Mobile Application Development android (CS-693)

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

(Week 1) Lecture 1-2

Objectives: **Learning objectives of this lecture are**

- **What is android?**
- **History of android**
- **Features of android**
- **Android Development tool**
- **Android project structure**
- **Creating android first application.**
- **Android layout**
- **Layout Types**
- **View Identification**
- **Relative Layout**
- **Linear Layout**

Mobile Application Development android (CS-693)

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

What is Android?

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

Google released the first beta of Android 10 under the preliminary name "**Android Q**" on March 13, 2019, exclusively on their **Pixel phones**. On August 22, 2019, it was announced that Android "Q" would be branded solely as **"Android 10"**, with no codename. Dave Burke did reveal during a podcast that, in addition, most desserts beginning with "Q" were **"exotic".** Android 10 was officially released on September 3, 2019

Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below

Feature	Description

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

Beautiful UI	Android OS basic screen provides a beautiful and intuitive user interface.
Connectivity	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
Storage	SQLite, a lightweight relational database, is used for data storage purposes.
Media support	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
Messaging	SMS and MMS
Web browser	Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
Multi-touch	Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
Multi-tasking	User can jump from one task to another and same time various application can run simultaneously.
Resizable widgets	Widgets are resizable, so users can expand them to show more content or shrink them to save space.
Multi-Language	Supports single direction and bi-directional text.
GCM	Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
Wi-Fi Direct	A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
Android Beam	A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit.

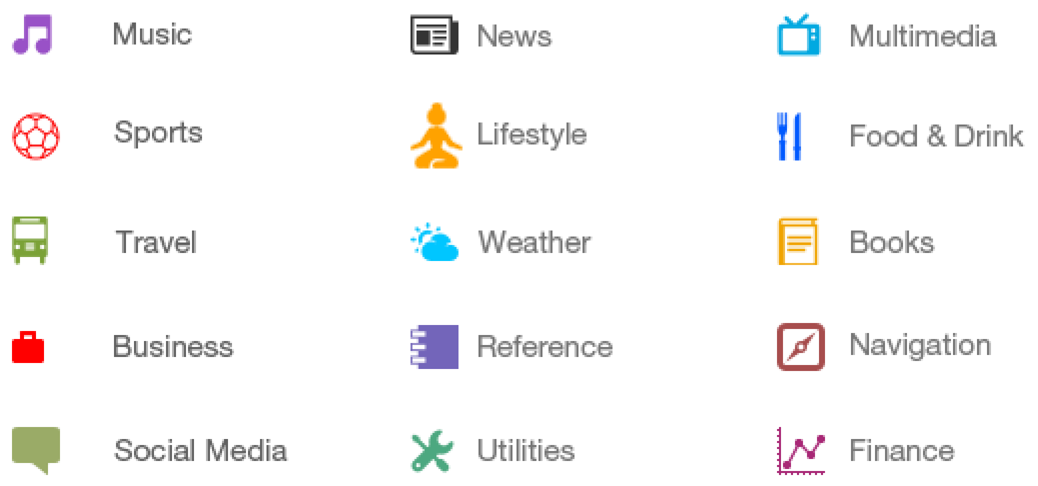
Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Categories of Android applications

There are many android applications in the market. The top categories are −

Mobile Application Development android (CS-693)

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



Android code and API level:

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Codename	Version	API level/NDK release
Android10	10	API level 29
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

Cupcake	1.5	API level 3, NDK 1
(no codename)	1.1	API level 2
(no codename)	1.0	API level 1

Android IDE

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows.

- Eclipse IDE(Depreciated)
- Android Studio

Download android studio:

Click on link below to open official android developer's website to download android studio.

https://developer.android.com/studio

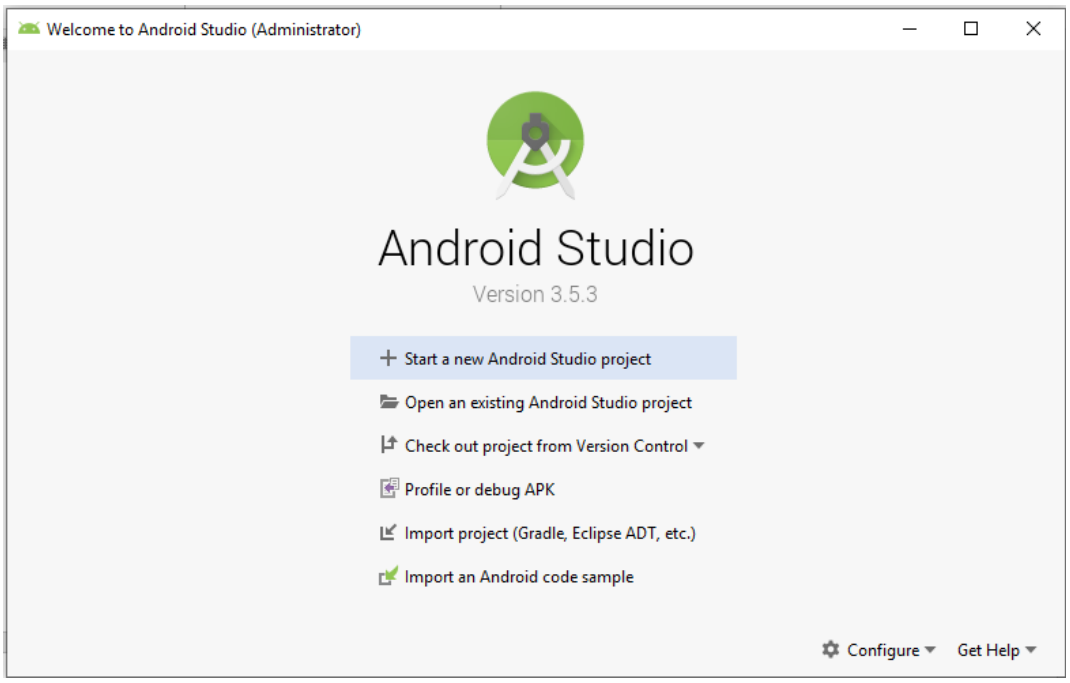
Prerequisites

Android programming is based on Java programming language so if you have basic understanding on Java programming then it will be a fun to learn Android application development.

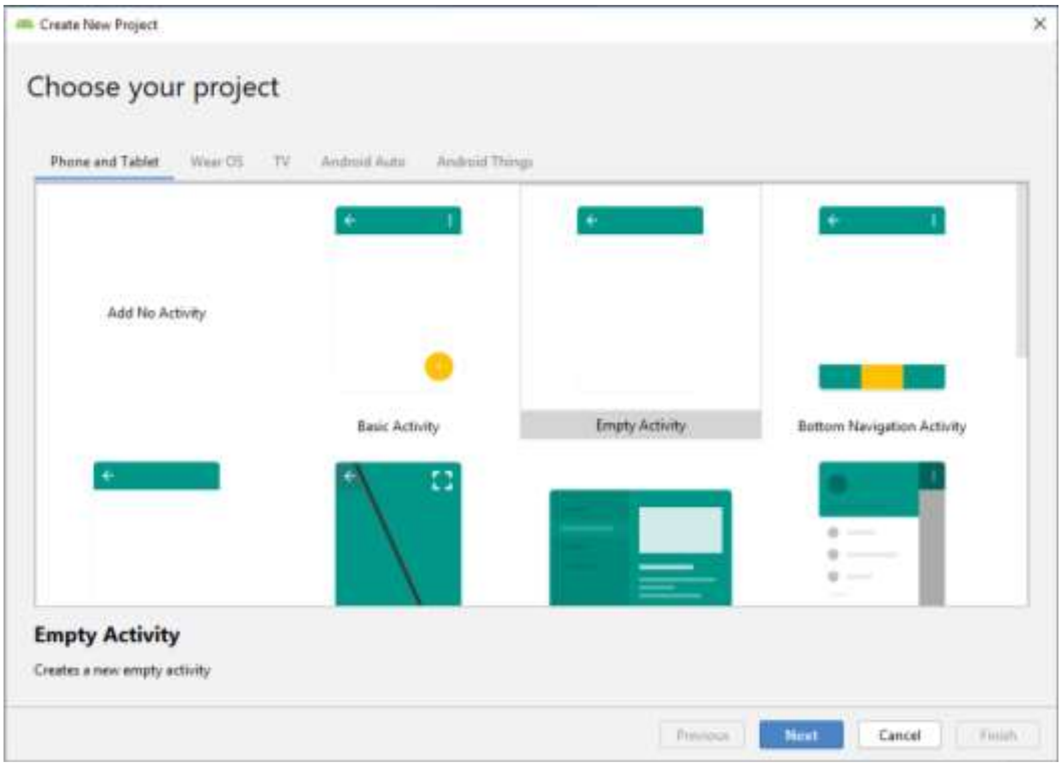
Create Android Application

The first step is to create a simple Android Application using Android studio. When you click on Android studio icon, it will show screen as shown below

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

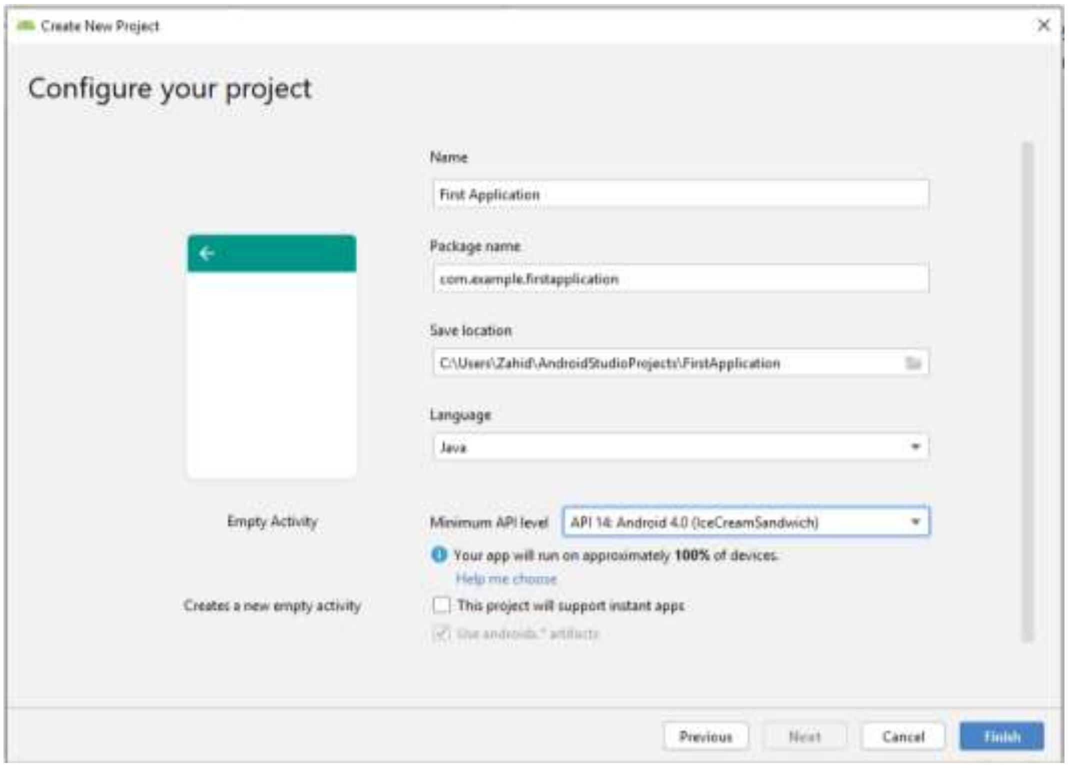

Welcome to Android Studio (Administrator)
Android Studio
Version 3.5.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

You can start your application development by calling **start a new android studio project**. After this you will Choose your project view, in which you can select type of your project either Phone and Tablet or Wear OS or TV or Android Auto or Android Things. Then select the activity you want. In this case project type is Phone and Tablet and Empty Activity is selected.



Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

Then you will see configuration view to configure your project by Typing name of application, package name will be automatically typed, can change location of project, select language either java or kotlin, and minimum SDK level.


Create New Project

Configure your project

Name: First Application

Package name: com.example.firstapplication

Save location: C:\Users\Zahid\AndroidStudioProjects\FirstApplication

Language: Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

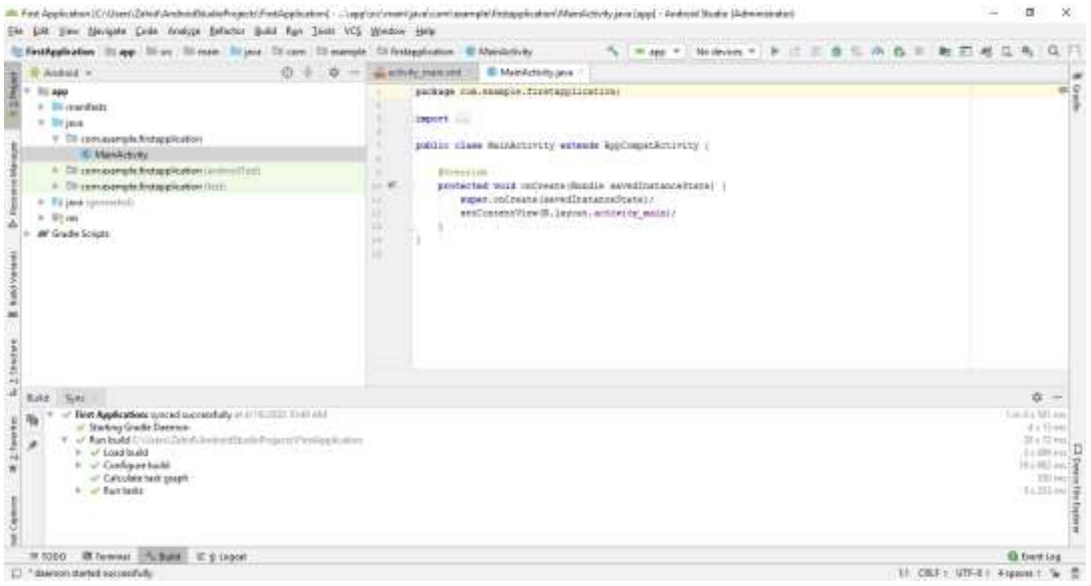
This project will support instant apps

Use androidx.* artifacts

Empty Activity

Creates a new empty activity

At the end development tool will open to write the application code.


```
package com.example.firstapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

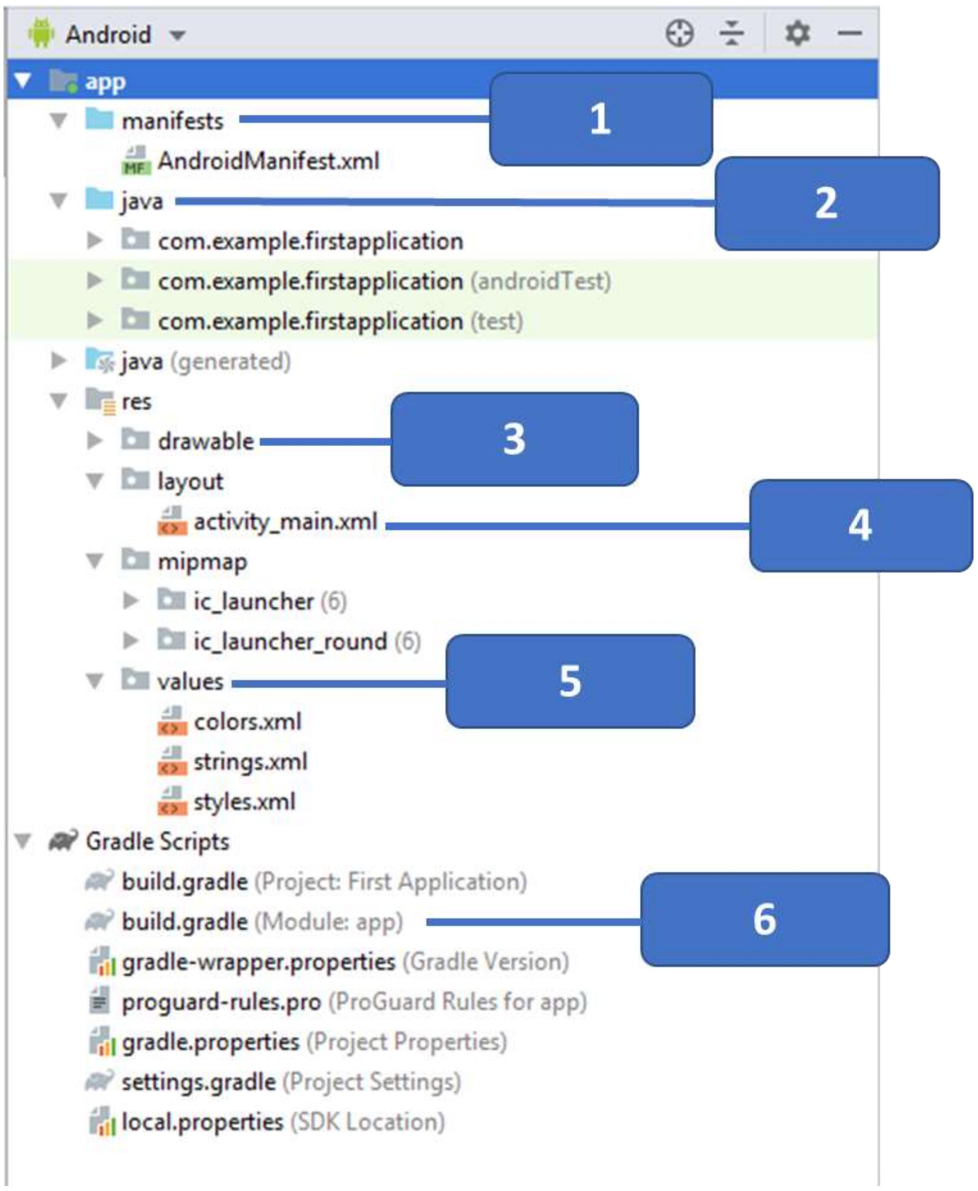
```
First Application: synced successfully
  Starting Gradle Daemon
  Run build
    Load build
    Configure build
    Calculate task graph
    Run tasks
```

Anatomy of Android Application

Before you run your app, you should be aware of a few directories and files in the Android project

Mobile Application Development android (CS-693)

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



```
▼ app
    ▼ manifests
          AndroidManifest.xml
    ▼ java
        ▶ com.example.firstapplication
        ▶ com.example.firstapplication (androidTest)
        ▶ com.example.firstapplication (test)
    ▶ java (generated)
    ▼ res
        ▶ drawable
        ▼ layout
              activity_main.xml
        ▼ mipmap
            ▶ ic_launcher (6)
            ▶ ic_launcher_round (6)
        ▼ values
              colors.xml
              strings.xml
              styles.xml
▼ Gradle Scripts
      build.gradle (Project: First Application)
      build.gradle (Module: app)
      gradle-wrapper.properties (Gradle Version)
      proguard-rules.pro (ProGuard Rules for app)
      gradle.properties (Project Properties)
      settings.gradle (Project Settings)
      local.properties (SDK Location)
```

Following section will give a brief overview few of the important application files.

Sr#	Folder, File	Description
1	AndroidManifest.xml	This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
2	Java	This contains the .java source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.
3	res/drawable	This is a directory for drawable objects that are designed screens.

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

4	res/layout	This is a directory for files that define your app's user interface.
5	res/values	This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.
6	Build.gradle	This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for Hello World! Application

```
package com.example.firstapplication;

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);
    }
}
```

Here, `R.layout.activity_main` refers to the **activity_main.xml** file located in the **res/layout** folder. The **onCreate()** method is one of many methods that are figured when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a **manifest.xml** which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Here `<application>...</application>`tags enclosed the components related to the application. Attribute `android:icon` will point to the application icon available under `mipmap/ic_launcher`.

The `<activity>` tag is used to specify an activity and `android:name` attribute specifies the fully qualified class name of the *Activity* subclass and the `android:label` attributes specifies a string to use as the label for the activity. You can specify multiple activities using `<activity>` tags.

The **action** for the intent filter is named `android.intent.action.MAIN` to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named `android.intent.category.LAUNCHER` to indicate that the application can be launched from the device's launcher icon.

The `@string` refers to the **strings.xml** file explained below. Hence, `@string/app_name` refers to the `app_name` string defined in the **strings.xml** file, which is "HelloWorld!". Similar way, other strings get populated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components −

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

The Strings File

The **strings.xml** file is located in the **res/values** folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and

Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file

```
<resources>
    <string name="app_name">First Application</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

The Layout File

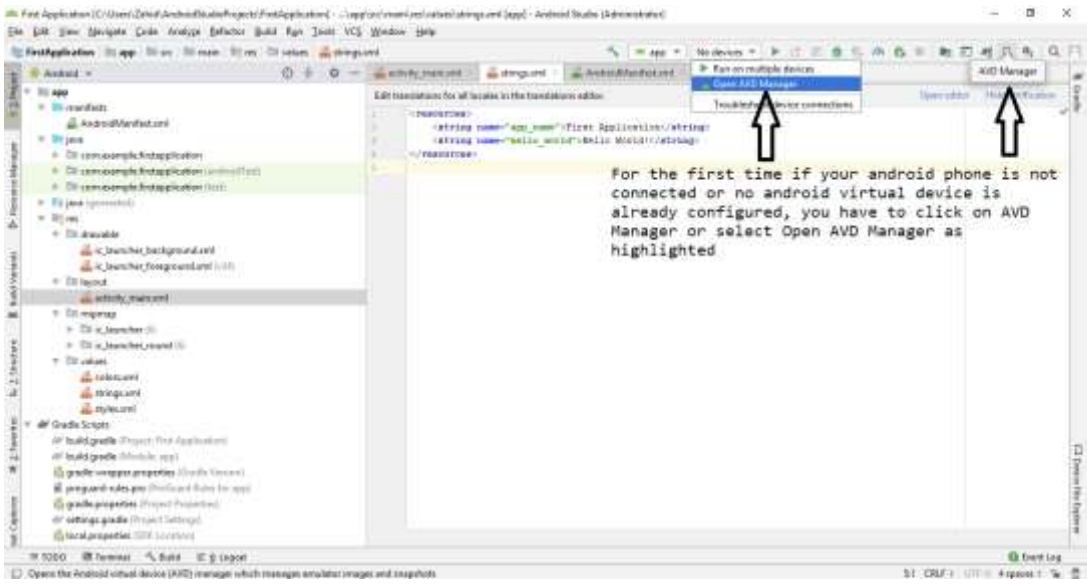
The **activity_main.xml** is a layout file available in **res/layout** directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout

```
<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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        />

</RelativeLayout>
```

This is an example of simple *RelativeLayout*. The *TextView* is an Android control used to build the GUI and it have various attributes like `android:layout_width`, `android:layout_height` etc which are being used to set its width and height etc.. The *@string* refers to the strings.xml file located in the res/values folder. Hence, `@string/hello_world` refers to the hello string defined in the strings.xml file, which is "Hello World!".

Configure AVD

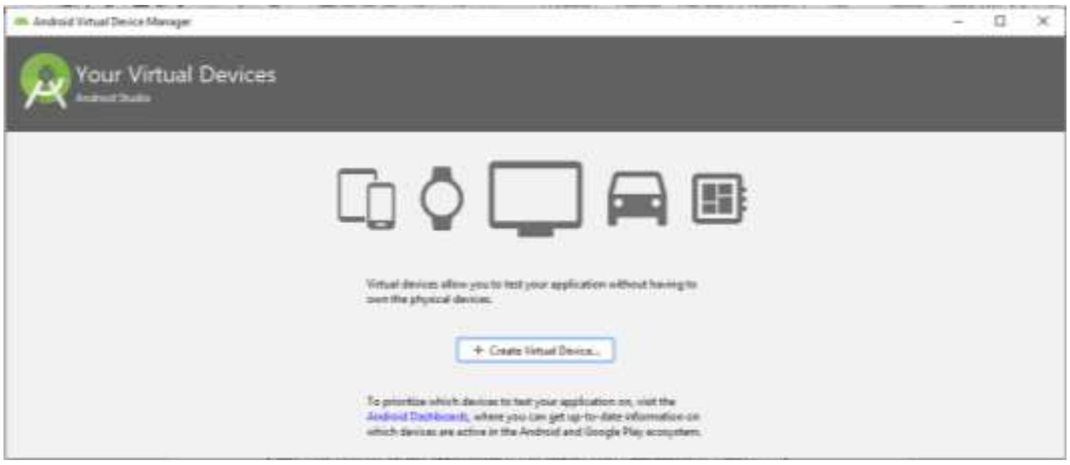
Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



```
<resources>
    <string name="app_name">First Application</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

For the first time if your android phone is not connected or no android virtual device is already configured, you have to click on AVD Manager or select Open AVD Manager as highlighted

Now click on Create Virtual Deices….

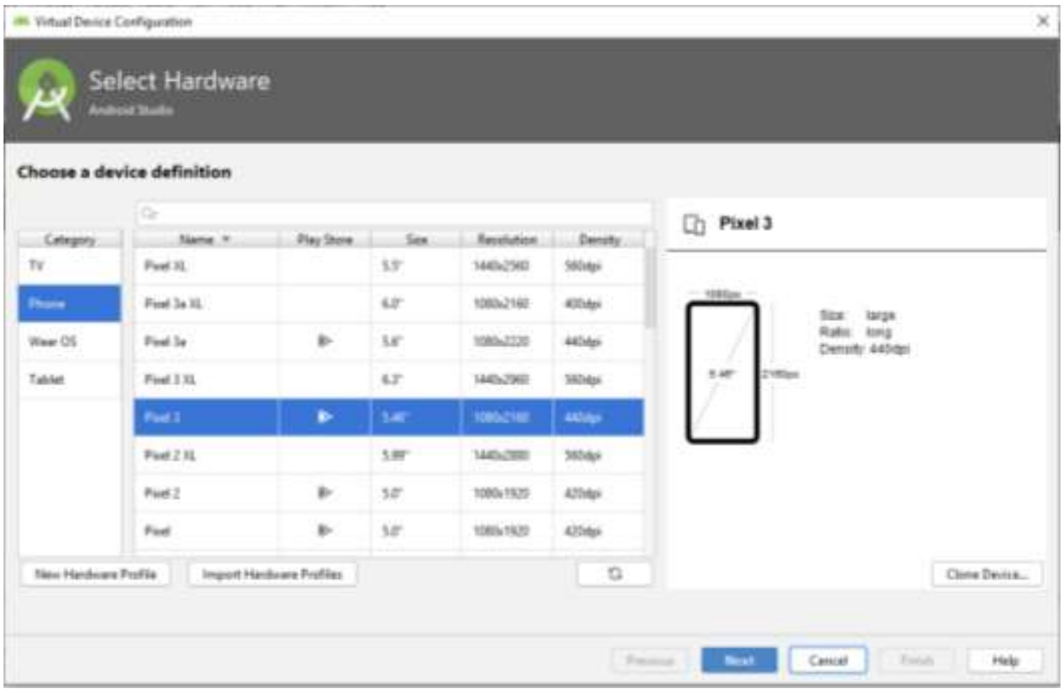


Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

Zahid Ahmed **email id:** zahid@biit.edu.pk **Whatsapp#03025367513**

Choose the device definition from available and click on next button.



Virtual Device Configuration

Select Hardware

Android Studio

Choose a device definition

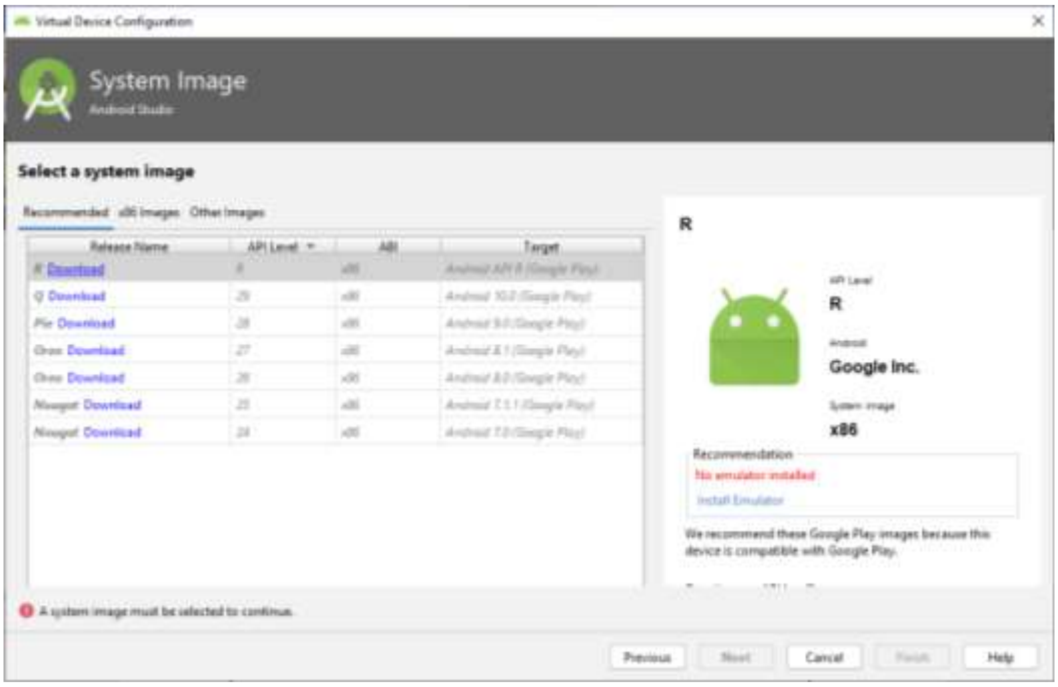
Category	Name	Play Store	Size	Resolution	Density
TV	Pixel XL		5.5"	1440x2560	560dpi
Phone	Pixel 3a XL		6.0"	1080x2160	400dpi
Wear OS	Pixel 3a	▶	5.6"	1080x2220	440dpi
Tablet	Pixel 3 XL		6.3"	1440x2960	560dpi
	Pixel 3	▶	5.46"	1080x2160	440dpi
	Pixel 2 XL		5.99"	1440x2880	560dpi
	Pixel 2	▶	5.0"	1080x1920	420dpi
	Pixel	▶	5.0"	1080x1920	420dpi

Pixel 3

Size: large
Ratio: long
Density: 440dpi

For the first time you have to downlaod device for relevant API Level. Click on download button to download a device.

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



Select a system image

Release Name	API Level ▾	ABI	Target
R Download	R	x86	Android API R (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

R

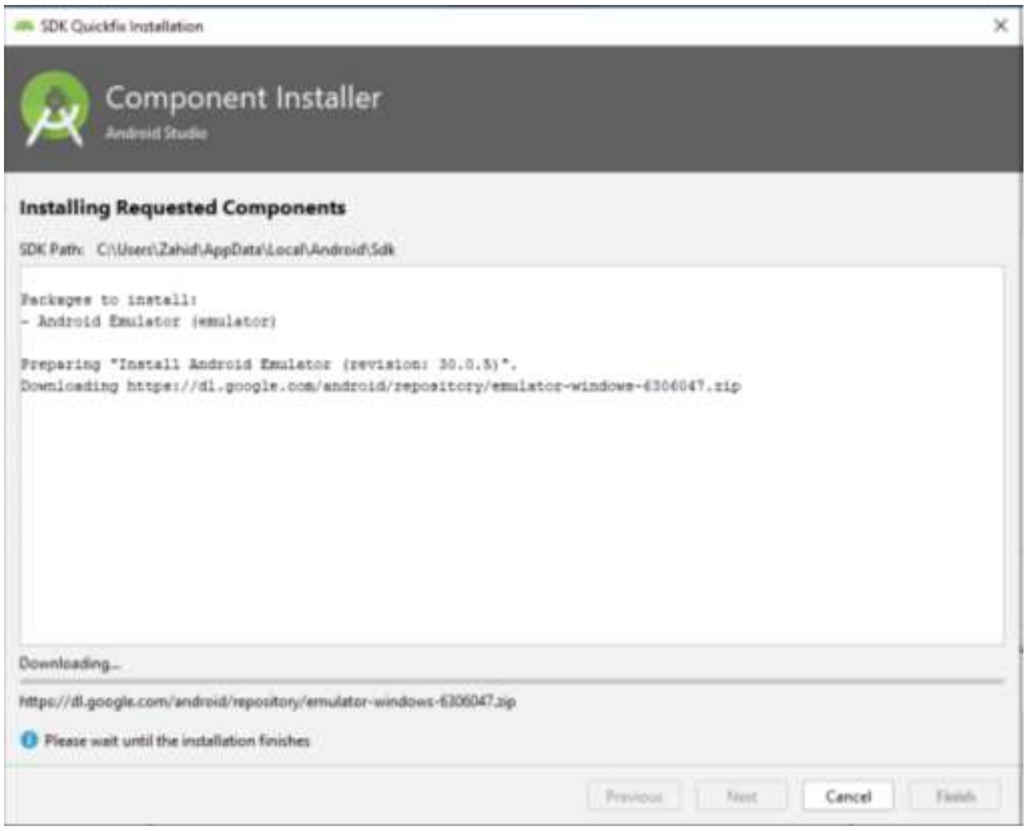
API Level
R
Android
Google Inc.
System image
x86

Recommendation
No emulator installed
Install Emulator

We recommend these Google Play images because this device is compatible with Google Play.

A system image must be selected to continue.

When download starts you will see following window.



Installing Requested Components

SDK Path: C:\Users\Zahid\AppData\Local\Android\Sdk

```
Packages to install:
- Android Emulator (emulator)

Preparing "Install Android Emulator (revision: 30.0.5)".
Downloading https://dl.google.com/android/repository/emulator-windows-6306047.zip
```

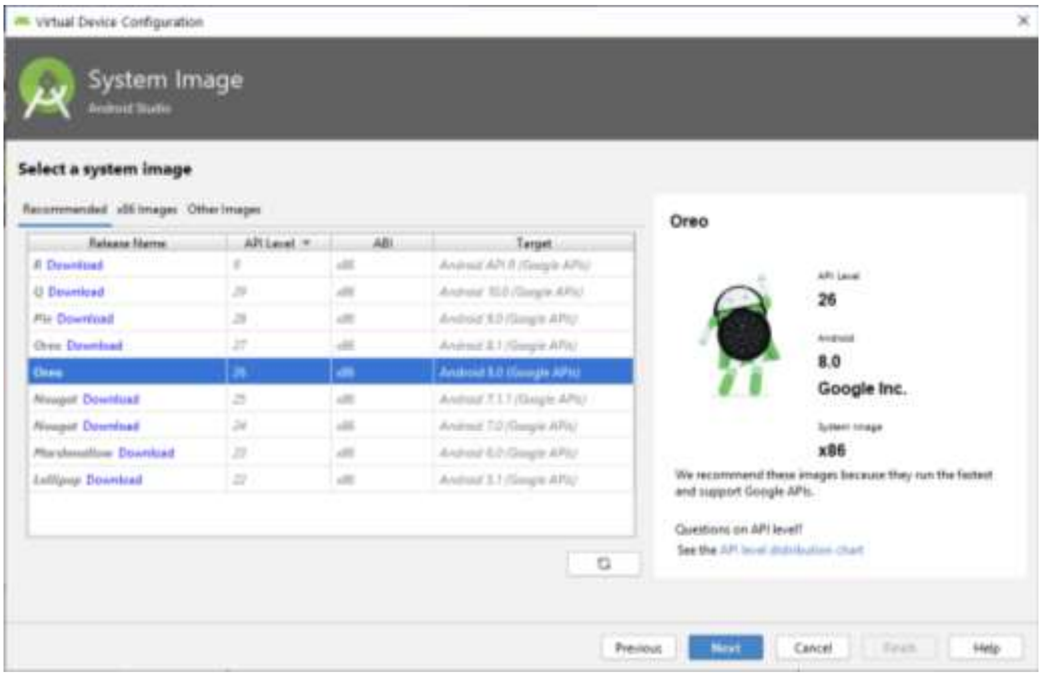
Downloading...

https://dl.google.com/android/repository/emulator-windows-6306047.zip

Please wait until the installation finishes

After completion of download you will see following window. Click on next button to create the AVD.

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



Select a system image

Release Name	API Level ▼	ABI	Target
R Download	R	x86	Android API R (Google APIs)
Q Download	29	x86	Android 10.0 (Google APIs)
Pie Download	28	x86	Android 9.0 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

Oreo

API Level: 26

Android: 8.0

Google Inc.

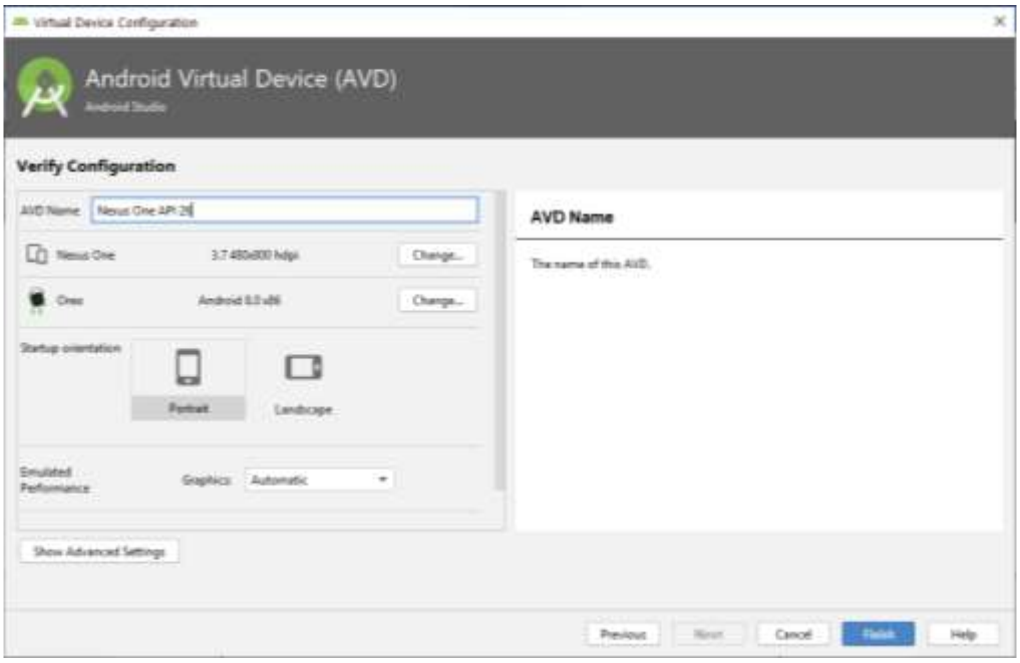
System Image: x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?

See the API level distribution chart

In Following window you have to name the AVD and its start up orientation and then click finish button.



Verify Configuration

AVD Name: Nexus One API 26

Nexus One 3.7 480x800 hdpi

Oreo Android 8.0 x86

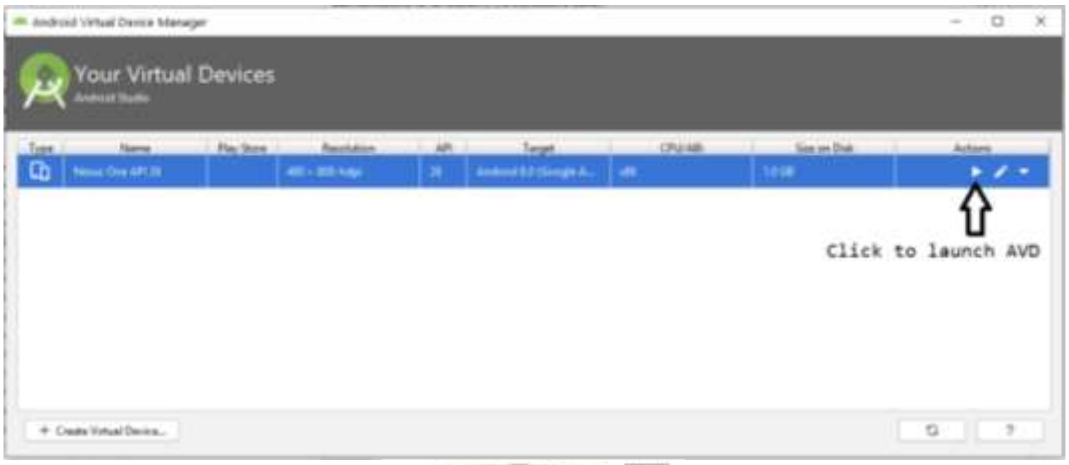
Startup orientation: Portrait

Emulated Performance Graphics: Automatic

AVD Name

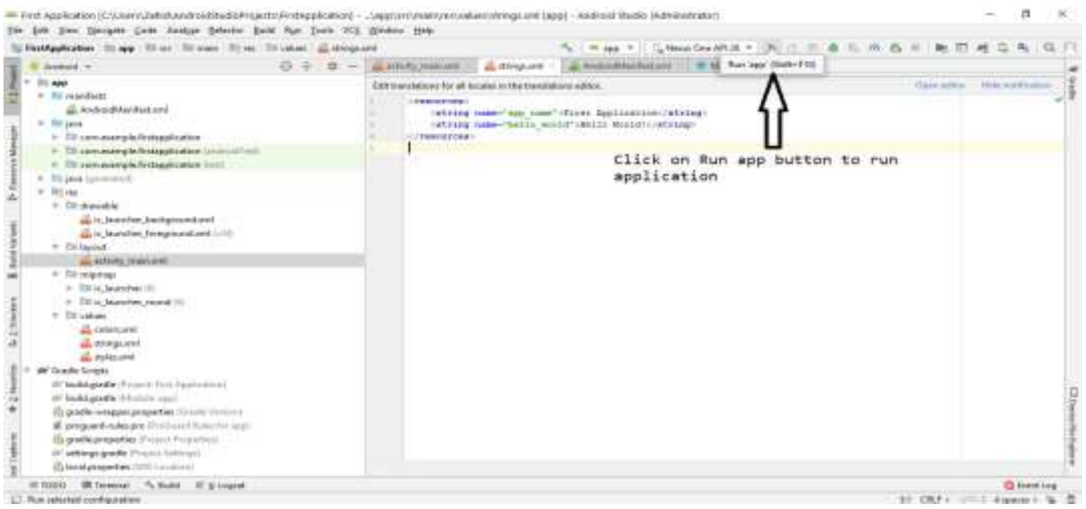
The name of this AVD.

Zahid Ahmed **email id:** zahid@biit.edu.pk **Whatsapp#03025367513**



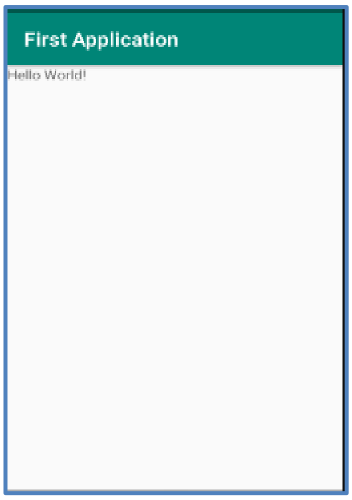
Running the Application

Let's try to run our **Hello World!** application we just created. To run the app from Android studio, open one of your project's activity files and click Run ▶ icon from the tool bar.



Android studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display following Emulator window.

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**



Android Layout

The basic building block for user interface is a **View** object that is created from the View class and occupies a rectangular area on the screen. Views are the base class for UI components like TextView, Button, EditText etc.

The **ViewGroup** is a subclass of View. One or more Views can be grouped together into a ViewGroup. A ViewGroup provides the android layout in which we can order the appearance and sequence of views. Examples of ViewGroup are Linear Layout, Relavative Layout etc.

Android Layout Types:

Android provides the following ViewGroups or layouts

- **Linear Layout:** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can define Linear Layout in XML as follows.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ LinearLayout >
```

- **Relative Layout:** RelativeLayout is a view group that displays child views in relative positions. You can define RelativeLayout in XML as follows.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ RelativeLayout >
```

- **Table Layout:** TableLayout is a view that groups views into rows and columns. You can define TableLayout in XML as follows.

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow
```

```
        android:id="@+id/row1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <!-- Add elements/columns in the first
row-->

    </TableRow>
</TableLayout>
```

- **Absolute Layout:** AbsoluteLayout enables you to specify the exact location of its children. You can define AbsoluteLayout in XML as follows.

```
< AbsoluteLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ AbsoluteLayout >
```

- **Frame Layout:** The FrameLayout is a placeholder on screen that you can use to display a single view. You can define FrameLayout in XML as follows.

```
< FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ FrameLayout >
```

- **List View:** ListView is a view group that displays a list of scrollable items. You can define ListView in XML as follows.

```
< ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

- **Grid View:** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. You can define GridView in XML as follows.

```
< GridView
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ GridView>
```

Android Layout Attributes

- **android:id**: This is the ID which uniquely identifies the view
- **android:layout_width** : This is the width of the layout
- **android:layout_height** : This is the height of the layout
- **android:layout_margin** : This is the extra space outside of the view. For example if you give android:marginLeft=20dp, then the view will be arranged after 20dp from left
- **android:layout_padding**:This is similar to **android:layout_margin** except that it specifies the extra space **inside** the view
- **android:layout_gravity** : This specifies how child Views are positioned
- **android:layout_weight** : This specifies how much of the extra space in the layout should be allocated to the view

- **android:layout_x** : This specifies the x-coordinate of the layout
- **android:layout_y** : This specifies the y-coordinate of the layout

android:layout_width=**”wrap_content”** tells the view to size itself to the dimensions required by its content.

android:layout_width=**”match_parent”** tells the view to become as big as its parent view.

View Identification

The syntax for an ID, inside an XML tag is:

- The at-symbol (@) at the beginning of the string indicates that the XML parser should parse and expand the rest of the ID string and identify it as an ID resource
- The plus-symbol (+) means that this is a new resource name that must be created and added to our resources.
- Define id of a resource

```
"@+id/btnFirstButton"
```

- Access resource using id in XML

```
"@id/btnFirstButton"
```

Android Button

Android Buttons are GUI components that the users can click upon to either goto the next screen, confirm an option/trigger any event created by you, the Android Developer. You can add a button to your layout with the Button object. For example.

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="This is button"/>
```

Android Button Attributes

- **android:drawableBottom:** This is the drawable to be drawn below the text
- **android:drawableRight:** This is the drawable to be drawn to the right of the text.
- **android:layout_width** : This is the width of the layout
- **android:layout_height** : This is the height of the layout
- **android:text:** This is the Text to display.
- **android:background:** This is a drawable to use as the background.

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

- **android:id:** This supplies an identifier name for this view.
- **android:visibility:** This controls the initial visibility of the view.
- **android:onClick:** This is the name of the method in this View's context to invoke when the view is clicked.

Android RelativeLayout

Android RelativeLayout lays out elements based on their relationships with one another, and with the parent container. This is one of the most complicated layouts and we need several properties to actually get the layout we desire.

That is, using RelativeLayout we can position a view to be toLeftOf, toRightOf, below or above its siblings.

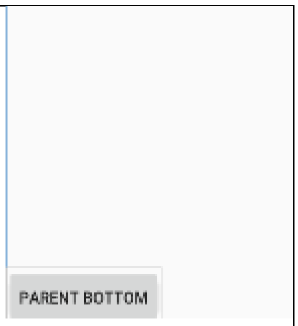
We can also position a view with respect to its parent such as centered horizontally, vertically or both, or aligned with any of the edges of the parent RelativeLayout. If none of these attributes are specified on a child view then the view is by default rendered to the top left position.

Android RelativeLayout attributes

The following are the major attributes used across **RelativeLayout**. They lay across three different categories:

- **Relative to Container**
- **android:layout_alignParentBottom :** Places the bottom of the element on the bottom of the container

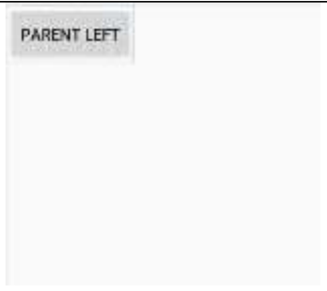
```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parent Bottom"
        android:layout_alignParentBottom="true"/>
</RelativeLayout>
```



- **android:layout_alignParentLeft :** Places the left of the element on the left side of the container

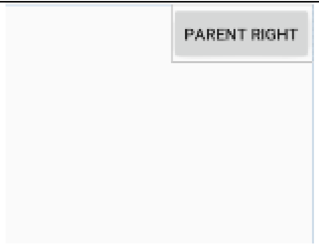
Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parent Left"
        android:layout_alignParentLeft="true"/>
</RelativeLayout>
```



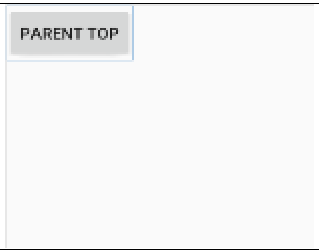
- **android:layout_alignParentRight :** Places the right of the element on the right side of the container

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parent Right"
        android:layout_alignParentRight="true"/>
</RelativeLayout>
```



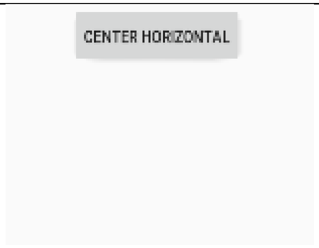
- **android:layout_alignParentTop :** Places the element at the top of the container

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parent Top"
        android:layout_alignParentTop="true"/>
</RelativeLayout>
```



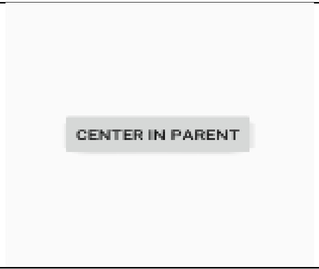
- **android:layout_centerHorizontal :** Centers the element horizontally within its parent container

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Center Horizontal"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```



- **android:layout_centerInParent :** Centers the element both horizontally and vertically within its container

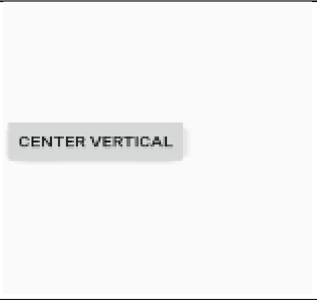
```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Center in Parent"
        android:layout_centerInParent="true"/>
</RelativeLayout>
```



Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**

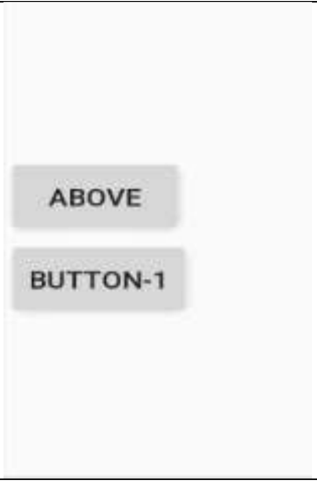
- **android:layout_centerVertical :** Centers the element vertically within its parent container.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Center Vertical"
        android:layout_centerVertical="true"/>
</RelativeLayout>
```



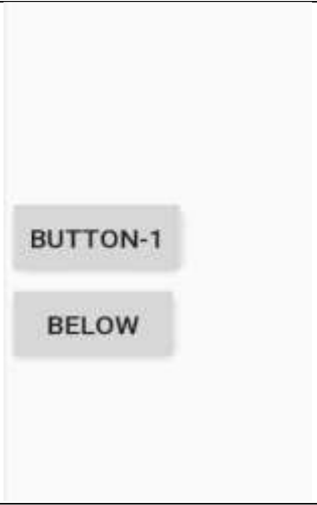
- **Relative to Siblings**
- **android:layout_above :** Places the element above the specified element

```
<RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <Button
   android:id="@+id/btn1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerVertical="true"
   android:text="Button-1" />
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Above"
   android:id="@+id/Above"
   android:layout_above="@id/btn1" />

</RelativeLayout>
```



- **android:layout_below :** Places the element below the specified element

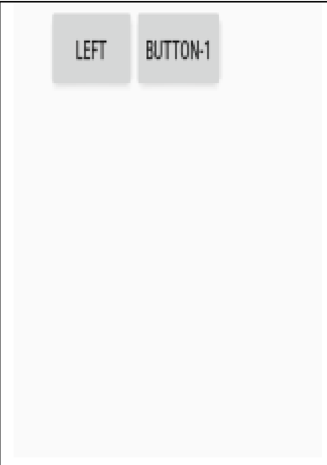
```
<RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <Button
   android:id="@+id/btn1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerVertical="true"
   android:text="Button-1" />
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="below"
   android:id="@+id/Below"
   android:layout_below="@id/btn1" />

</RelativeLayout>
```



- **android:layout_toLeftOf :** Places the element to the left of the specified element

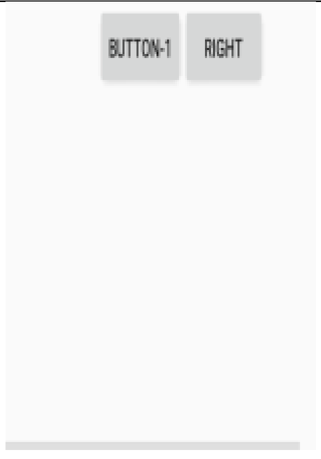
Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

```
 <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="Button-1" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Left"
    android:id="@+id/Left"
    android:layout_toLeftOf="@id/btn1" />

 </RelativeLayout>
```



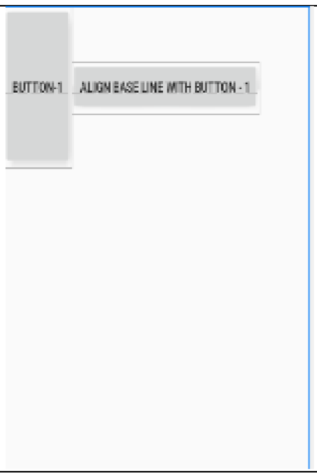
- **android:layout_toRightOf :** Places the element to the right of the specified element

```
 <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="Button-1" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Right"
    android:id="@+id/Right"
    android:layout_toRightOf="@id/btn1" />

 </RelativeLayout>
```



- **Alignment with Other Elements**
- **android:layout_alignBaseline :** Aligns baseline of the new element with the baseline of the specified element

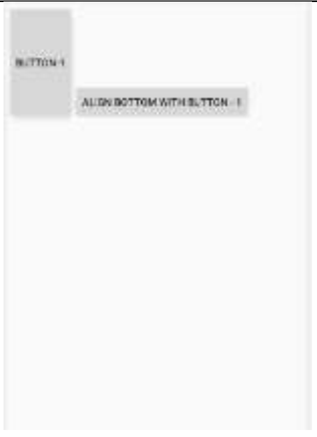
```
 <RelativeLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent">
     <Button
     android:id="@+id/btn1"
     android:layout_width="wrap_content"
     android:layout_height="150dp"
     android:text="Button-1" />
 <Button
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Align base line with button -
 1"
     android:layout_alignBaseline="@id/btn1"
     android:layout_toRightOf="@id/btn1"/>

 </RelativeLayout>
```



Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

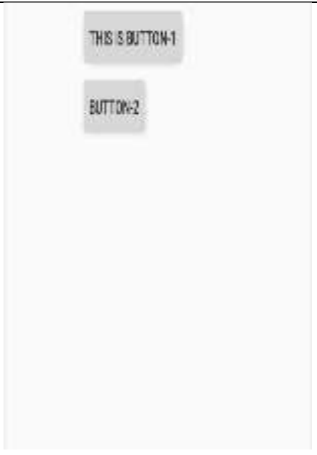
- **android:layout_alignBottom :** Aligns the bottom of new element in with the bottom of the specified element

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:text="Button-1" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Align bottom with button - 1"
    android:layout_alignBottom="@id/btn1"
    android:layout_toRightOf="@id/btn1"/>

</RelativeLayout>
```



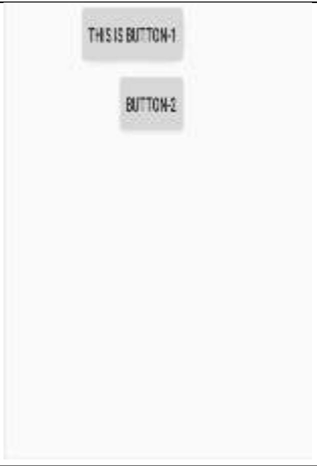
- **android:layout_alignLeft :** Aligns left edge of the new element with the left edge of the specified element

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is Button-1"
    android:layout_marginLeft="100dp"
    />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button-2"
    android:layout_below="@id/btn1"
    android:layout_alignLeft="@id/btn1"/>
</RelativeLayout>
```



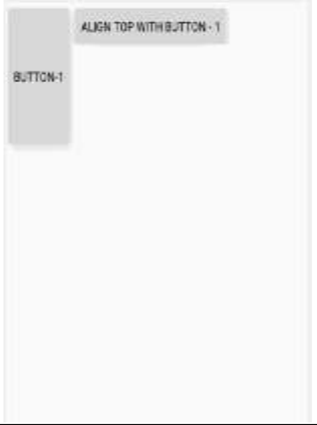
- **android:layout_alignRight :** Aligns right edge of the new element with the right edge of the specified element

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is Button-1"
    android:layout_marginLeft="100dp"
    />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button-2"
    android:layout_below="@id/btn1"
    android:layout_alignRight="@id/btn1"/>
</RelativeLayout>
```



- **android:layout_alignTop :** Places top of the new element in alignment with the top of the specified element

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:text="Button-1" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Align Top with button - 1"
    android:layout_alignTop="@id/btn1"
    android:layout_toRightOf="@id/btn1"/>
</RelativeLayout>
```

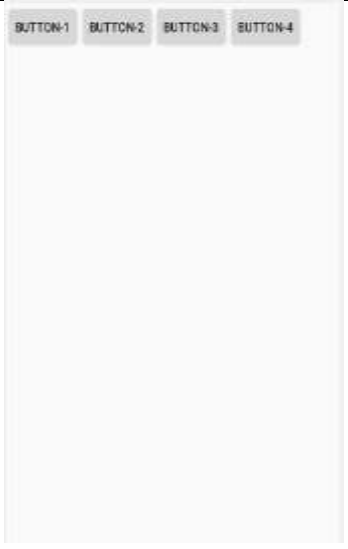


Android LinearLayout

Android LinearLayout organizes elements along a single line. We can specify whether that line is vertical or horizontal using android:orientation. The orientation is horizontal by default.

Linear Layout with horizontal orientation:

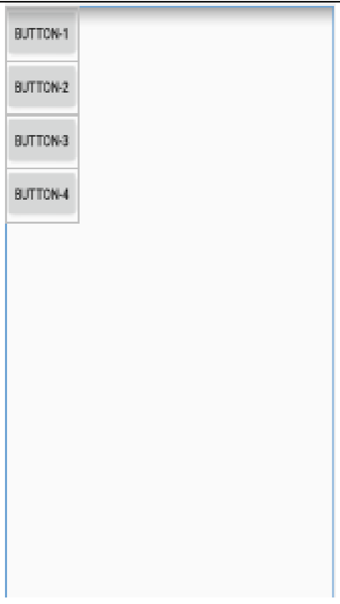
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-2" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-3" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-4" />
</LinearLayout>
```



Linear Layout with vertical orientation:

Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
android:orientation="vertical">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-2" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-3" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button-4" />
</LinearLayout>
```



A vertical LinearLayout will only have one child per row (so it is a column of single elements), and a horizontal LinearLayout will only have one single row of elements on the screen.

Example

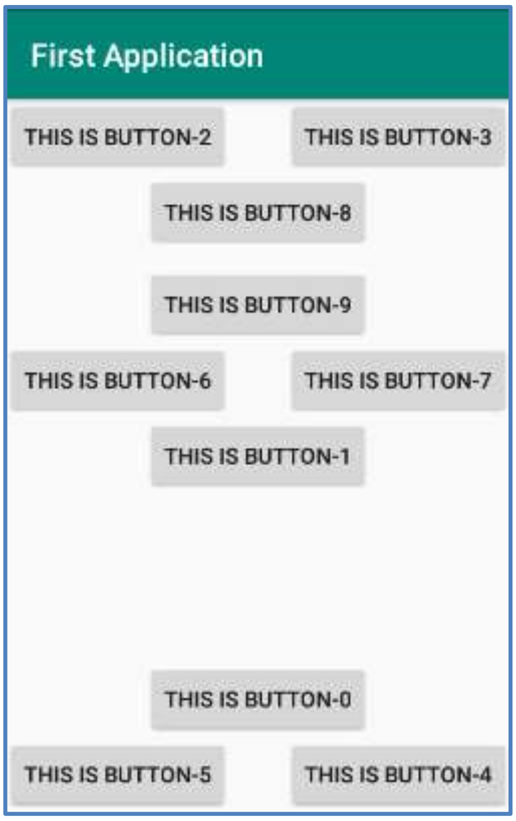
activity_main.xml

```
<?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"
    tools:context=".Main2Activity"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-1"
        android:layout_centerInParent="true"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-2" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-3"
        android:layout_alignParentRight="true"
        android:id="@+id/btn3"
     />


    <Button
        android:layout_width="wrap_content"
```

Zahid Ahmed email id: zahid@biit.edu.pk Whatsapp#03025367513

```
        android:layout_height="wrap_content"
        android:text="This is Button-4"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:id="@+id/btn4"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-5"
        android:layout_alignParentBottom="true"
        android:id="@+id/btn5"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-6"
       android:layout_above="@id/btn1"
        android:id="@+id/btn6"

        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-7"
        android:layout_above="@id/btn1"
        android:id="@+id/btn7"
        android:layout_alignParentRight="true"

        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-8"
        android:layout_below="@id/btn3"
        android:id="@+id/btn8"
        android:layout_centerHorizontal="true"

        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-9"
        android:layout_above="@id/btn7"
        android:id="@+id/btn9"
        android:layout_centerHorizontal="true"

        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Button-0"
        android:layout_above="@id/btn5"
        android:id="@+id/btn0"
        android:layout_centerHorizontal="true"

        />

</RelativeLayout>
```

Zahid Ahmed **email id: zahid@biit.edu.pk** **Whatsapp#03025367513**


First Application
THIS IS BUTTON-2
THIS IS BUTTON-3
THIS IS BUTTON-8
THIS IS BUTTON-9
THIS IS BUTTON-6
THIS IS BUTTON-7
THIS IS BUTTON-1
THIS IS BUTTON-0
THIS IS BUTTON-5
THIS IS BUTTON-4