1. Aim: Give the steps to setup Android Environment using Android Studio IDE
Resources Required: A computer system with Android Studio IDE Installed.
Procedure:

To make the android development environment setup process simple Google introduced a new android IDE called **Android Studio**. The **Android Studio** will contain all the required components like Eclipse IDE, Eclipse Plugin and Android SDK so we don't need to download the components separately.

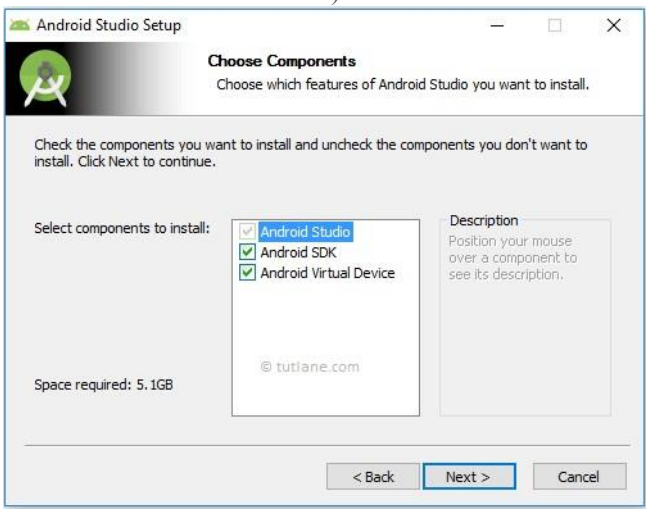
Android Studio is the official IDE for android development and it's based on **IntelliJ IDEA** software. It's available for Windows, MAC and LINUX operating systems.

In this tutorial, we are going to explain how to install android studio on a windows machine which is having windows 10 operating system.

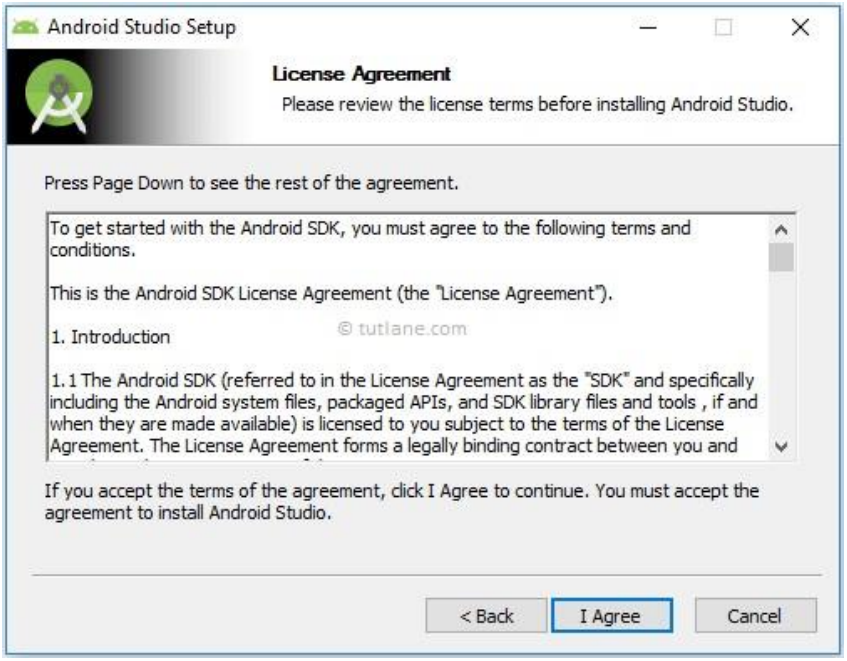
Download the latest version of Android Studio from the above URL and launch **Android Studio.exe** file by double-clicking on it.

The initial android studio setup screen will open like as shown below in that click **Next** to continue for further steps of environment setup.

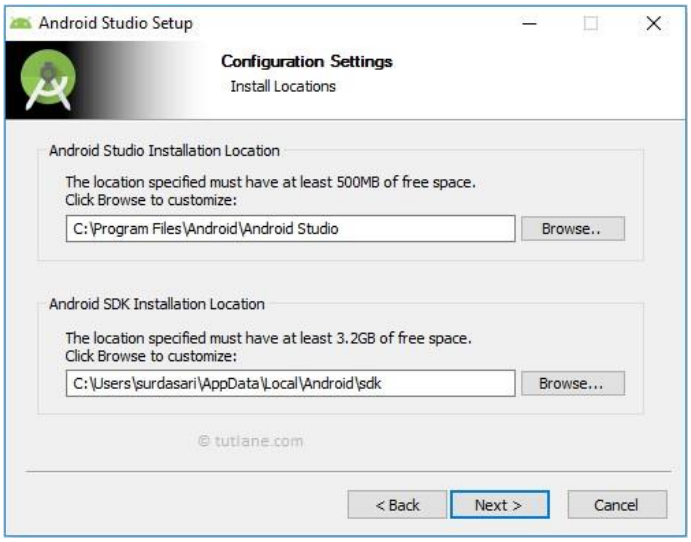
Now we need to select the required components to set up an android environment. Here we selected all three components (**Android Studio**, **Android SDK** and **Android Virtual Device**) and click **Next** like as shown below.



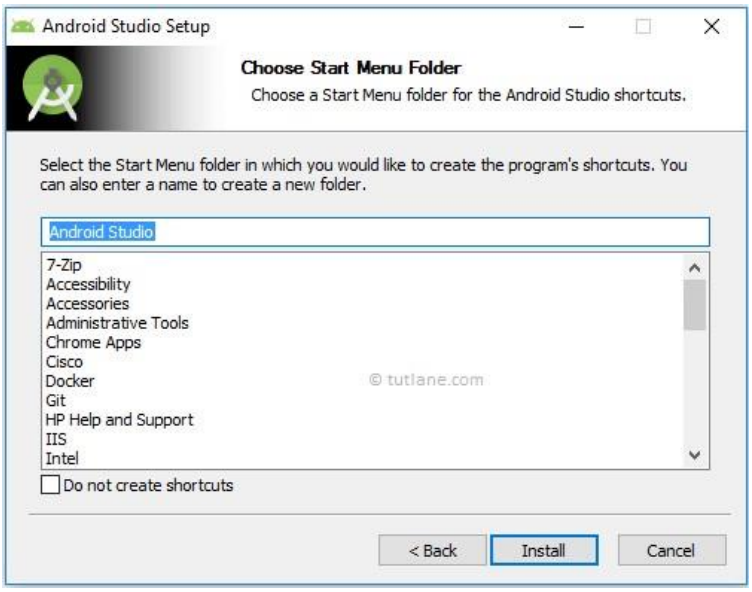
Now we need to agree on the License agreements to proceed further, click on **I Agree** button like a shown below.



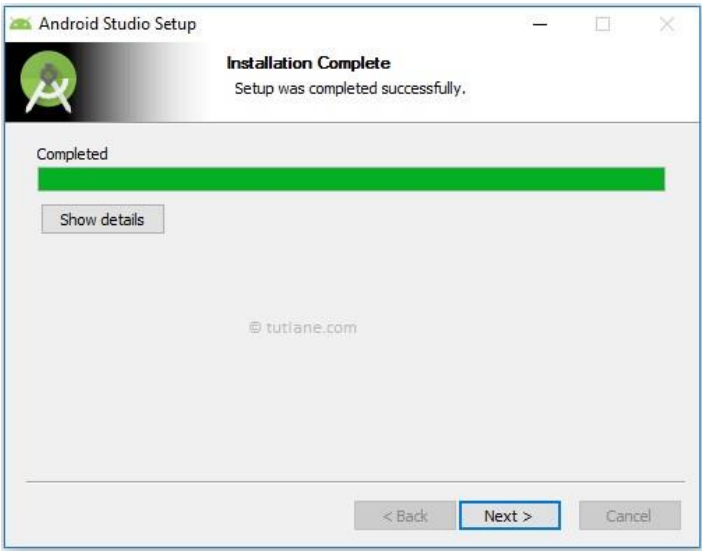
Now we need to specify the local machine drive location to install Android Studio and Android SDK. After selecting the location path to install the required components, click **Next** like as shown below.



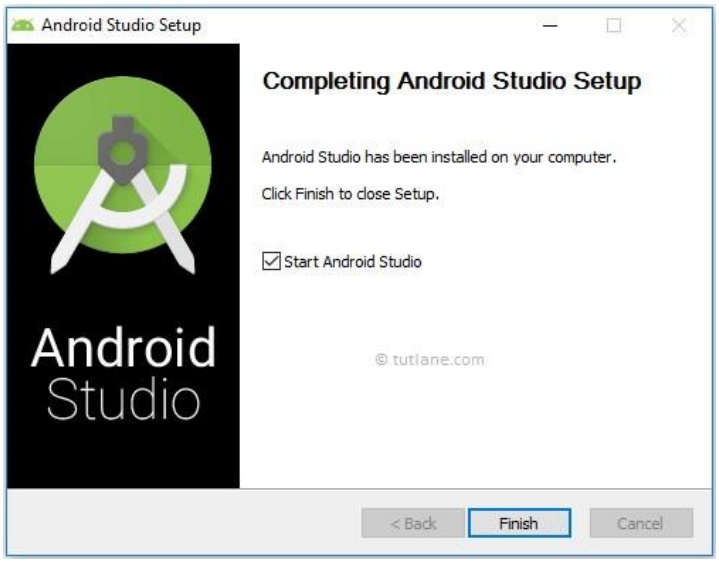
Now select the start menu folder to create a shortcut for android studio and click **Install** like as shown below.



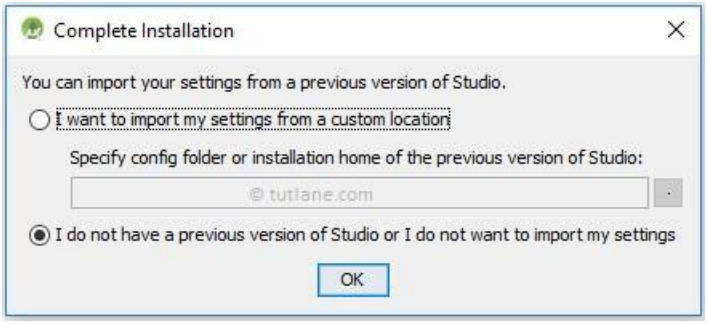
Once we click the Install button the installation process will start and click **Next** after completion of installation like as shown below.



After that, it will show installation completion wizard in that click **Finish** to launch an android studio like as shown below.

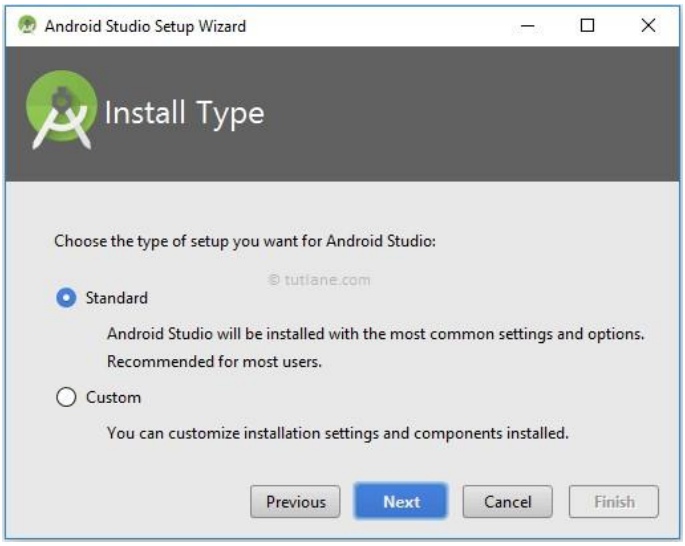


While launching **Android Studio** it will give you an option to import settings from the previous version of the studio. In case if you don't have any previous version, select the second option and click **OK** like as shown below.

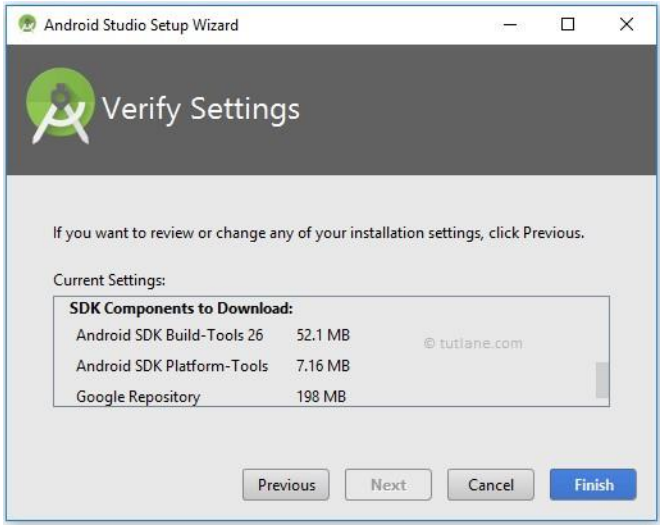


Now android studio will open a welcome wizard window in that click **Next** to validate our current Android SDK and development environment setup like as shown below.

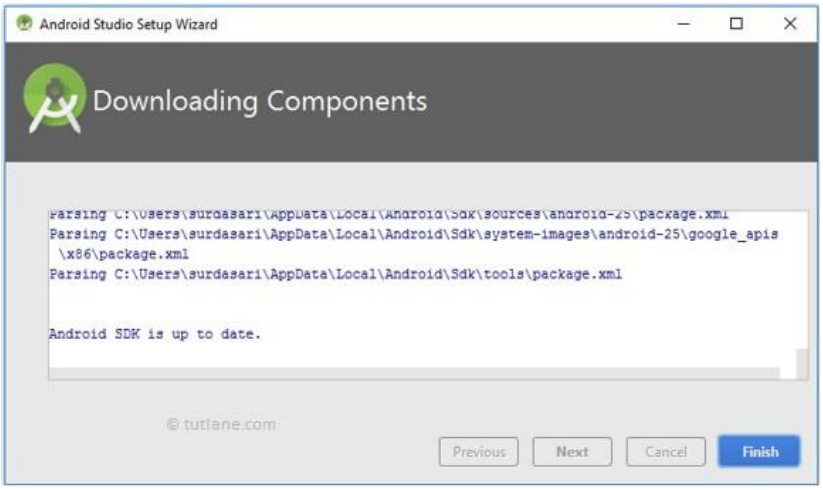
Now select a **Standard** installation type and click **Next** to install common settings and options like as shown below.



Now verify settings and click **Finish** to complete the android studio setup process like as shown below.



After completion of the installation of the required components click on **Finish** like as shown below.



After completion of all required components installation, we will be able to see the Android Studio welcome window like as shown below.

This is how we can set up an Android development environment on a windows machine which is having windows 10 operating system using android studio IDE.

2. Aim: Give the steps to setup Android Environment using Eclipse IDE **Resources required:**

1. Install the JDK
2. Download and install the Eclipse for developing android application
3. Download and Install the android SDK
4. Intall the ADT plugin for eclipse
5. Configure the ADT plugin
6. Create the AVD
7. Create the hello android application

1) Install the Java Development Kit (JDK)

For creating android application, JDK must be installed if you are developing the android application with Java language. download the JDK

2) Download and install the Eclipse IDE

For developing the android application using eclipse IDE, you need to install the Eclipse. you can download it from this location download the Eclipse. Eclipse classic version is recommended but we are using the Eclipse IDE for JavaEE Developers.

3) Download and install the android SDK

First of all, download the android SDK. In this example we have installed the android SDK for windows (.exe version).

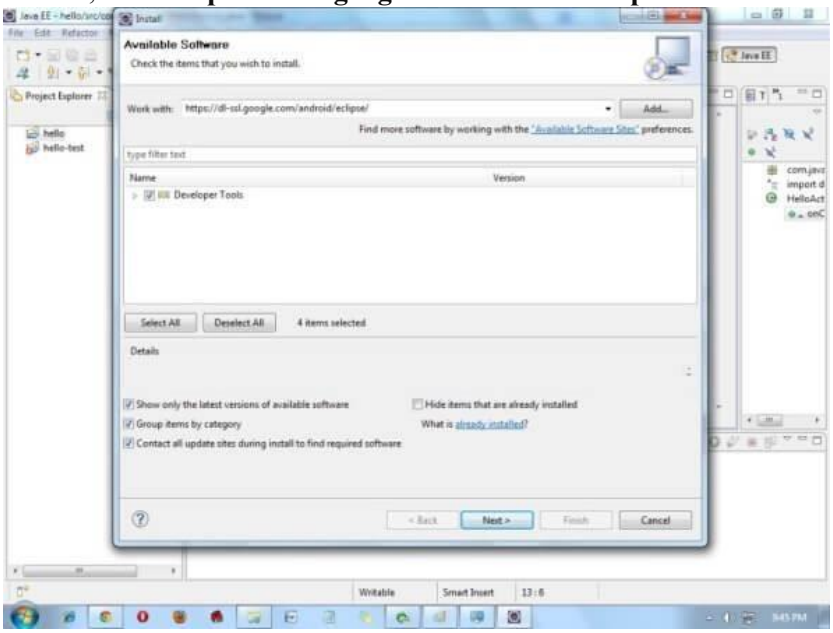
Now double click on the exe file, it will be installed. I am using the android 2.2 version here.

4) Download the ADT plugin for eclipse

ADT (Android Development Tools) is required for developing the android application in the eclipse IDE. It is the plugin for Eclipse IDE that is designed to provide the integrated environment.

For downloading the ADT, you need to follow these steps:
1) Start the eclipse IDE, then select **Help** > **Install new software...**

2) In the **work with** combo box, write **https://dl-ssl.google.com/android/eclipse/**



3) select the checkbox next to Developer Tools and click next

4) You will see, a list of tools to be downloaded here, click next

5) click finish

6) After completing the installation, restart the eclipse IDE **5) Configuring the ADT plugin**

After the installing ADT plugin, now tell the eclipse IDE for your android SDK location. To do so:

1. Select the **Window menu > preferences**
2. Now select the android from the left panel. Here you may see a dialog box asking if you want to send the statistics to the google. Click **proceed**.
3. Click on the browse button and locate your SDK directory e.g. my SDK location is C:\Program Files\Android\android-sdk .
4. Click the apply button then OK.

6) Create an Android Virtual Device (AVD)

For running the android application in the Android Emulator, you need to create and AVD. For creating the AVD:

1. Select the **Window menu > AVD Manager** 2.

Click on the **new** button, to create the AVD

3. Now a dialog appears, write the AVD name e.g. myavd. Now choose the target android version e.g. android2.2.

4. click the **create AVD**

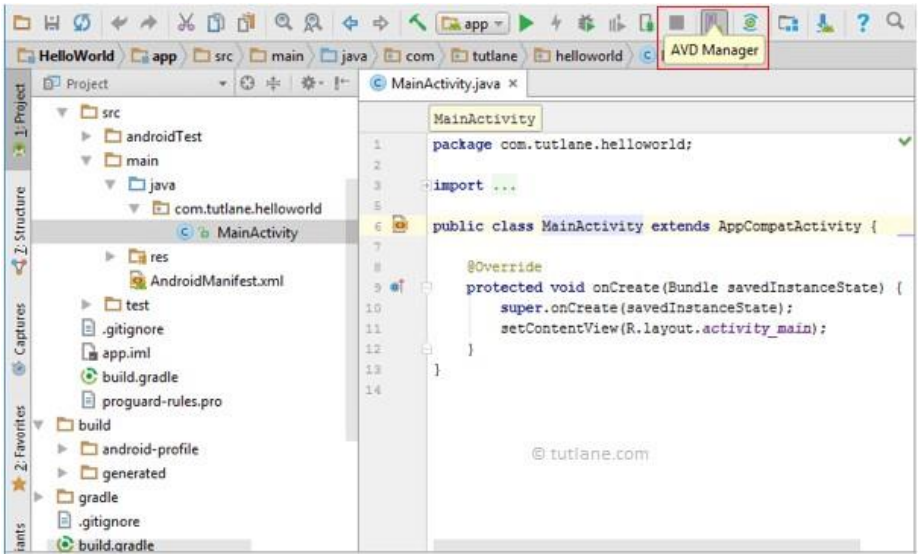
7) create and run the simple android example

Visit the next page to create first android application.

3. Aim: Give the steps to create Android Virtual Device (AVD)

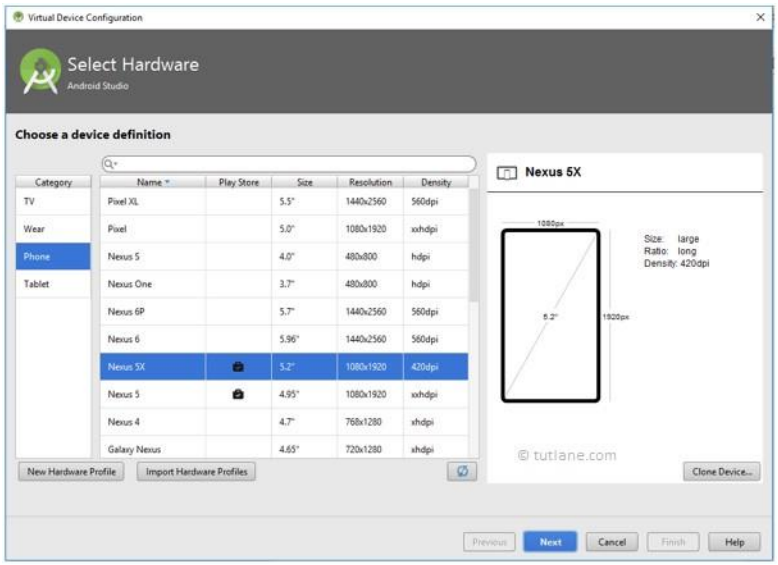
Resources: A computer with Android studio IDE and Eclipse IDE installed

Procedure: To test our android application we should have an Android Virtual Device (AVD). We can create virtual device by click on AVD Manager like as shown below.

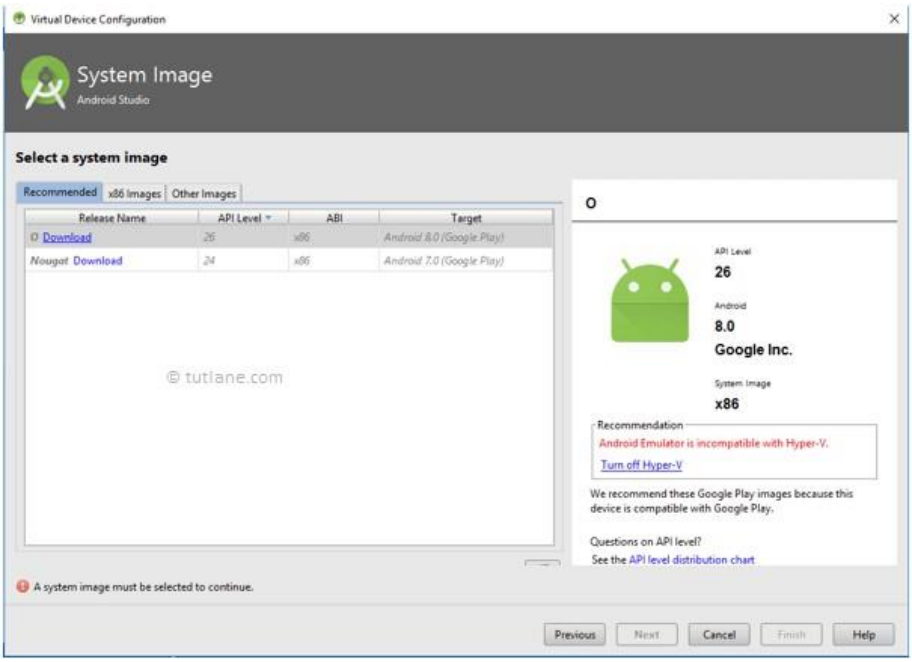


When we click on AVD Manager, a new window will open in that click on Create Virtual Device like as shown below**.**

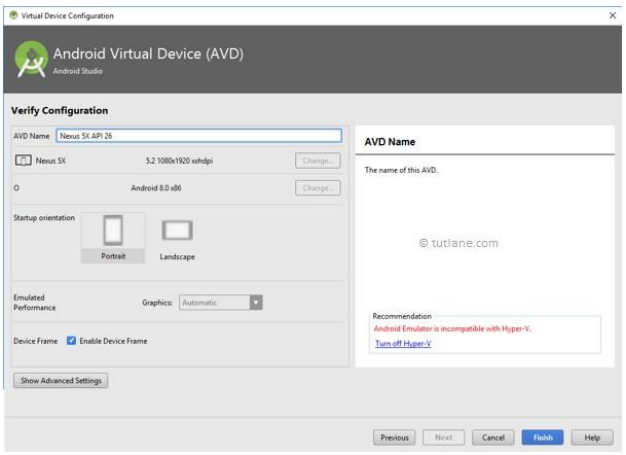
Now select the required device type and Click Next to create a virtual device like as shown below.



Now we need to download and select the system image and click Next like as shown below.

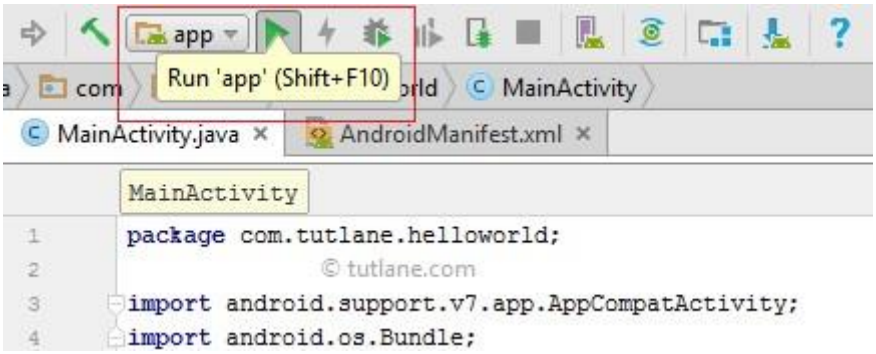


Now verify the configuration of android virtual device (AVD) and click Finish like as shown below.

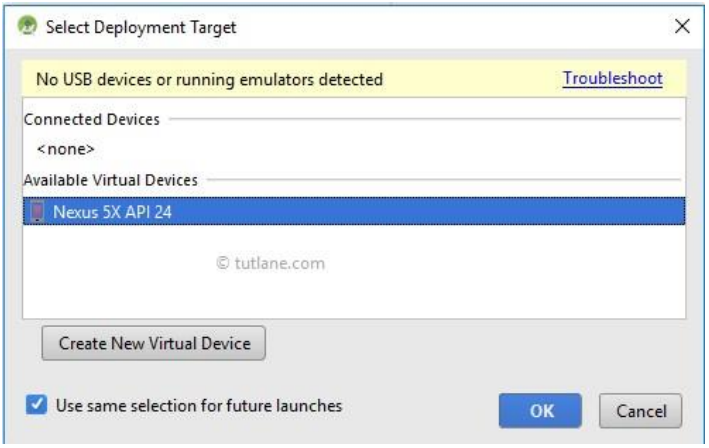


This is how we need to add android virtual device (AVD) in an android studio to test our android applications. Once we are done with the setup of android virtual device in an android studio, create a sample application in the android studio and run the app using AVD manager. In case if you are not aware of creating an app, check this Android Hello World App. Run Android Application

To run android applications, we need to click on Run button or press Shift + F10 like as shown below



After clicking on play button new window will open in that select Android Virtual Device (AVD) and click OK like as shown below.



This is how we can setup an android virtual device (AVD) emulator in android studio to replicate the functionality of real android devices.

4. Aim: Write about Android project structure **Answer:**

An Android project consists of several directories and files that contain the code, resources, and manifest file for your app. If you're working on an Android project, it's important to familiarize

yourself with this structure so that you know where to find the files you need and how to properly organize your own files.

The project structure for an Android application is as follows:

1. **app:** This directory contains the code, resources, and manifest file for your app.
2. **manifests:** This directory contains the AndroidManifest.xml file, which is the central configuration file for your app. It specifies the app's package name, the minimum required Android version, the components that are included in the app, and any permissions that the app requires.
3. **src:** This directory contains the source code files for your project, including the main activity class and any other Java classes you've created.
4. **res:** This directory contains all the resources for your project, including layouts, drawables, and strings. It is organized into subdirectories based on the type of resource. For example, the drawable directory contains image files, and the layout directory contains XML files that define the layout of your app's user interface.

The res directory in an Android project contains resources that are used by the app, such as layouts, drawables, and strings. This directory is organized into several subdirectories, each of which contains a specific type of resource. Here is a list of the common subdirectories found in the res directory:

drawable: This directory contains image files and other drawable resources.
layout: This directory contains XML files that define the layout of the user interface for your app. **values**: This directory contains XML files that define various types of values, such as strings, dimensions, and colors. **menu**: This directory contains XML files that define the menus used in your app. **mipmap**: This directory contains the app launcher icons for different densities. **anim**: This directory contains XML files that define property animations.
raw: This directory contains raw files that can be accessed by the app, such as audio or video files.
xml: This directory can contain any XML files that are used by the app.

5. **assets:** This directory can be used to store any files that your app needs to access, but that are not compiled into the APK file.
6. **libs:** This directory contains any third-party libraries that your app depends on.
7. **build:** This directory contains the files that are generated by the Android build system, such as the APK file that is used to install your app on a device.
   build.gradle: This file is used to configure the build for your app. It specifies the dependencies for your app, as well as any custom build options that you need.
8. **proguard-rules.pro:** This file is used to configure ProGuard, which is a tool that is used to shrink and optimize your app's code.

5. Aim: Develop an android application to display "hello world" **Resources required:** A desktop with Android studio IDE installed **Procedure:**
Open the android studio and that will be like as shown below.

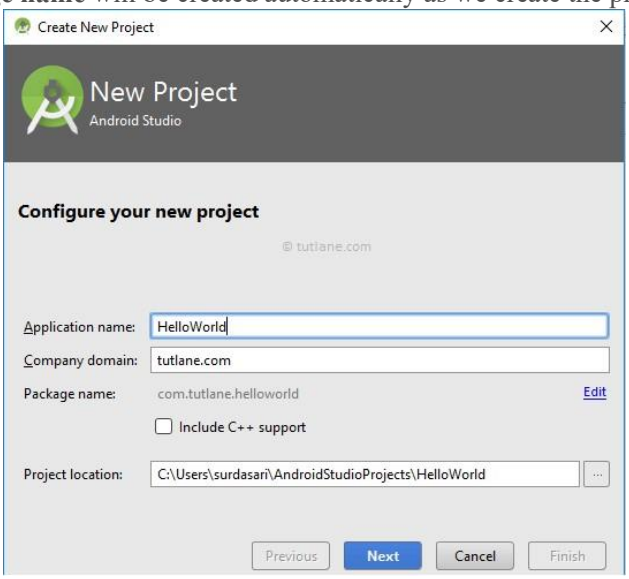
Here we're going to choose the **New Project** option because we haven't created any other project and we need to create a new one. So, we will select the New Project from the given options.

However, we can choose **Import Project** if we'd like to import a project from any other way, for example, Eclipse project into Android Studio. Android Studio will convert the Eclipse project to an Android Studio project, adding the necessary configuration files for us.

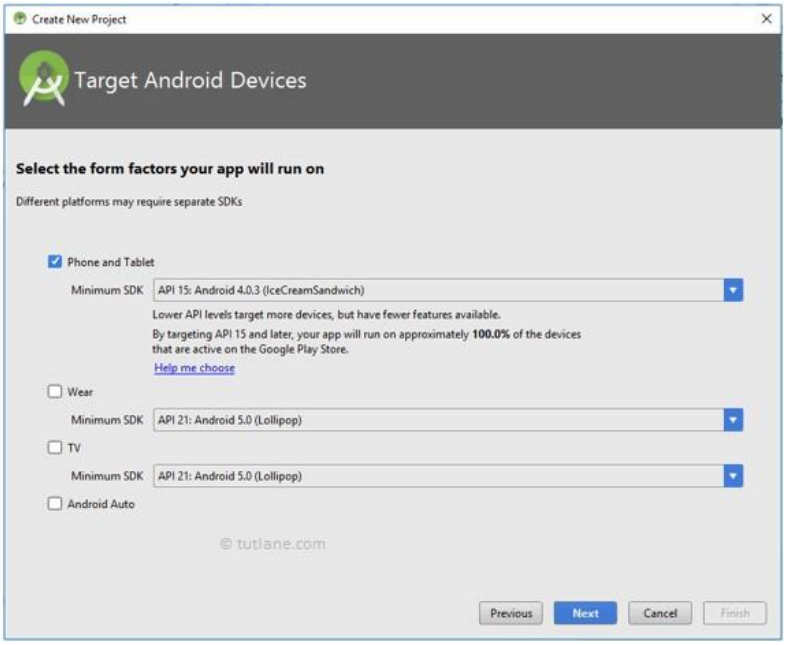
If we select **Open Project** from the list of options, we can open projects created with either Android Studio or IntelliJ IDEA.

Check out from Version Control, we can check out a copy of a project that's under version control. This is a great way to quickly get up to speed with an existing project.

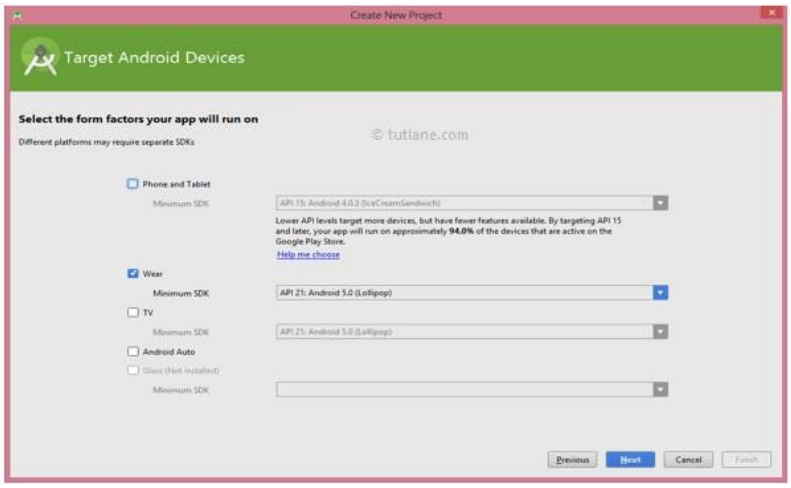
To get us started, choose **New Project** from the list of options. This will show us a list of options to configure our new project. As we click on "**New Project**" from the above option, then the next screen will be open like this, where we have to mention our **Project's name**, **Company domain** and **Project location** (we called it the main path where this application will be saved) because the **Package name** will be created automatically as we create the project in Android Studio.



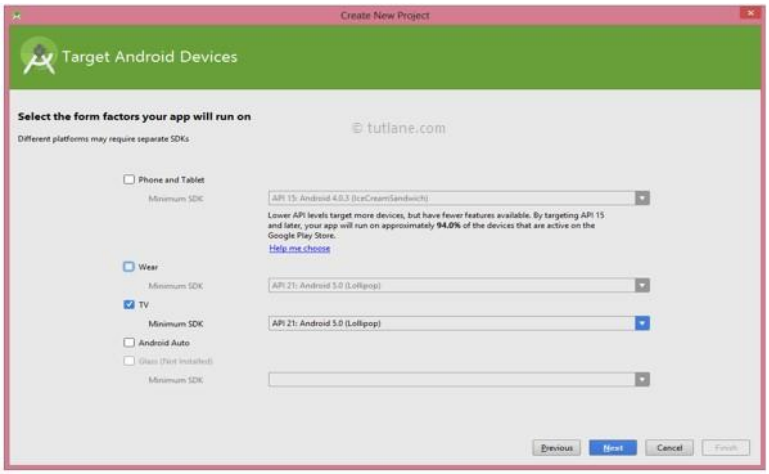
After entering all the details if we click on the "**Next**" button another screen will appear where we have select the different platforms and SDK targets like as shown below based on our requirements.



Here we need to select the type of Platform which we are going to use for the Application development like if we select "**Phone and Tablet**", then it will show it's different **API** and **SDK** version and similar to others. If we choose "**Wear**", then it will show it's API and SDK versions like as shown below.



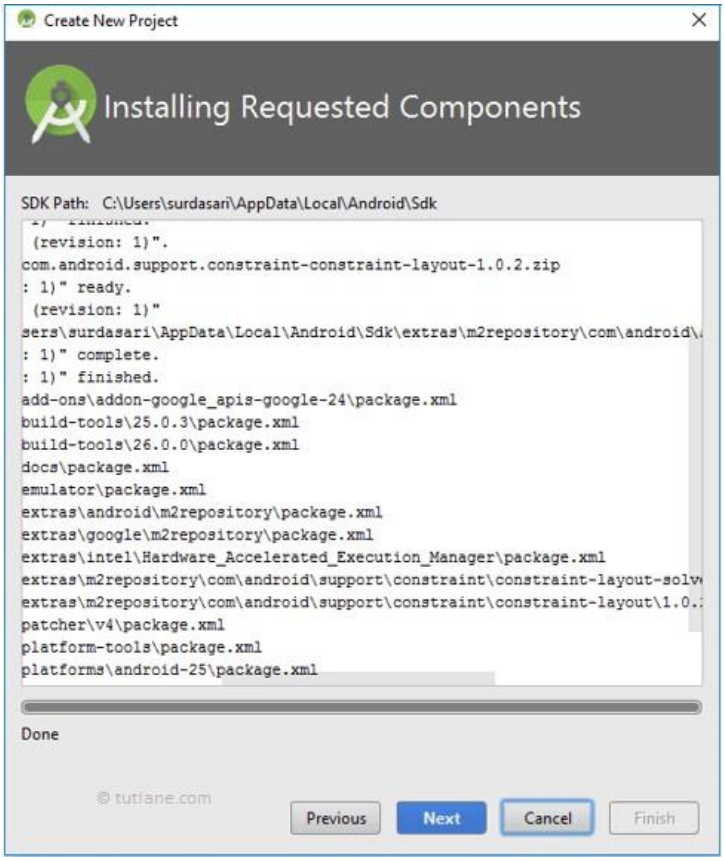
In case if we choose "**TV**", then it will show it's API and SDK versions like as shown below.



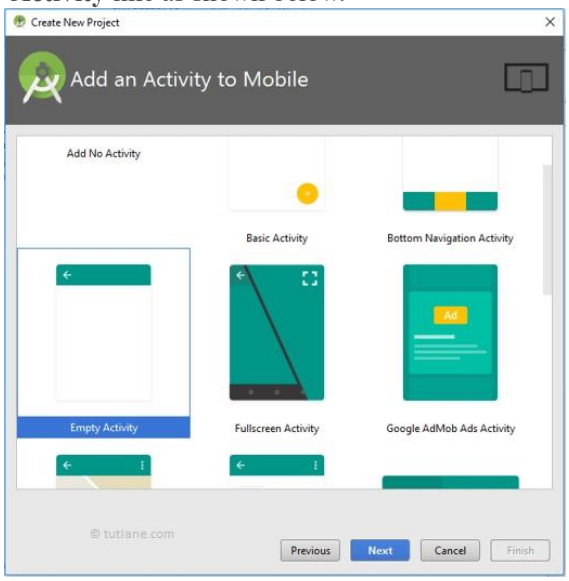
Wear: We use this option for **Android Watches** which we can wear to our hand and use the same functionality as we do with the Android devices. You can call, set the alarm, capture images, and many more things easily.

TV: We use this option for **SmartIPTV** which is very common these days. We can see our favorite channels like we see in our Home Televisions and make the changes in the channel easily.

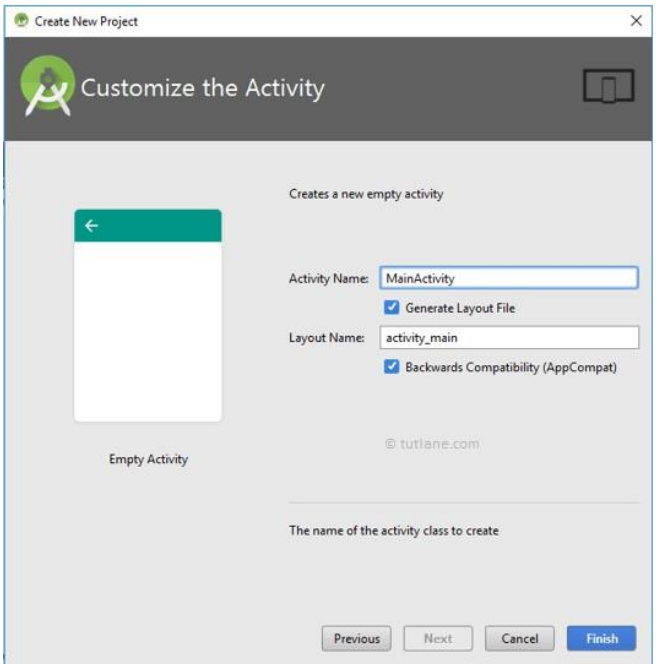
Here we are going to implement an app for phones and tablets, so we selected a **Phone and Tablet** option and click "**Next**" and it will install required components like as shown below.



Now click **Next** to select the particular Activity for our requirement. If we will select the "**Empty Activity**", then it will show the empty activity in our layout. In case if we choose other options, then it will show the activity which we have chosen. Here we are selecting **Empty Activity** like as shown below.

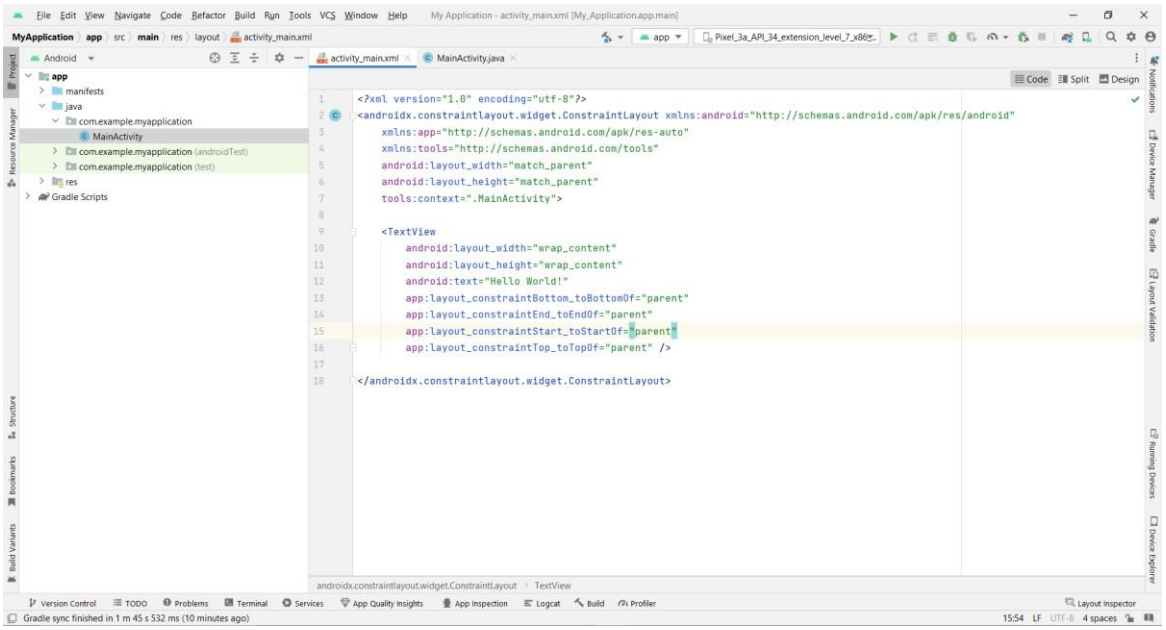


After choosing the "**Activity**" for our application, then click on the "**Next**" button and it will take you to the next screen like as shown below.



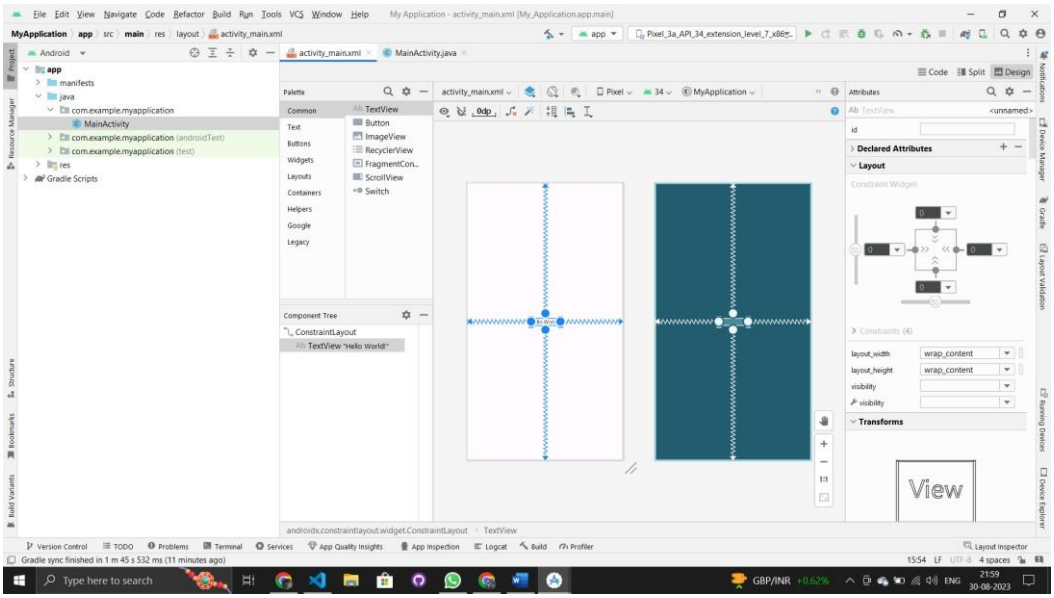
Here we can see that the Activity i.e. **EmptyActivity** which we selected in the previous section and the java file name i.e. "**MainActivity**". Now we are ready for the final step, just click on the "**Finish**" button and it will take you to the Main page where we have to do the coding and create new layouts over there.

After clicking **Finish**, we will be presented with Android Studio's user interface with the project explorer on the left and the workspace on the right like as shown below.



To know more about the folders in android application, check this Android App Folder Structure. The following are the important files that we need to build our app in android studio.

Android Layout File (activity_main.xml):
The UI of our application will be designed in this file and it will contain **Design** and **Text** modes. It will exist in the **layouts** folder and the structure of **activity_main.xml** file in **Design** mode like as shown below.



We can make required design modifications in **activity_main.xml** file either using **Design** or **Text** modes. If we switch to **Text** mode **activity_main.xml** file will contain code like as shown below.

Android Main Activity File (MainActivity.java):
The main activity file in android application is **MainActivity.java** and it will exists in **java** folder.
The **MainActivity.java** file will contain the java code to handle all the activities related to our app.

Following is the default code of **MainActivity.java** file which is generated by our **HelloWorld** application.

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity; import
android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
    }
}
```

Android Manifest File (AndroidManifest.xml)
Generally, our application will contain multiple **activities** and we need to define all those **activities** in the **AndroidManifest.xml** file. In our manifest file, we need to mention the main activity for our app using the **MAIN** action and **LAUNCHER** category attributes in **intent filters** (<intent-filter>). In case if we didn't mention the MAIN action or LAUNCHER category for the main activity, our app icon will not appear in the home screen's list of apps.

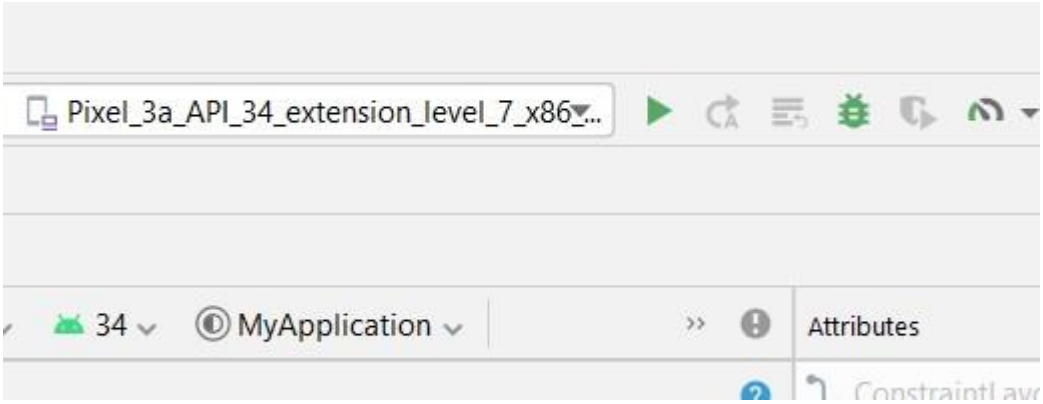
Following is the default code of the **AndroidManifest.xml** file which is generated by our **HelloWorld** application.

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

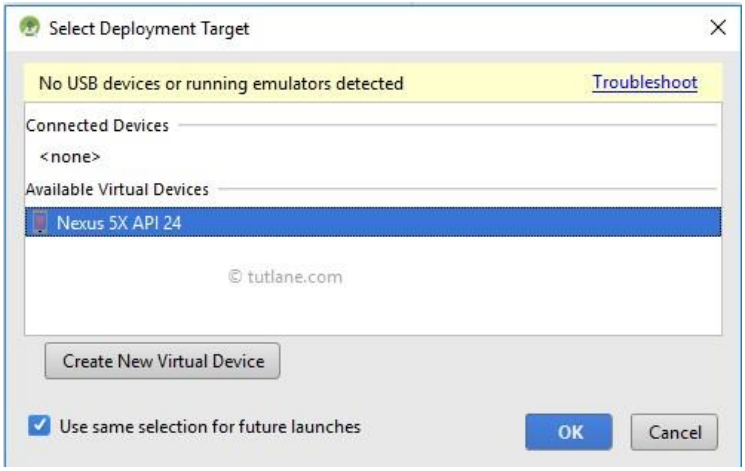
```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"     xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
   android:layout_height="match_parent"
tools:context=".MainActivity">

   <TextView
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_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

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

Run Android Hello World App

To run android applications, we need to click on **Run** button or press **Shift + F10** like as shown below

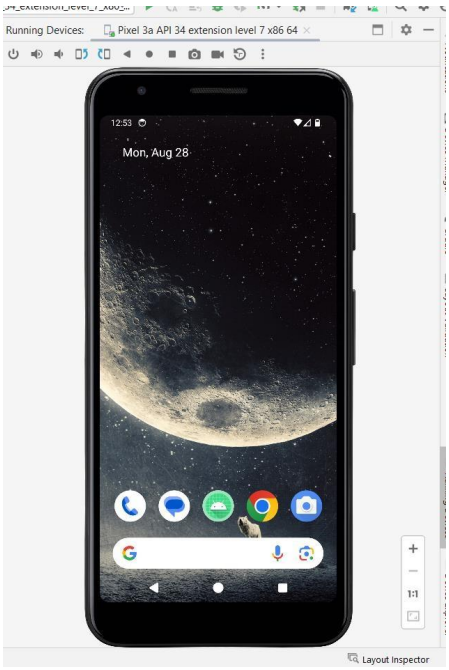


After clicking on the play button new window will open in that select **Android Virtual Device** (AVD) and click **OK** like as shown below.

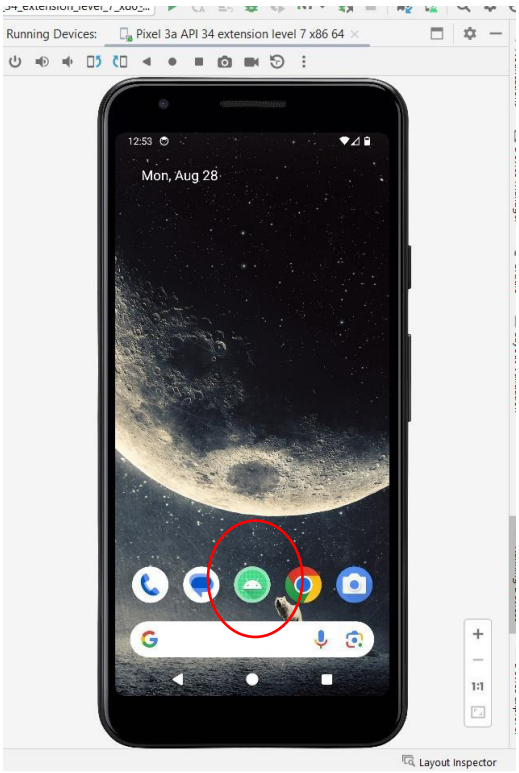


In case if you are not able to see any virtual device, then you need to create a virtual device to run your application for that check this Android Virtual Device Setup.

Now our android hello world application will show the result like as shown below



In our **AndroidManifest.xml** file, we mentioned the MAIN action and LAUNCHER category attributes for our main activity file due to that our app icon will create in Home screen list of apps like as shown below.



This is how we can create apps in android and execute applications based on our requirements.

6. Aim: Develop android application that will get the Text Entered in Edit Text and display that Text using toast message on clicking a button
Resources required: A desktop with Android studio IDE installed with atleast 8gb of ram and 2Ghz Proxessor
Procedure:

-Create a new Android application and setup your AVD in your Android Studio IDE **Activity_main.xml :**

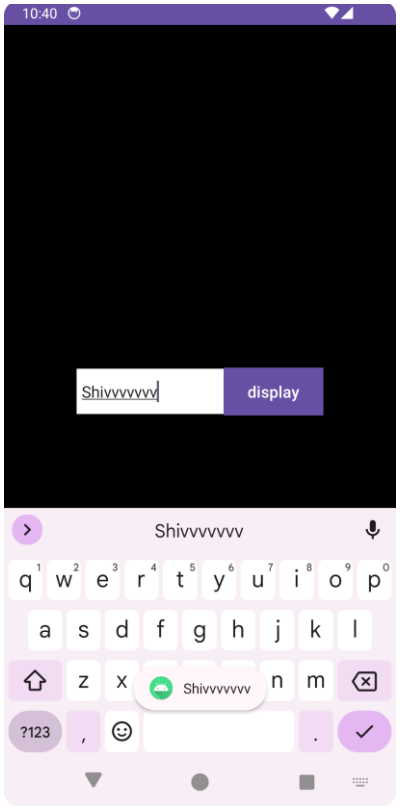
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent" android:orientation="horizontal" android:gravity="center"
android:background="@color/black" tools:context=".MainActivity">

  <EditText
     android:id="@+id/et"
     android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:paddingTop="12sp"
android:paddingBottom="12sp"
android:paddingLeft="5sp" android:paddingRight="5sp"
     android:hint="@string/enter_your_text_here"
android:textSize="16sp" android:background="#ffffff"
     android:inputType="text"
     />
  <Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/display"
android:textSize="16sp"
     android:background="@color/black"
android:id="@+id/mybtn"
    />
</androidx.appcompat.widget.LinearLayoutCompat>
```

MainActivity.java

```
package com.example.myapplication;

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

```
import android.os.Bundle;
import android.widget.*; import
android.view.View;
public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        setContentView(R.layout.activity_main);
       Button btn;
       EditText editText;          btn =
findViewById(R.id.mybtn);
editText = findViewById(R.id.et);
      btn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
           String text = editText.getText().toString();
           if(!text.isEmpty())
               Toast.makeText(MainActivity.this,text,Toast.LENGTH_LONG).show();
else
              Toast.makeText(MainActivity.this,"please enter text",Toast.LENGTH_SHORT).show();

         }
      });
  } }
```

Output:



7.Aim: Create an Android app to accept two numbers in EditText and display the sum od them in a Toast message on clicking a button
Resources required: An Android studio installed Desktop with atleast 16gb of ram and 2ghz processor **Procedure:**
activity_main.xml:

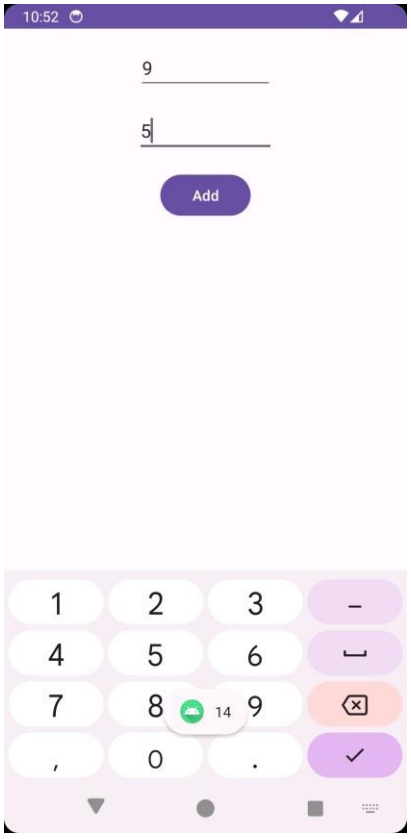
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerHorizontal="true"     tools:context=".MainActivity">

   <EditText
android:id="@+id/firstEditText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
android:hint="Enter number 1"
      android:inputType="number" />

   <EditText
android:id="@+id/secondEditText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/firstEditText"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
android:hint="Enter Number 2"
      android:inputType="number" />

   <Button
      android:id="@+id/addButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/secondEditText"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
      android:text="Add" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.myapplicationforaddingtwonnumbers;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
   private EditText firstEditText;
private EditText secondEditText;
private Button addButton;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        setContentView(R.layout.activity_main);

       firstEditText = findViewById(R.id.firstEditText);
secondEditText = findViewById(R.id.secondEditText);        addButton
= findViewById(R.id.addButton);
       addButton.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
try {
                int num1 = Integer.parseInt(firstEditText.getText().toString());
int num2 = Integer.parseInt(secondEditText.getText().toString());
int sum = num1 + num2;
                Toast.makeText(MainActivity.this,String.valueOf(sum),Toast.LENGTH_LONG).show();

             } catch (NumberFormatException e) {
                Toast.makeText(MainActivity.this,"error",Toast.LENGTH_LONG).show();
             }
          }
       });
   } }
```

Output:



8. Aim: Create an Android ap to accept a number EditText and Display the factorial of it in a Toast message on clicking a button

Resources required: An Android studio installed Desktop with atleast 16gb of ram and 2ghz processor **Procedure:**

activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerHorizontal="true"     tools:context=".MainActivity">

   <EditText
      android:id="@+id/firstEditText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"        android:hint="Enter
number 1"        android:inputType="number" />

   <EditText
      android:id="@+id/secondEditText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/firstEditText"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"        android:hint="Enter
Number 2"        android:inputType="number" />

   <Button
      android:id="@+id/addButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/secondEditText"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
```

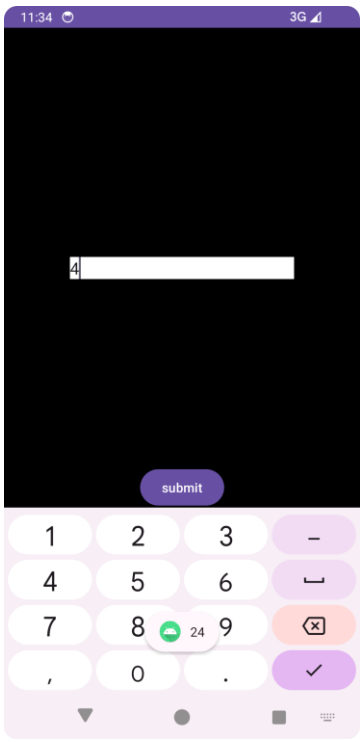
android:text="Add" /> </RelativeLayout>

MainActivity.java

```
package com.example.myapplicationforaddingtwonnumbers;


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


import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {


   private EditText firstEditText;
private EditText secondEditText;
private Button addButton;




   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        setContentView(R.layout.activity_main);
firstEditText = findViewById(R.id.firstEditText);        secondEditText =
findViewById(R.id.secondEditText);        addButton =
findViewById(R.id.addButton);
       addButton.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
try {
```

```
            int num1 = Integer.parseInt(firstEditText.getText().toString());
int num2 = Integer.parseInt(secondEditText.getText().toString());
int sum = num1 + num2;
            Toast.makeText(MainActivity.this,String.valueOf(sum),Toast.LENGTH_LONG).show();


          } catch (NumberFormatException e) {
            Toast.makeText(MainActivity.this,"error",Toast.LENGTH_LONG).show();
          }
        }
      });
  }
}
```

Output:



9.Aim: Design a simple calculator application to perform addition, subtraction, multiplication and division using different buttons.
Resources required: A desktop with Android studio IDE with a minimum of 8gb ram and 2ghz processor
Procedure:
Activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   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:layout_width="match_parent"
```

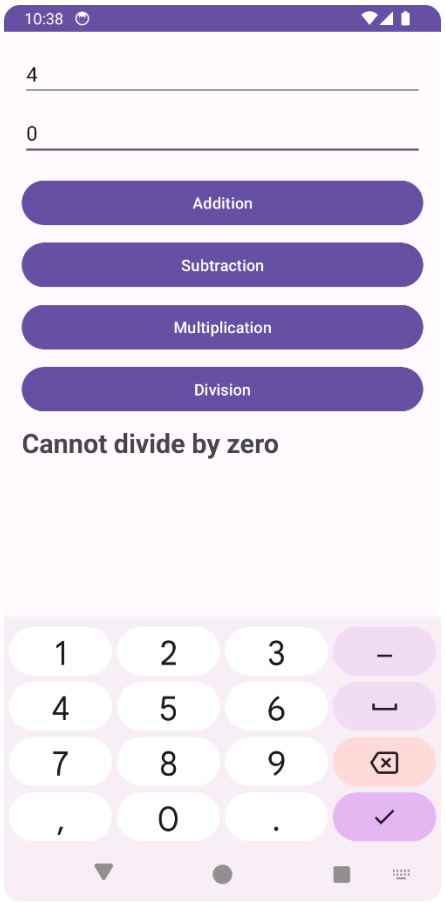
```
android:layout_height="match_parent"
android:orientation="vertical"     android:padding="16dp"
   tools:context=".MainActivity">

   <EditText
      android:id="@+id/number1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Enter number 1"
android:inputType="number" />

   <EditText
      android:id="@+id/number2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp"         android:hint="Enter
number 2"
      android:inputType="number" />

   <Button
      android:id="@+id/addButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"         android:text="Addition"
/>

   <Button
android:id="@+id/subtractButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp"
android:text="Subtraction" />

   <Button
      android:id="@+id/multiplyButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp"         android:text="Multiplication"
/>

   <Button
      android:id="@+id/divideButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp"         android:text="Division" />

   <TextView
android:id="@+id/resultTextView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="8dp"
android:textSize="24sp"
android:textStyle="bold" />
```

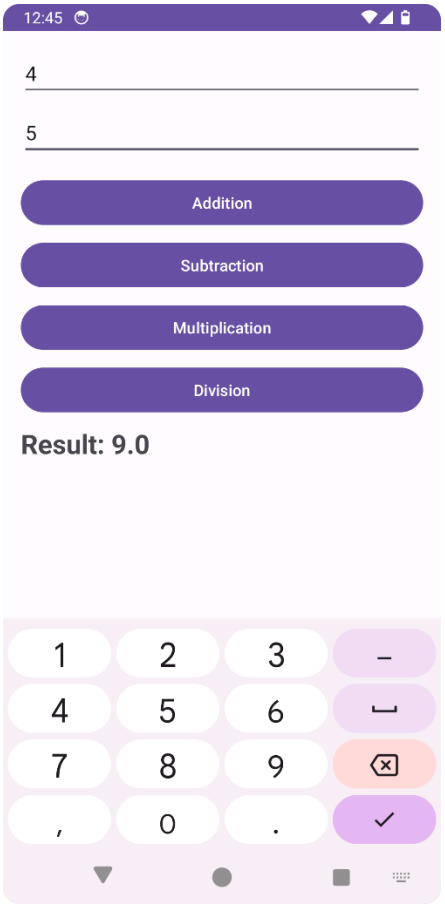
```
</LinearLayout>
```

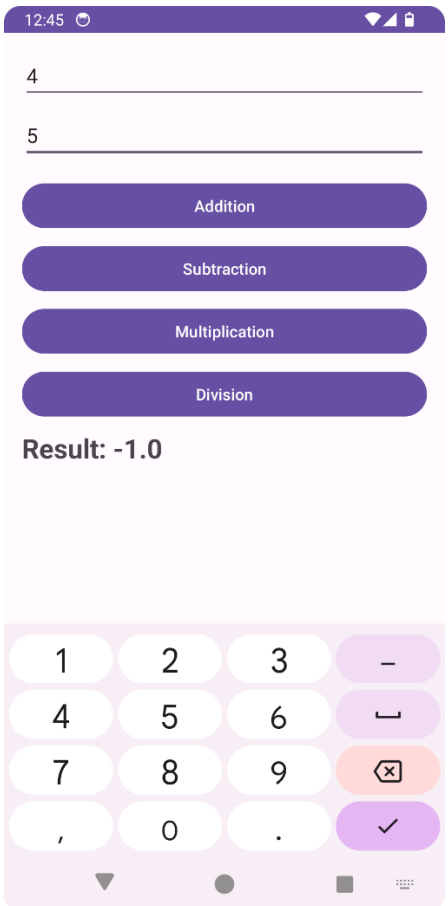
MainActivity.java:

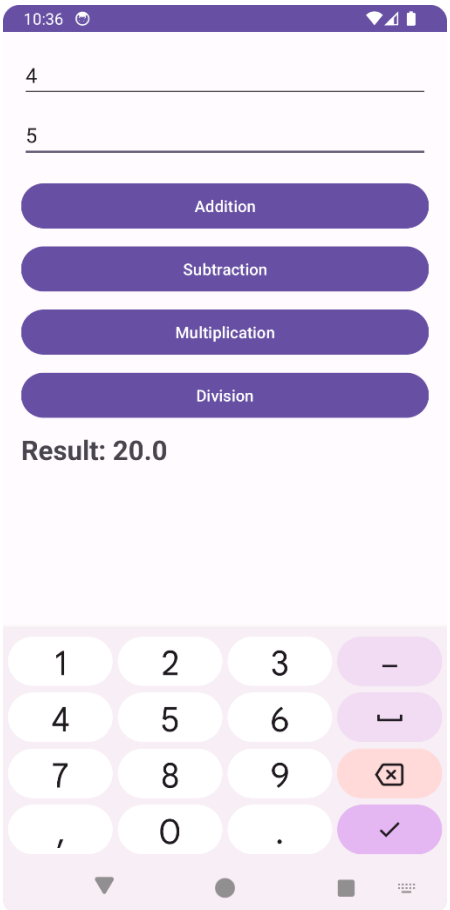
```
package com.example.myapplication1;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   private EditText number1EditText;
private EditText number2EditText;
private Button addButton;    private
Button subtractButton;    private
Button multiplyButton;    private
Button divideButton;
   private TextView resultTextView;

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

      number1EditText = findViewById(R.id.number1);
number2EditText = findViewById(R.id.number2);
addButton = findViewById(R.id.addButton);       subtractButton
= findViewById(R.id.subtractButton);       multiplyButton =
findViewById(R.id.multiplyButton);       divideButton =
findViewById(R.id.divideButton);       resultTextView =
findViewById(R.id.resultTextView);

      addButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
calculateAndDisplayResult('+');
         }
      });

      subtractButton.setOnClickListener(new View.OnClickListener() {
@Override
         public void onClick(View v) {
calculateAndDisplayResult('-');
         }
      });

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

```
            calculateAndDisplayResult('*');
        }
    });

    divideButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            calculateAndDisplayResult('/');
        }
    });
}

  private void calculateAndDisplayResult(char operator) {
try {
        double num1 = Double.parseDouble(number1EditText.getText().toString());
double num2 = Double.parseDouble(number2EditText.getText().toString());          double
result = 0;

        switch (operator) {
case '+':
            result = num1 + num2;
            break;
case '-':
            result = num1 - num2;
            break;
case '*':
            result = num1 * num2;
            break;
case '/':
            if (num2 != 0) {
result = num1 / num2;
            } else {
                resultTextView.setText("Cannot divide by zero");
return;
            }
break;
        }

        resultTextView.setText("Result: " + result);
} catch (NumberFormatException e) {
resultTextView.setText("Invalid input");
    }
  }
}
```









Student name: P.Supriya

Pin: 21001-CS-106

Faculty: M. Suresh MTech. PhD

10.Aim: Develop adroid Application using Linear Layout
Resources required: A desktop with Android studio IDE with a minimum of 8gb ram and 2ghz processor
Procedure:
activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="match_parent"
android:layout_height="match_parent"     android:orientation="vertical"
android:padding="16dp"
   tools:context=".MainActivity">

   <TextView
android:id="@+id/counterTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:text="Clicked: 0 times"
      android:textSize="18sp" />

   <Button
android:id="@+id/incrementButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:layout_marginTop="16dp"
      android:text="click" />

</LinearLayout>
```

MainActivity.java

```
package com.example.myapplicationlinearlayout;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   private TextView counterTextView;
private Button incrementButton;
   private int counter = 0;

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

```
        counterTextView = findViewById(R.id.counterTextView);
incrementButton = findViewById(R.id.incrementButton);        updateCounter();

        incrementButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                counter++;
                updateCounter();
            }
        });
    }

    private void updateCounter() {
        counterTextView.setText("Clicked " + counter+"times");
    }
}
```

Output:

11.Aim: Develop android application using Relative Layout
Resources required: A desktop with Android studio IDE with a minimum of 8gb ram and 2ghz processor **Procedure:**
Activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:layout_width="match_parent"
android:layout_height="match_parent"     android:padding="16dp"
tools:context=".MainActivity">

   <TextView
      android:id="@+id/counterTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
      android:text="Counter: 0"
android:textSize="18sp" />

   <Button
android:id="@+id/incrementButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/counterTextView"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
      android:text="Increment" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.realativelayout;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   private TextView counterTextView;
private Button incrementButton;
   private int counter = 0;

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

      counterTextView = findViewById(R.id.counterTextView);
incrementButton = findViewById(R.id.incrementButton);

      updateCounter();
```

```
        incrementButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                counter++;
updateCounter();
            }
        });
    }

    private void updateCounter() {
        counterTextView.setText("Counter: " + counter);
    }
}
```

Output:

12.Aim: Develop android application using Table Layout
Resources required: A desktop with Android studio IDE with a minimum of 8gb ram and 2ghz processor **Procedure:**
Activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/tableLayout"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"    android:padding="16dp"
   >

   <TableRow>       <TextView
android:text="Name"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>

      <TextView          android:text="Age"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>
   </TableRow>

   <TableRow>       <TextView
android:text="Shiv"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>

      <TextView          android:text="17"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>
   </TableRow>

   <TableRow>
<TextView
        android:text="Chiruu"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>

      <TextView          android:text="18"
android:padding="5dp"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="wrap_content"
android:textSize="19sp"/>
   </TableRow>

</TableLayout>
```

MainActivity.java

```
package com.example.tablelayout;

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


public class MainActivity extends AppCompatActivity {

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

Output:

13.Aim: Develop android application using Constraint Layout
Resources required: A desktop with Android studio IDE with a minimum of 8gb ram and 2ghz processor **Procedure:**
activity_main.xml :

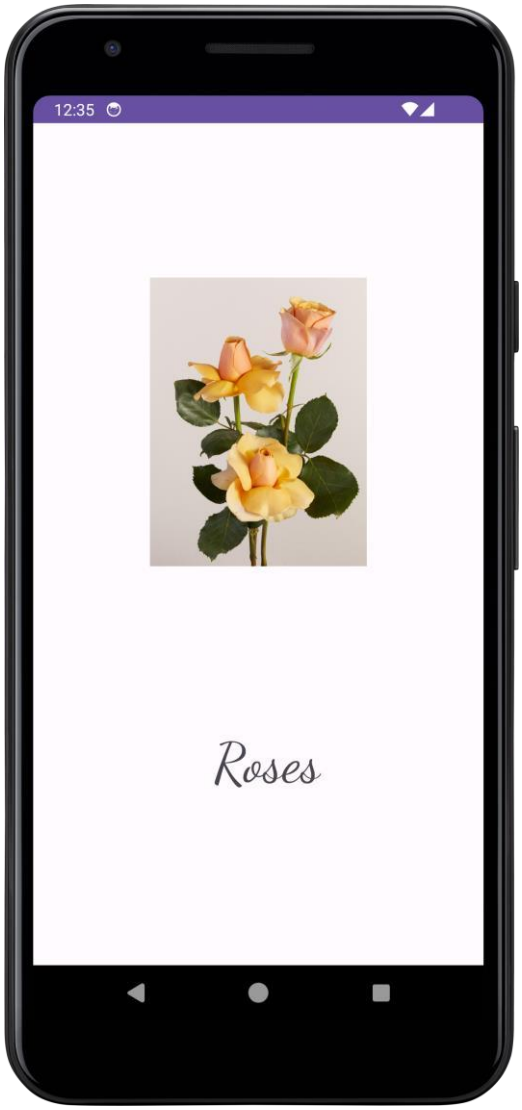
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"     xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"     android:layout_height="match_parent"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

    <TextView
```

```
      android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:fontFamily="cursive"        android:text="Roses"
android:textSize="48sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.526"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.78" />

   <ImageView
android:id="@+id/imageView"
android:layout_width="189dp"
android:layout_height="409dp"
      app:layout_constraintBottom_toTopOf="@+id/textView"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"        app:srcCompat="@drawable/flower"
/>

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

MainActivity.java

```
package com.example.constraintlayout;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Output:



Student name: P.Supriya

Pin: 21001-CS-106

Faculty: M. Suresh MTech. PhD