Progress Report on App Development (Week 01)

-by Ritesh Kumar

An Intern in UpSkill Campus, 15/06/2023

Ritesh Kumar

I am pleased to present you with a comprehensive report on app development (Week 01), which provides an overview of the process, challenges, and best practices for successful app development. This report aims to make understanding of the key aspects of app development and making informed decisions in this domain.

**Progress Report on App Development (Week 01)**

# An Overview on App Development

In the 1st week of “6 Weeks of App Development – Project-Based Learning”, you are going to learn how to set up an Android Studio for Android App Development, How to Create a New Project in Android Studio, How to Run Your Project on a Real Device or Emulator and also you are going to develop two basic Android Apps in 1st week. In this part, you are just going to create the following two very basic android apps

1. The very famous “Hello World” Android App
2. Android app to add two numbers

## Content:

* Android Studio Setup (Windows, Mac, Linux)
* Component of Android Studio Environment
* Understanding the Designing Part
* Understanding the Coding Part
* Run Android App
* Explanation
* Output

### Challenges and Hurdles:

* **System Requirements (Windows | Mac | Linux)**

**For Windows:**

* Microsoft Windows 7/8/10 (32-bit or 64-bit)
* 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
* 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution

**For Mac:**

* Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (High Sierra)
* 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
* 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution

**For Linux:**

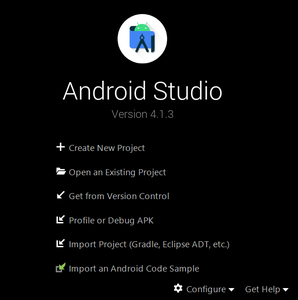
* GNOME or KDE desktop. Tested on Ubuntu 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)
* 64-bit distribution capable of running 32-bit applications
* 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
* 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution

#### Lesson Learned:

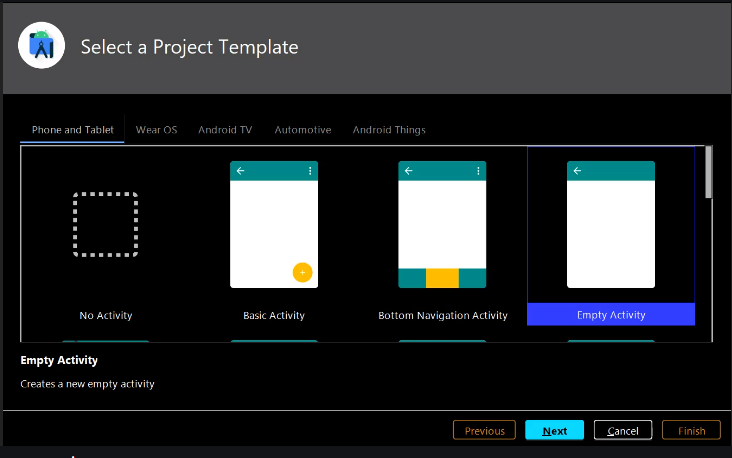
* **Android Studio Setup (Windows, Mac, Linux)**

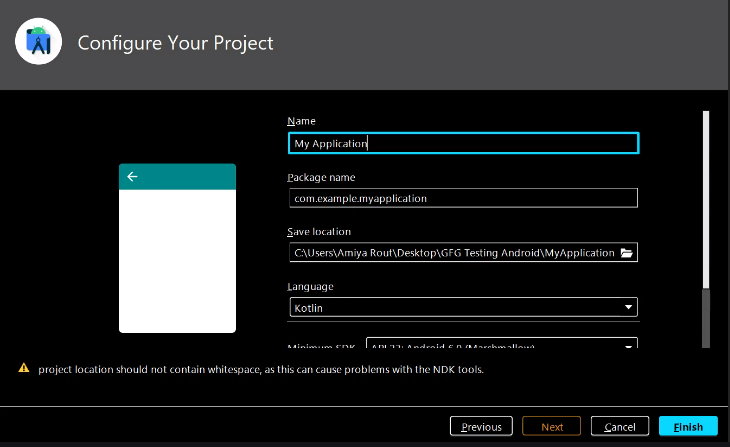
Before setting up Android Studio, let’s install it on your system.

Considering that you have successfully installed the Android Studio and you are on the screen below now.



So just click on the “**Create New Project**” and on the next screen, you can see lots of **Activities** are there. You can explore each activity, let’s choose **Empty Activity** and move further.

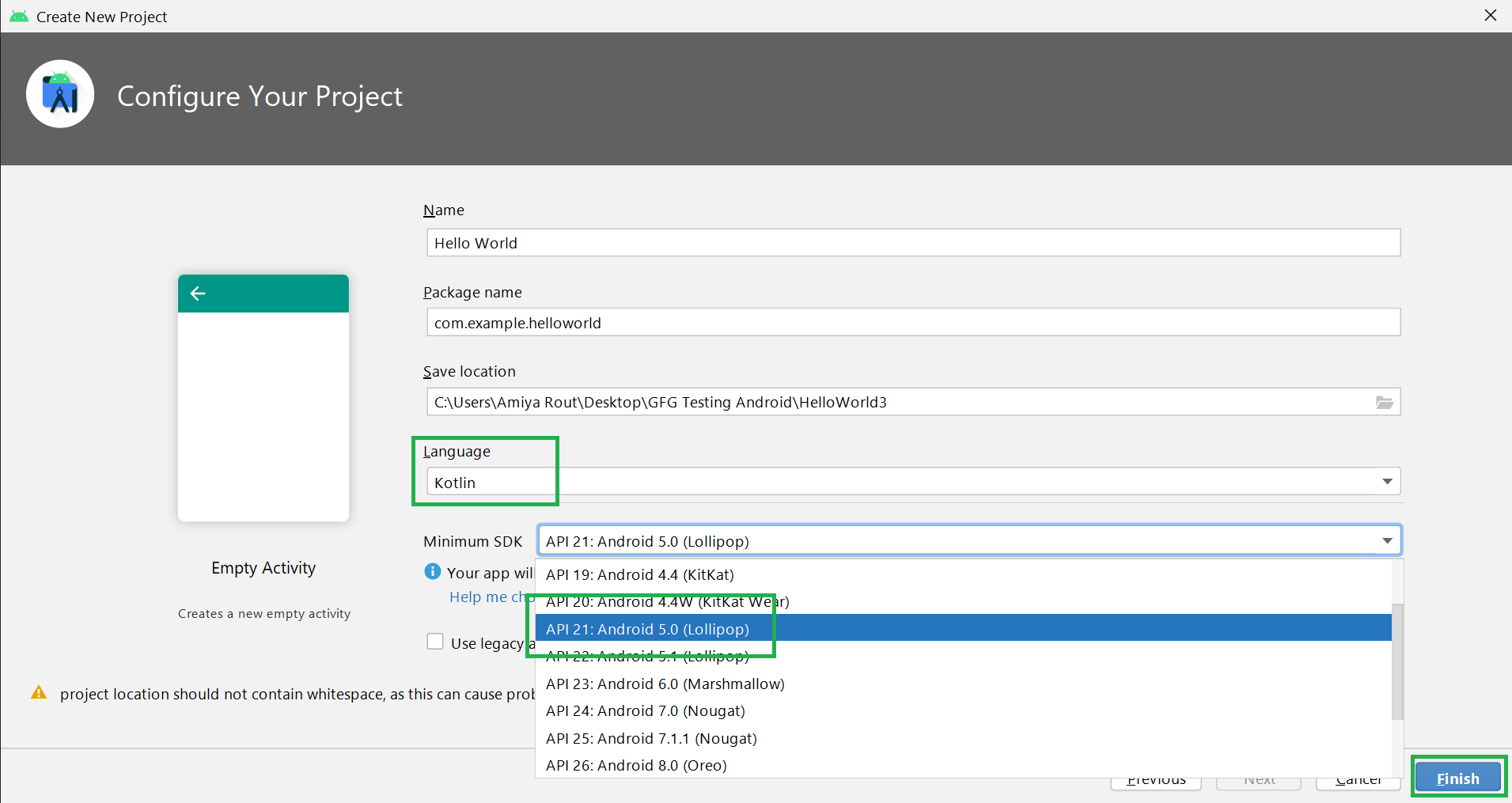




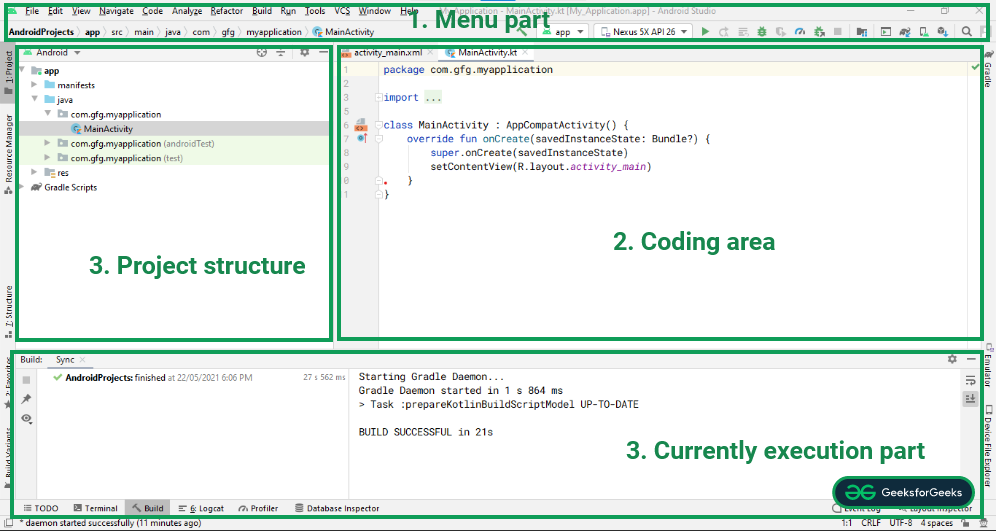
Now on the next screen

* **Name:** Write down your suitable project name.
* **Package name:** Let’s keep it by default.
* **Save location:** Choose the location where you want to save this project.
* **Language:** Choose your language between Java and Kotlin
* **Minimum SDK**(Mark this word)**:** As you know, Android released its version at regular intervals of time and they provide some new features in the latest version which are not supported by the older versions. So you have to choose which user base you want to hit. Suppose you select “API 21: Android 5.0 (Lollipop)”, then you can see “Your app will run on approximately 94.1% of devices”. Otherwise, you can click on the “Help me choose” button also.

Finally, click on the **Finish** button.



After click on the finish button, the first Android Studio project is ready. Initially, it shows the tip of the day, which gives tips to improve work efficiency. Before going to explore the Android Studio, get familiar with the initial screen which is shown below.



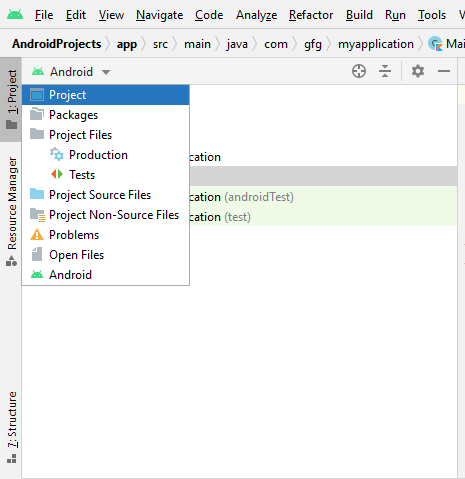
* **Component of Android Studio Environment**

**For a better understanding of the Android Studio environment, it has been divided into 4 parts:**

**1. Menu Part:** In this section, it provides options to create a new project, open an existing android studio project, a button to run the application, dropdown for selecting the desired device to run and test an application on.

**2. Coding Area:** This section provides for writing code for files like .xml, .java, .kt. It allows you to open multiple files at a time in different tabs, but it allows you to edit one file at a time.

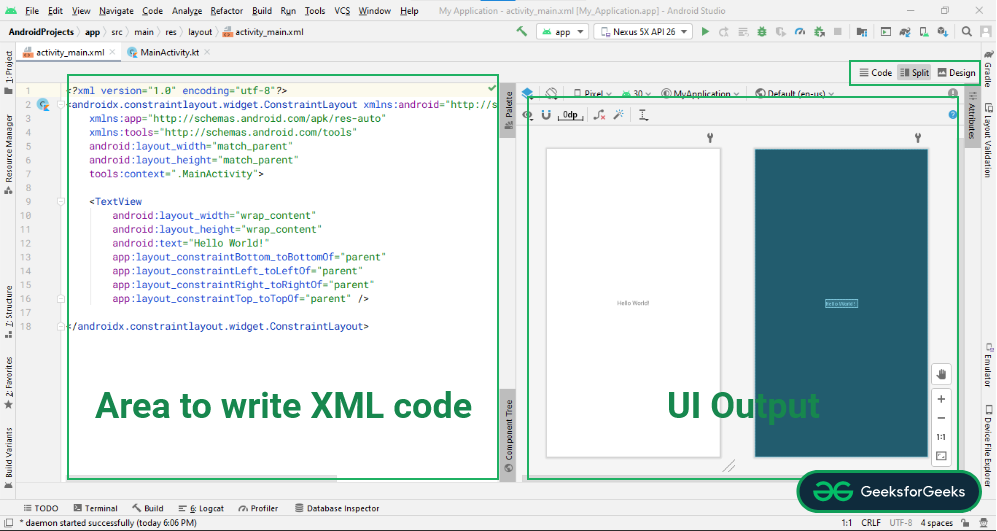
**3. Project Structure:** This area allows us to explore every file of the project. There are various views of the structure. Android is a summarized view of the project structure hierarchy and Project is a detailed view of the project hierarchy. Have a look at the following image.



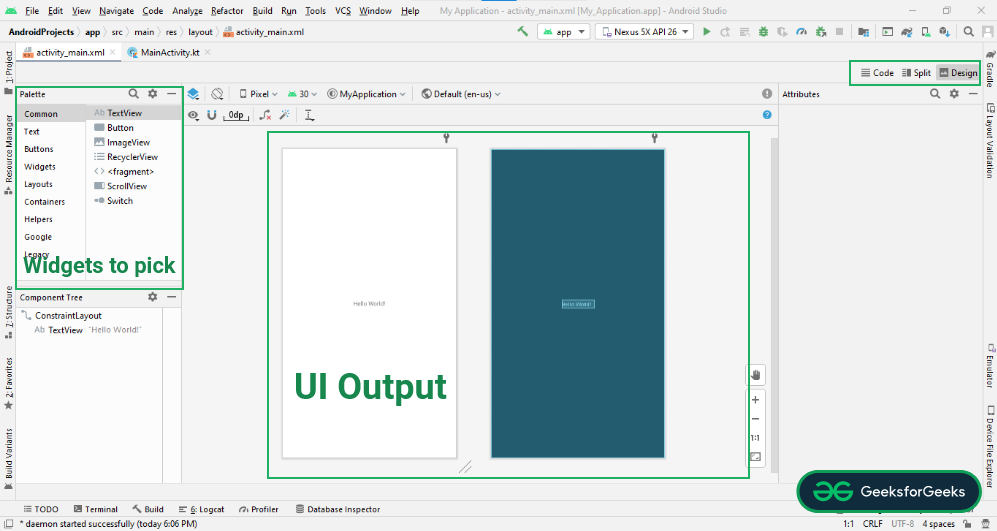
**4. Current Execution Part:**One of the most important parts of the Android Studio environment. It provides a detailed view of the current execution of the process or activity. It shows what errors, build outputs, logcat results, etc.

* **Understanding the Designing Part**

All the UI layouts of the application can be found under the res/layout directory. These are of the .xml extension, responsible for building the UIs for application. Android Studio provides two types of ways of designing UI for applications. One is Text, where one needs to design the UI by hardcoding and one is the Design section that provides a convenient drag and drop mechanism for building UI. Have a look at the following image which shows how the code is being written, it goes on to show the result in the right pane.

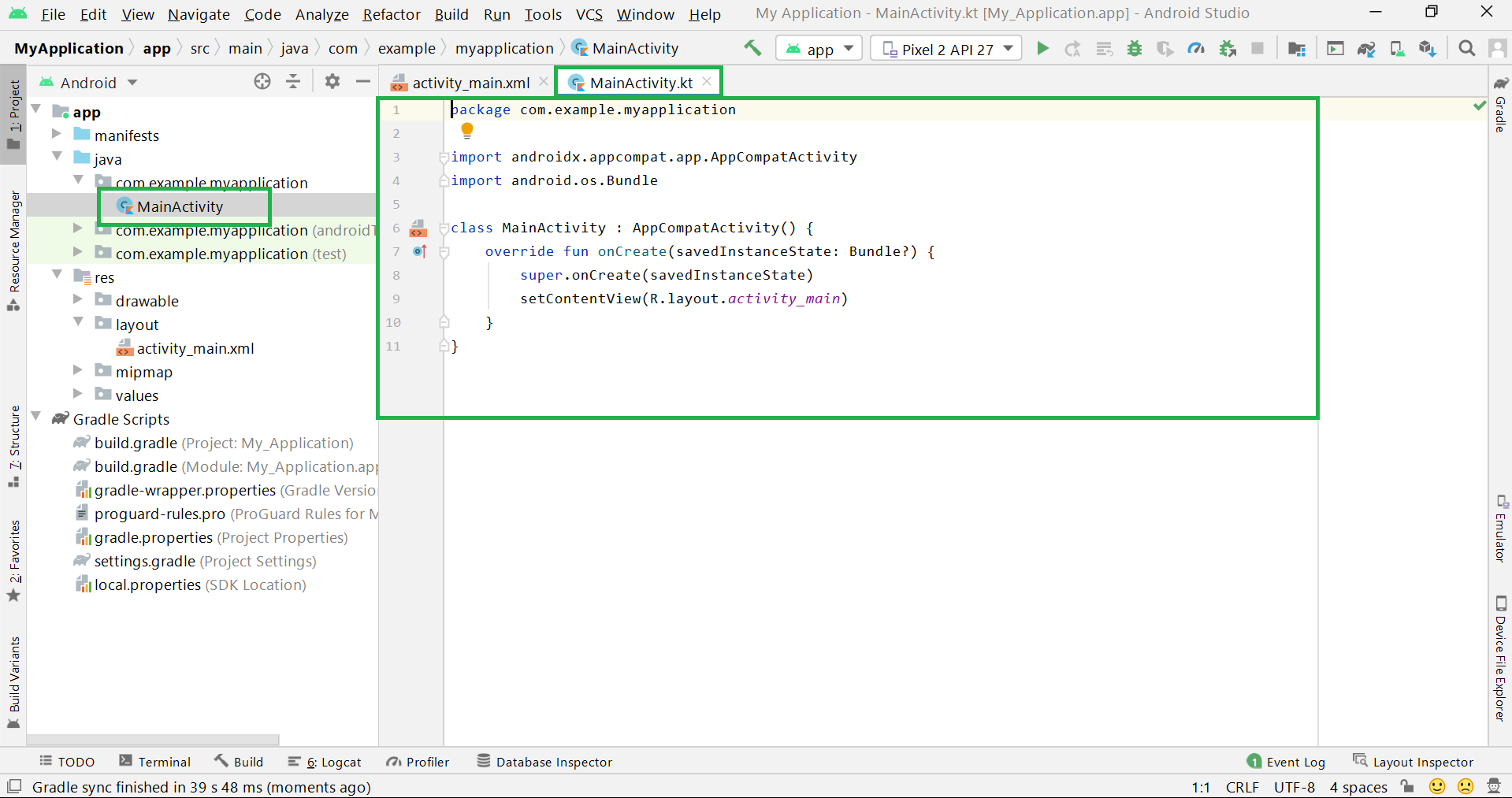


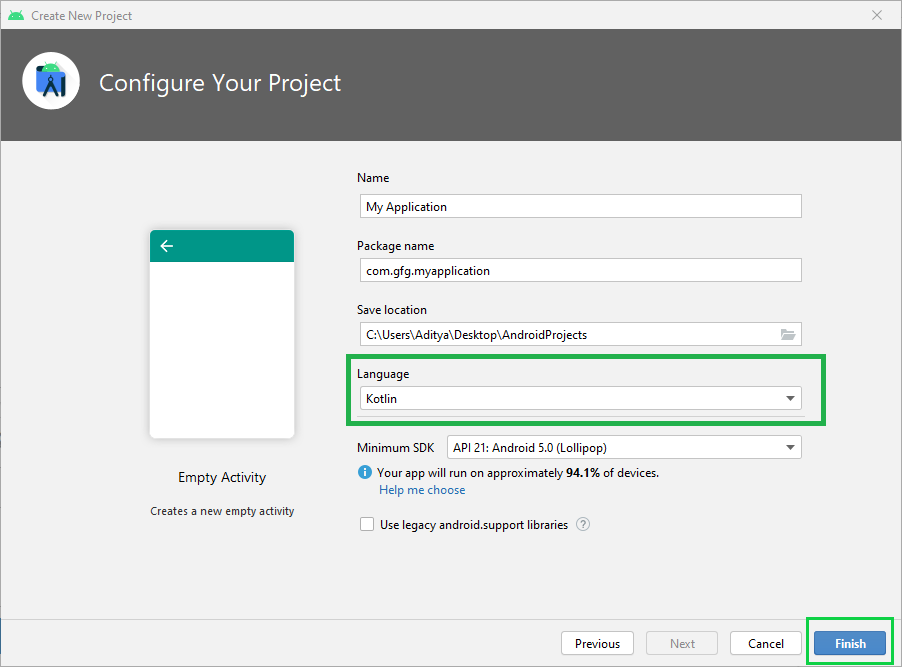
The following image shows the drag and drops environment for widgets to pick and add in the UI of the application. One can switch to the **Design**layout by clicking on the bottom left button, shown as the following.



* **Understanding the Coding Part**

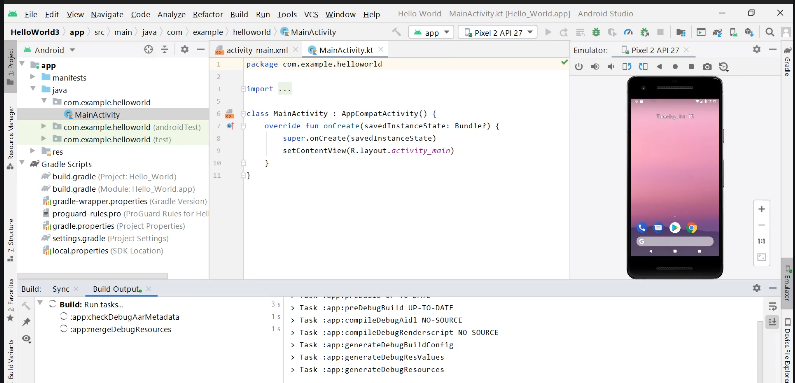
As we have understood the design part similarly, another main part is the coding part. In the image below you can see a file named **MainActivity.kt.** Yes, this is the file where you write the backend part. The logic part. For example, you want to add two numbers. So, after entering two numbers in the EditText, when the user clicks on the Button, it will give the output as the sum of two numbers. So to calculate the sum, you have to write the logic in its part.

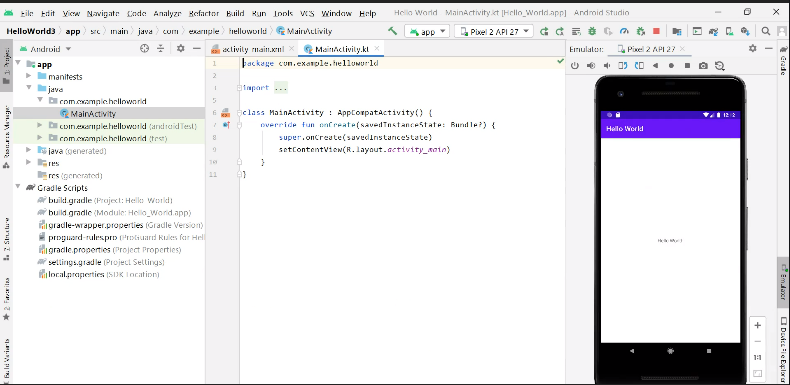




* **Run Android App**

To run an application there are two ways to do it. One is using an emulator, another is using a physical device by connecting it through a USB cable. Refer to [How to install Android Virtual Device(AVD)](https://www.geeksforgeeks.org/how-to-install-android-virtual-deviceavd/) how to set up an android virtual device or refer to [How to Run the Android App on a Real Device?](https://www.geeksforgeeks.org/how-to-run-the-android-app-on-a-real-device/) setting up a real physical android device and running applications on it. Have a look at the following video on how to run your very famous “**Hello World Android App**” in your Android Emulator.



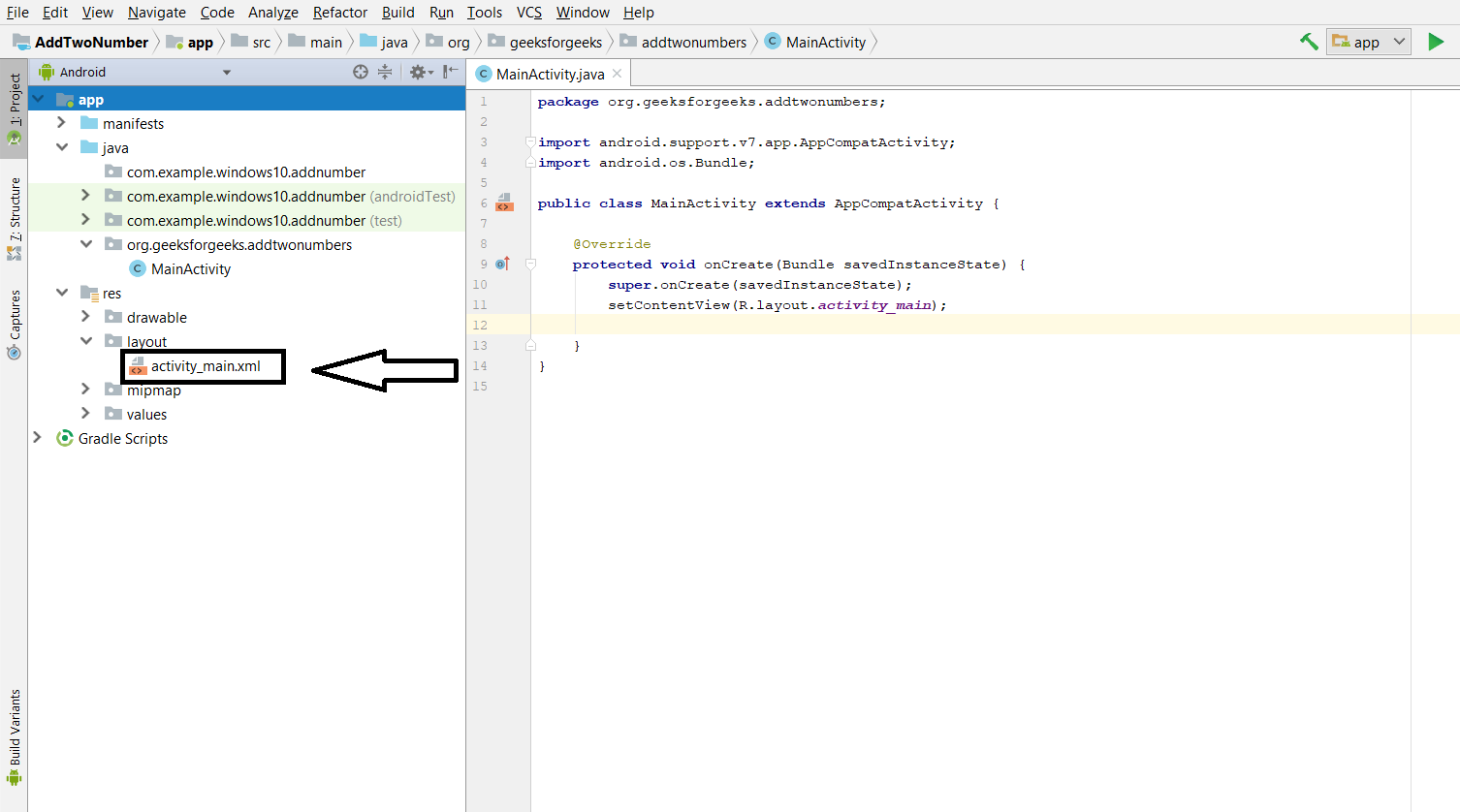


Yes!! You have successfully built your first Android App. Now let’s built the second one “**Android App to add two numbers**“

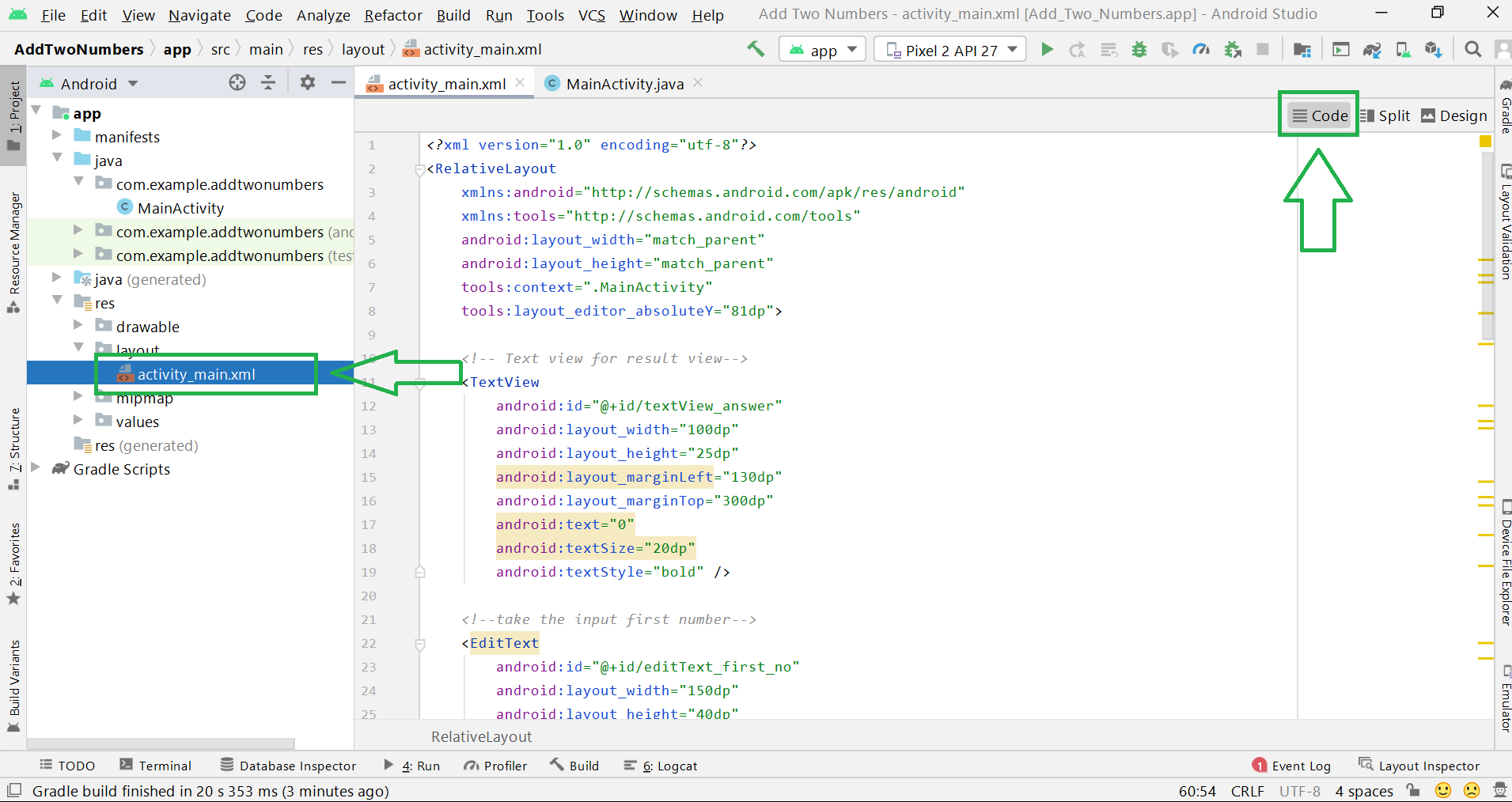
**Step By Step Implementation of Project 1: Android App to Add Two Numbers**

**Step 1:** Similarly, create a new project but this time chooses **Java**as the programming language. Choose the name of the project by yourself.

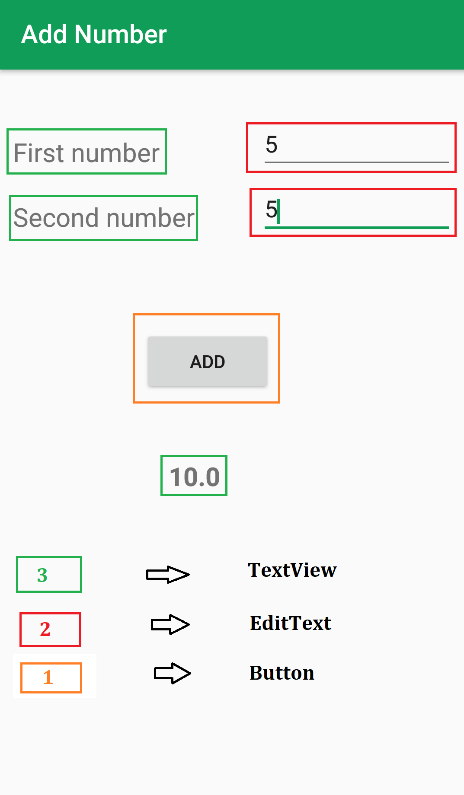
**Step 2:** First of all go to the XML file



Now go to the **Code** section and write the code for adding 3 **TextView**, 2 **EditText**, and 1 **Button**and Assign an ID to each component. Assign margin-top, left, right for the location.



So we are going to develop the UI something like that



Below is the code for the **activity\_main.xml** file.

**Xml**

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

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

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

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    tools:context=".MainActivity"

    tools:layout\_editor\_absoluteY="81dp">

    <!-- for message input first number -->

    <**TextView**

        android:id="@+id/textView\_first\_no"

        android:layout\_width="150dp"

        android:layout\_height="25dp"

        android:layout\_marginLeft="10dp"

        android:layout\_marginTop="50dp"

        android:text="First number"

        android:textSize="20dp" />

    <!-- take the input first number -->

    <**EditText**

        android:id="@+id/editText\_first\_no"

        android:layout\_width="150dp"

        android:layout\_height="40dp"

        android:layout\_marginLeft="200dp"

        android:layout\_marginTop="40dp"

        android:inputType="number" />

    <!-- for message input second number -->

    <**TextView**

        android:id="@+id/textView\_second\_no"

        android:layout\_width="150dp"

        android:layout\_height="25dp"

        android:layout\_marginLeft="10dp"

        android:layout\_marginTop="100dp"

        android:text="Second number"

        android:textSize="20dp" />

    <!-- take input for second number -->

    <**EditText**

        android:id="@+id/editText\_second\_no"

        android:layout\_width="150dp"

        android:layout\_height="40dp"

        android:layout\_marginLeft="200dp"

        android:layout\_marginTop="90dp"

        android:inputType="number"

        tools:ignore="MissingConstraints" />

    <!-- button for run add logic and view result -->

    <**Button**

        android:id="@+id/add\_button"

        android:layout\_width="100dp"

        android:layout\_height="50dp"

        android:layout\_marginLeft="110dp"

        android:layout\_marginTop="200dp"

        android:text="ADD" />

    <!-- Text view for result view-->

    <**TextView**

        android:id="@+id/textView\_answer"

        android:layout\_width="100dp"

        android:layout\_height="25dp"

        android:layout\_marginLeft="130dp"

        android:layout\_marginTop="300dp"

        android:text="0"

        android:textSize="20dp"

        android:textStyle="bold" />

</**RelativeLayout**>

**Step 3: Working with the MainActivity.java file**

 In this file, we are going to write the logic for adding two numbers. Let’s have a look at the following code.

**Java**

**package** com.example.addtwonumbers;

// Each new activity has its own layout and Java files,

// here we build the logic for adding two number

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.TextView;

**import** androidx.appcompat.app.AppCompatActivity;

**public** **class** MainActivity **extends** AppCompatActivity {

    // define the global variable

    // variable number1, number2

    // for input input number

    // Add\_button, result textView

    EditText number1;

    EditText number2;

    Button Add\_button;

    TextView result;

**int** ans = 0;

    @Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // by ID we can use each component which id is assign in xml file

        number1 = (EditText) findViewById(R.id.editText\_first\_no);

        number2 = (EditText) findViewById(R.id.editText\_second\_no);

        Add\_button = (Button) findViewById(R.id.add\_button);

        result = (TextView) findViewById(R.id.textView\_answer);

        // Add\_button add clicklistener

        Add\_button.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

                // num1 or num2 double type

                // get data which is in edittext, convert it to string

                // using parse Double convert it to Double type

**double** num1 = Double.parseDouble(number1.getText().toString());

**double** num2 = Double.parseDouble(number2.getText().toString());

                // add both number and store it to sum

**double** sum = num1 + num2;

                // set it ot result textview

                result.setText(Double.toString(sum));

            }

        });

    }

}

* **Explanation**
* The package name of your Android app.

**package com.example.addtwonumbers;**

* List of libraries used in your app.

**import android.os.Bundle;**

**import android.view.View;**

**import android.widget.Button;**

**import android.widget.EditText;**

**import android.widget.TextView;**

**import androidx.appcompat.app.AppCompatActivity;**

* Variable **number1, number2** for input number. Similarly **add\_button**variable is for our Button used in the XML code and the **result**variable is for TextView.

**EditText number1;**

**EditText number2;**

**Button add\_button;**

**TextView result;**

* Remember you have assigned an ID for each component in the XML file. Yes, they are going to use it here. By ID we can use each component which id is assigned in the XML file like the following.

**number1 = (EditText) findViewById(R.id.editText\_first\_no);**

**number2 = (EditText) findViewById(R.id.editText\_second\_no);**

**add\_button = (Button) findViewById(R.id.add\_button);**

**result = (TextView) findViewById(R.id.textView\_answer);**

* And at last, we add a setOnClickListener() to our Button. So when the user clicks on the Button this will give us the Output. And inside that, we have written simple logic to add those numbers. And also setText() for our output TextView. As simple as that. That’s it.

**add\_button.setOnClickListener(new View.OnClickListener() {**

**public void onClick(View v) {**

**double num1 = Double.parseDouble(number1.getText().toString());**

**double num2 = Double.parseDouble(number2.getText().toString());**

**double sum = num1 + num2;**

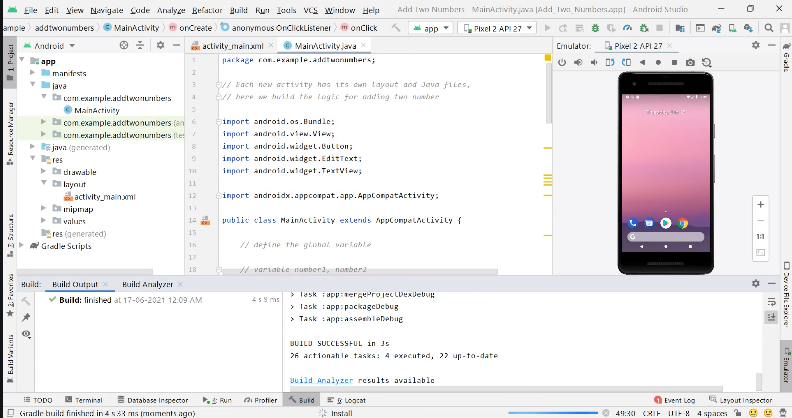
**result.setText(Double.toString(sum));**

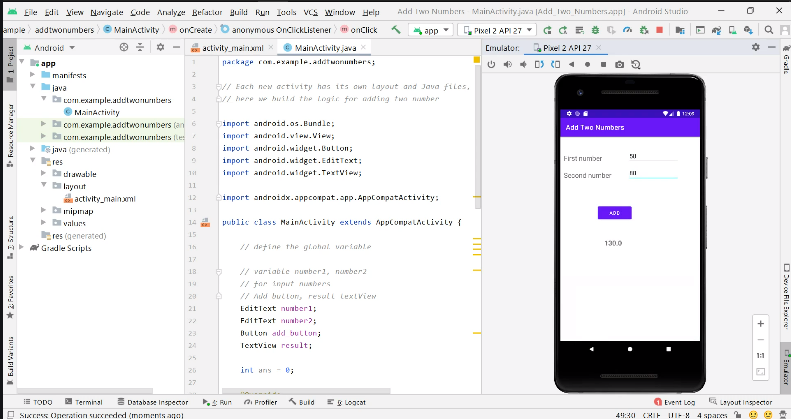
**}**

**});**

* Yes, you have successfully developed an Android App which takes two numbers from the users and adds them. Now run the app as before and you can see the output as follows.

* **Output**





##### **References:**

* <https://developer.android.com/studio>
* https://developer.android.com/
* <https://www.geeksforgeeks.org/>

Please note that this report provides a general overview of app development, and specific recommendations or strategies should be tailored to your organization's unique requirements and goals. Should you require further assistance or have any questions, please do not hesitate to contact me.

Thank you for the opportunity to prepare this report, and I hope it proves valuable in your app development endeavors. I look forward to discussing this topic further and assisting you in your future projects.

Sincerely,

Ritesh Kumar (9330771068)

Email: kumarritesh6769@gmail.com