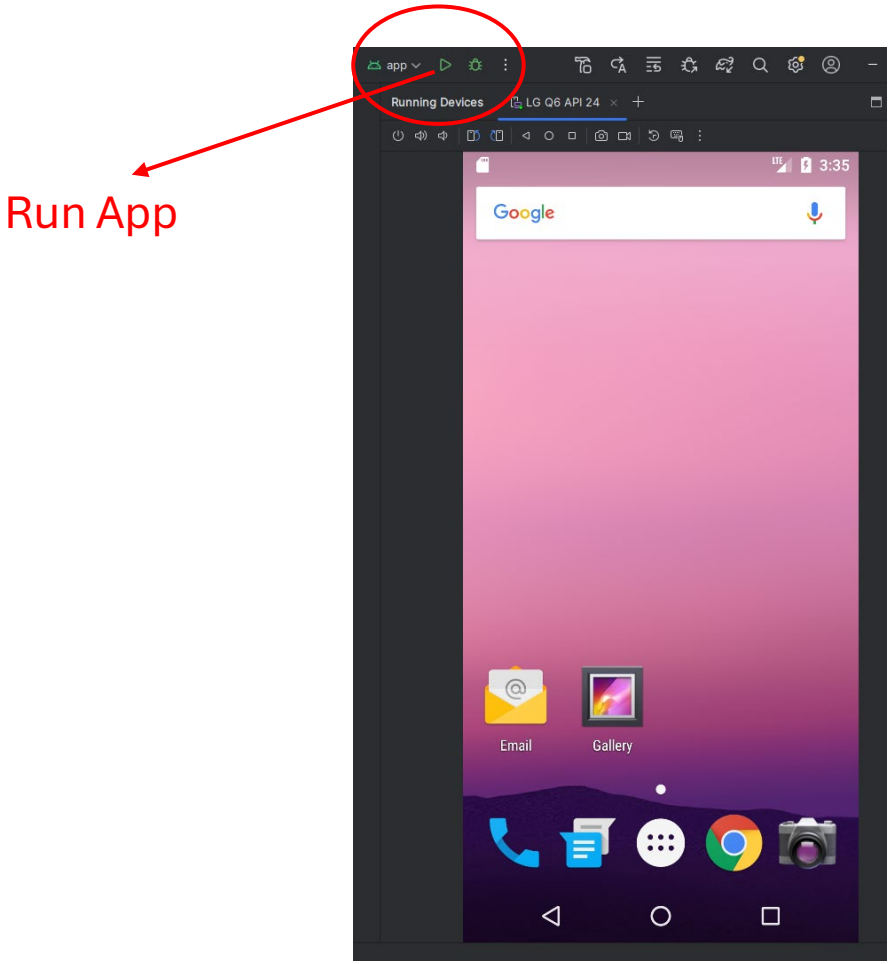
Click “Running Device”

Choose the one that you have created previously

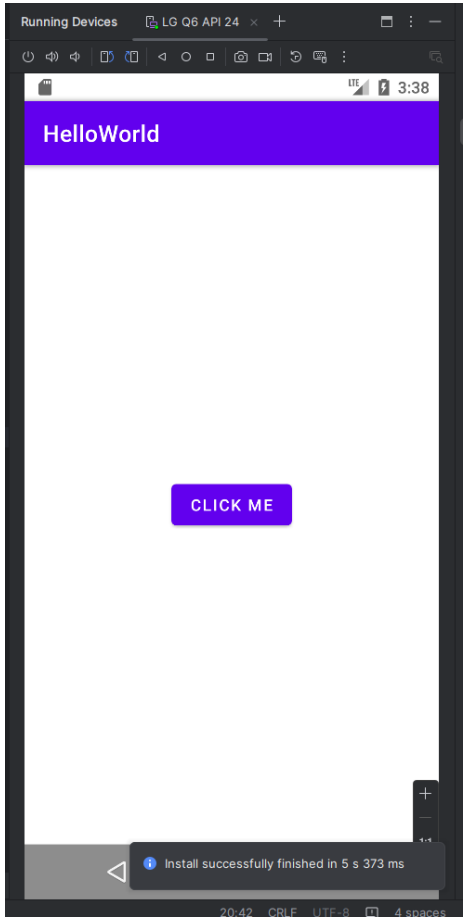


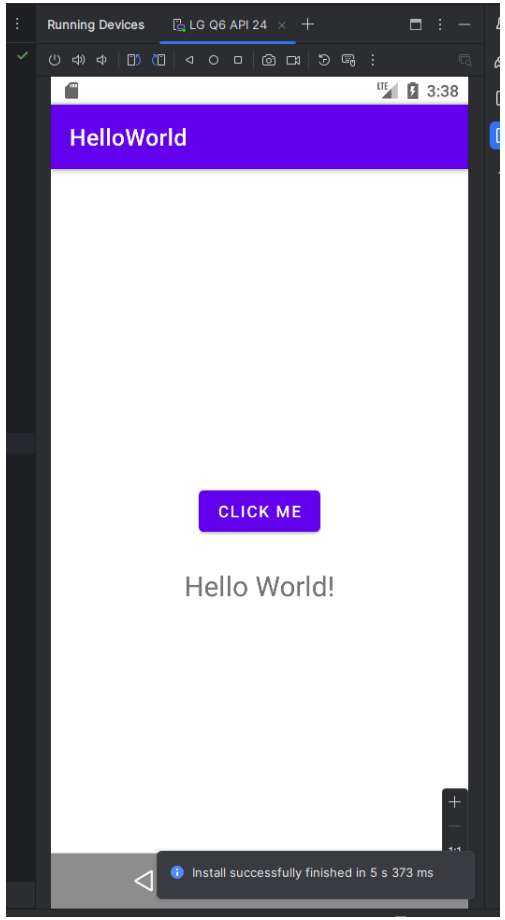
If success, the Home Screen will show up

Click “Run App” to run the app in the emulator. The build process will take some time. If success, the app will start immediately.



The final app





Sending the Data from the Emulator to the Server through Telnet

Why learning telnet?

- If you are using emulator, our app will communicate to the server through the telnet protocol

- High-level steps:
  - Start the server
  - Start the app (completed)
  - Initiate the communication between the server and app

- Installing telnet: https://chatgpt.com/share/67dbada6-8dc4-800d-847a-d7407853fcdf

A simple python server

Welcome server.py 1

server.py > ...

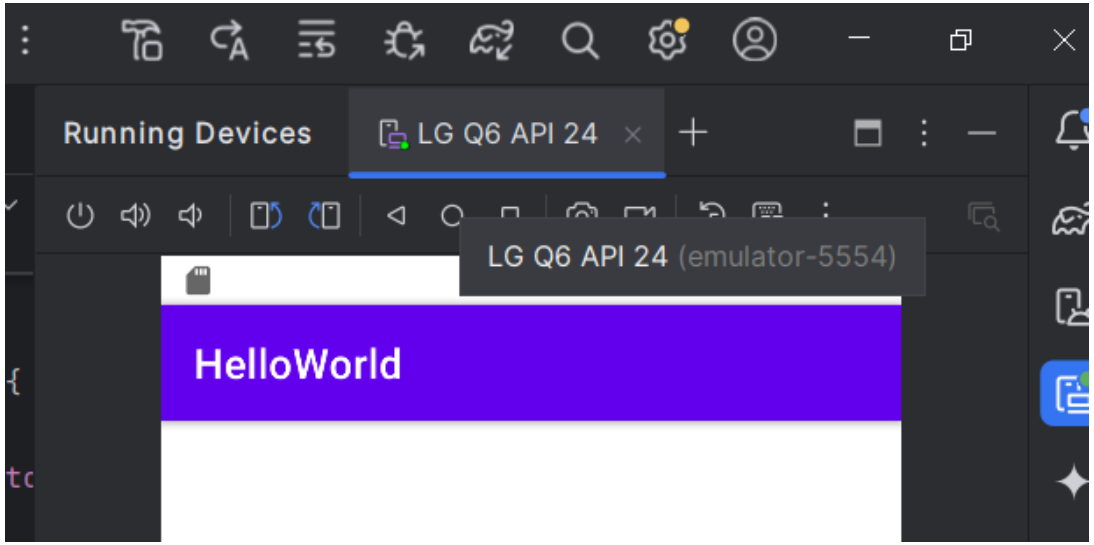
```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/hello', methods=['POST'])
def receive_hello():
    data = request.json
    message = data.get('message', 'No message received')
    print(f"Received message: {message}")
    return jsonify({"status": "success", "received": message})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Modify the **MainActivity.java**

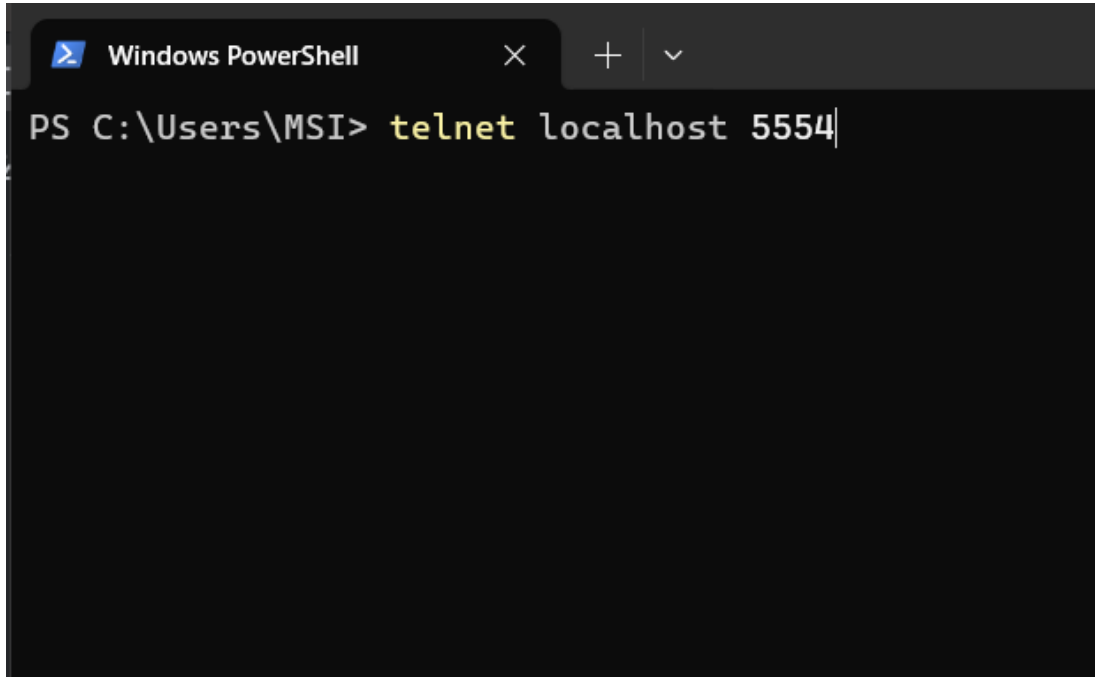
```java
// Method to send the message to our server
1 usage
private void sendHelloMessage() {
    // We need to create a new thread for network operations
    // (Android doesn't allow network calls on the main UI thread)
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                // Create a connection to our server
                URL url = new URL(SERVER_URL);
                HttpURLConnection connection = (HttpURLConnection) url.openConnection();

                // Configure the connection for a POST request
                connection.setRequestMethod("POST");
                connection.setRequestProperty("Content-Type", "application/json");
                connection.setDoOutput(true);

                // Create a simple JSON messageS
                String message = "{\"message\":\"Hello World\"}";

                // Send the data
                try (OutputStream os = connection.getOutputStream()) {
                    byte[] input = message.getBytes(StandardCharsets.UTF_8);
                    os.write(input, off: 0, input.length);
                }

                // Check if the server accepted our message
                int responseCode = connection.getResponseCode();

                // Close the connection
                connection.disconnect();

                // Show a success message (must be done on the UI thread)
                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText( context: MainActivity.this,
                                text: "Message sent to server!", Toast.LENGTH_SHORT).show();
                    }
                });

            } catch (Exception e) {
                // Silently handle any errors (no error message shown)
            }
        }
    }).start();
}
```

Setting up the telnet connection

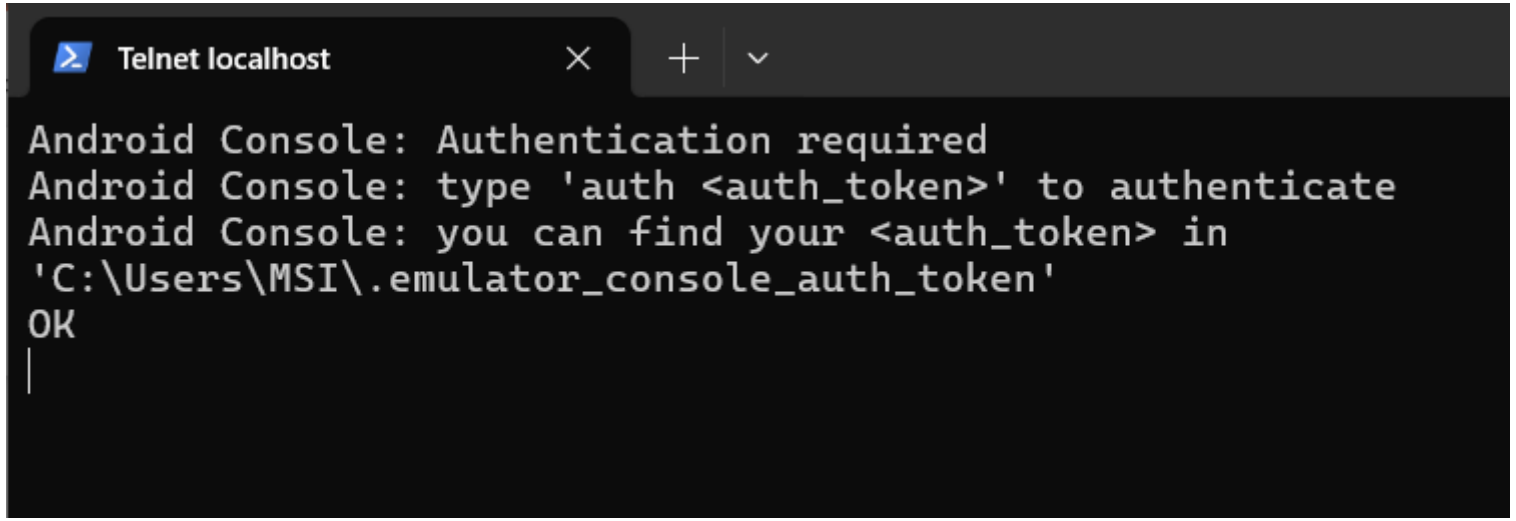
Check the port



Open the cmd and initiate telnet connection



Open the file > copy > auth {auth_token}



Add redirection: redir add tcp:5000:5554

```
    proxy
    phonenumber

Try 'help-verbose' for more description
Try 'help <command>' for command-specific help
OK
redir add tcp:5000:5554
```

Server port Emulator port

Sometimes, the cmd will response with “KO: bad redirection format, try (tcp|udp):hostport:guestport”. This is ok. Just try again until response is “OK”

```
redir add tcp:5000:5554
KO: bad redirection format, try (tcp|udp):hostport:guestport

KO: unknown command, try 'help'

KO: unknown command, try 'help'
redir add tcp:5000:5554
OK
```

Common Issues

Confirm with “redir list”

```
redir list
ipv4 tcp:5000  => 5554
OK
```

Add internet permission in the **AndroidManifest.xml** inside the **manifest** folder

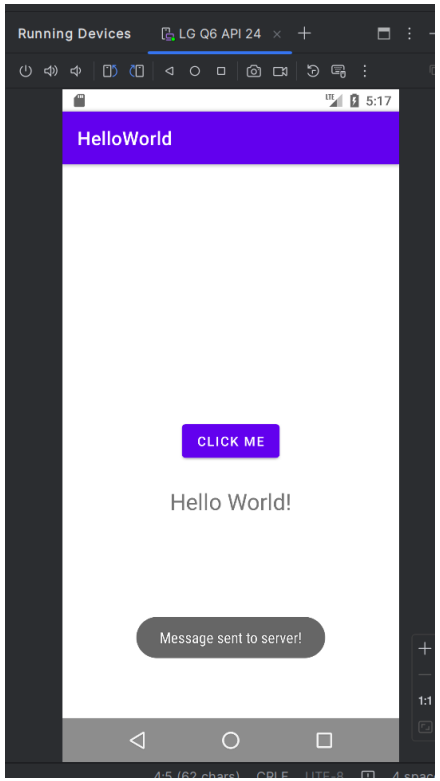
AndroidManifest.xml × MainActivity.java Bak.java

```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.HelloWorld"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Internet Permission

Test the app

The toast message will show up every time the button is clicked



The server will log the received message

```
Received message: Hello World
10.169.1.142 - - [20/Mar/2025 13:16:02] "POST /hello HTTP/1.1" 200 -
Received message: Hello World
10.169.1.142 - - [20/Mar/2025 13:16:09] "POST /hello HTTP/1.1" 200 -
Received message: Hello World
10.169.1.142 - - [20/Mar/2025 13:17:39] "POST /hello HTTP/1.1" 200 -
```

Common Issues

Build Errors

- **Issue**: Gradle sync failed **Solution**: Check internet connection, update Gradle, or invalidate caches (File > Invalidate Caches / Restart)
- **Issue**: Cannot resolve symbol 'R' **Solution**: Clean project (Build > Clean Project) or check for errors in resource files

Emulator Problems

- **Issue**: Emulator is slow **Solution**: Enable hardware acceleration in BIOS, use a lighter emulator image, or increase allocated RAM
- **Issue**: App crashes on launch **Solution**: Check Logcat for error details, verify manifest settings, ensure minimum SDK compatibility

Code-Related Issues

- **Issue**: findViewById() returns null **Solution**: Verify ID names match between XML and Java, ensure layout is loaded before finding views
- **Issue**: Button click does nothing **Solution**: Verify listener is properly implemented, check for logic errors in onClick method