**Chapter 1**

**INTRODUCTION TO MOBILE APPLICATION DEVELOPMENT**

Android is an [operating system](https://en.wikipedia.org/wiki/Operating_system). That is, it’s software that connects hardware to software and provides general services. But more than that, it’s a mobile specific operating system: an OS designed to work on mobile (read: handheld, wearable, carry-able) devices.

Note that the term “Android” also is used to refer to the “platform” (e.g., devices that use the OS) as well as the ecosystem that surrounds it. This includes the device manufacturers who use the platform, and the applications that can be built and run on this platform. So “Android Development” technically means developing applications that run on the specific OS, it also gets generalized to refer to developing any kind of software that interacts with the platform.

* 1. **Android History**
* **2003**: The platform was originally founded by a start-up “Android Inc.” which aimed to build a mobile OS operating system (similar to what [Nokia’s Symbian](https://en.wikipedia.org/wiki/Symbian) was doing at the time)
* **2005**: Android was acquired by Google, who was looking to get into mobile
* **2007**: Google announces the [Open Handset Alliance](https://www.openhandsetalliance.com/), a group of tech companies working together to develop “open standards” for mobile platforms. Members included phone manufacturers like HTC, Samsung, and Sony; mobile carriers like T-Mobile, Sprint, and NTT DoCoMo; hardware manufacturers like Broadcom and Nvidia; and others.
* **2008**: First Android device is released: the [HTC Dream](https://en.wikipedia.org/wiki/HTC_Dream) (a.k.a. T-Mobile G1)
  + Specs: 528Mhz ARM chip; 256MB memory; 320x480 resolution capacitive touch; slide-out keyboard! Author’s opinion: a fun little device.
* **2010**: First [Nexus](https://en.wikipedia.org/wiki/Google_Nexus) device is released: the [Nexus One](https://en.wikipedia.org/wiki/Nexus_One). These are Google-developed “flagship” devices, intended to show off the capabilities of the platform. **SPECS:** 1Ghz Scorpion; 512MB memory; .37” at 480x800 AMOLED capacitive touch.
* For comparison, the [iPhone 7 Plus](https://en.wikipedia.org/wiki/IPhone_7) (2016) has: 2.34Ghz dual core A10 64bit Fusion; 3GB RAM; 5.5" at 1920x1080 display.
* **2014**: Android Wear, a version of Android for wearable devices (watches) is announced.
* **2016**: Daydream, a virtual reality (VR) platform for Android is announced.

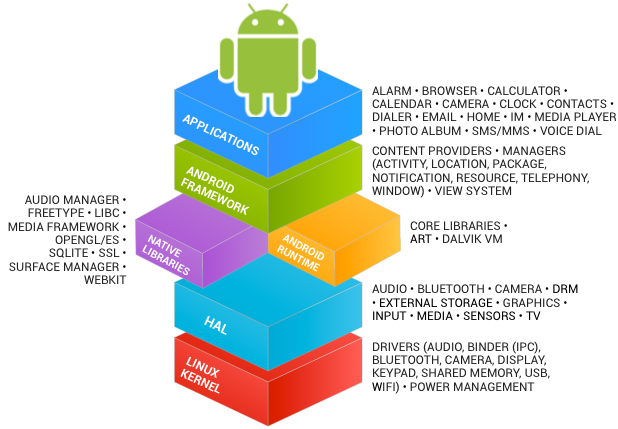
**1.1.1 Android Versions**

| **Date** | **Version** | **Nickname** | **API Level** |
| --- | --- | --- | --- |
| Sep 2008 | 1.0 | Android | 1 |
| Apr 2009 | 1.5 | Cupcake | 3 |
| Sep 2009 | 1.6 | Donut | 4 |
| Oct 2009 | 2.0 | Éclair | 5 |
| May 2010 | 2.2 | Froyo | 8 |
| Dec 2010 | 2.3 | Gingerbread | 9 |
| Feb 2011 | 3.0 | Honeycomb | 11 |
| Oct 2011 | 4.0 | Ice Cream Sandwich | 14 |
| July 2012 | 4.1 | Jelly Bean | 16 |
| Oct 2013 | 4.4 | KitKat | 19 |
| Nov 2014 | 5.0 | Lollipop | 21 |
| Oct 2015 | 6.0 | Marshmallow | 23 |
| Aug 2016 | 7.0 | Nougat | 24 |
| Mar 2017 | O preview | Android O Developer Preview |  |

***Table 1.1: Android Versions***

**1.2 Android Architecture**

Developing Android applications involves interfacing with the Android platform and framework. Thus you need a high level understanding of the architecture of the Android platform.



***Fig 1.1: Android Architecture***

The Android platform is built as a layered architecture:

* At it’s base, Android runs on a Linux kernel for interacting with the device’s processor, memory, etc. Thus, an Android device can be seen as a Linux computer.
* On top of that kernel is the Hardware Abstraction Layer: an interface to drivers that can programmatically access hardware elements, such as the camera, disk storage, WIFI antenna, etc.
  + These drivers are generally written in C; we won’t interact with them directly in this course.

**1.3 Programming Languages**

There are two programming languages we will be working with in this course:

1. **Java:** Android code (program control and logic, as well as data storage and manipulation) is written in Java.

Writing Android code will feel a lot writing any other Java program: you create classes, define methods, instantiate objects, and call methods on those objects. But because you’re working within a [**framework**](https://martinfowler.com/bliki/InversionOfControl.html)**,** there is a set of code that already exists to call specific methods. As a developer, your task will be to fill in what these methods do in order to run your specific application.

* + In web terms, this is closer to working with Angular (a framework) than jQuery (a library).
  + Importantly: this course expects you to have “journeyman”-level skills in Java (apprenticeship done, not yet master). We’ll be using a number of intermediate concepts (like generics and inheritance) without much fanfare or explanation.

1. **XML:** Android user interfaces and resources are specified in [XML](https://en.wikipedia.org/wiki/XML) (E**X**tensible **M**arkup **L**anguage). To compare to web programming: the XML contains what would normally go in the HTML/CSS, while the Java code will contain what would normally go in the JavaScript. XML is just like HTML, but you get to make up your own tags. Except we’ll be using the ones that Android made up; so it’s like defining web pages, except with a new set of elements. This course expects you to have some familiarity with HTML or XML, but if not you should be able to infer what you need from the examples.

**Chapter 2**

**ABOUT OUR PROJECT**

The name of our Project is **“Tic-Tac-Toe Game”**. This game is very popular and is fairly simple by itself. It is actually a two-player game. In this game, there is a board with n x n squares. In our game, it is 3 x 3 squares. The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally - on a 3 x 3 grid.

Our objective is to evolve several Tic-tac-toe strategies which never lose (meaning a draw or a win by the computer). This makes the problem to have a single objective of minimizing the number of losses. The evaluation of fitness of any strategy is done by first allowing it to play all possible games it could play, both as a first player and as a second player. For example, note from Figure 5 that there are two possible ways a game can move for the first player from level 1 to level 2, depending on whether the opponent made the left or the right-side move. Our evaluation procedure considers all such intermediate possibilities an opponent can have and count the total number of possible games resulting in wins, draws and losses. This is continued for the above strategy to be played as the second player. The total number of games lost in both cases as a first player and a second player is calculated.

****

***Fig 2.1: Tic-Tac-Toe Board***

**Chapter 3**

**SYSTEM REQUIREMENTS**

Software Requirement Specification (SRS) is a fundamental document, which forms the foundation of the software development process. SRS not only lists the requirements of a system but also has a description of its major features. These recommendations extend the IEEE standards. The recommendations would form the basis for providing clear visibility of the product to be developed serving as baseline for execution of a contract between client and the developer. SRS constitutes the agreement between clients and developers regarding the contents of the software product that is going to be developed. SRS should accurately and completely represent the system requirements as it makes a huge contribution to the overall project plan. The software being developed may be a part of the overall larger system or may be a complete standalone system in its own right.

**3.1 System Requirements**

**3.1.1 Hardware Requirements**

* **Processor** : Pentium(Any) or higher | AMD Athlon
* **Hard Disk**  : 2 GB or higher
* **Monitor** : 14.1 inch or higher
* **RAM**  : 1 GB or higher

**3.1.2 Software Requirements**

* **Operating System** : Windows 7 or higher
* **Coding Language** : XML | Java

**3.2 Installation Procedure of Android Studio**

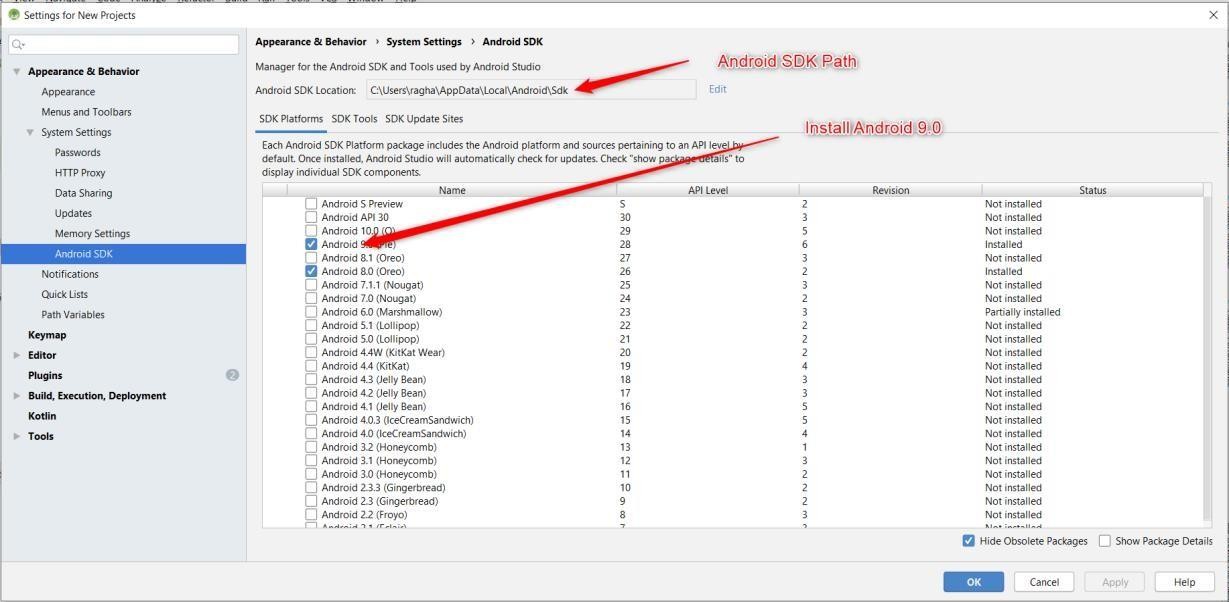
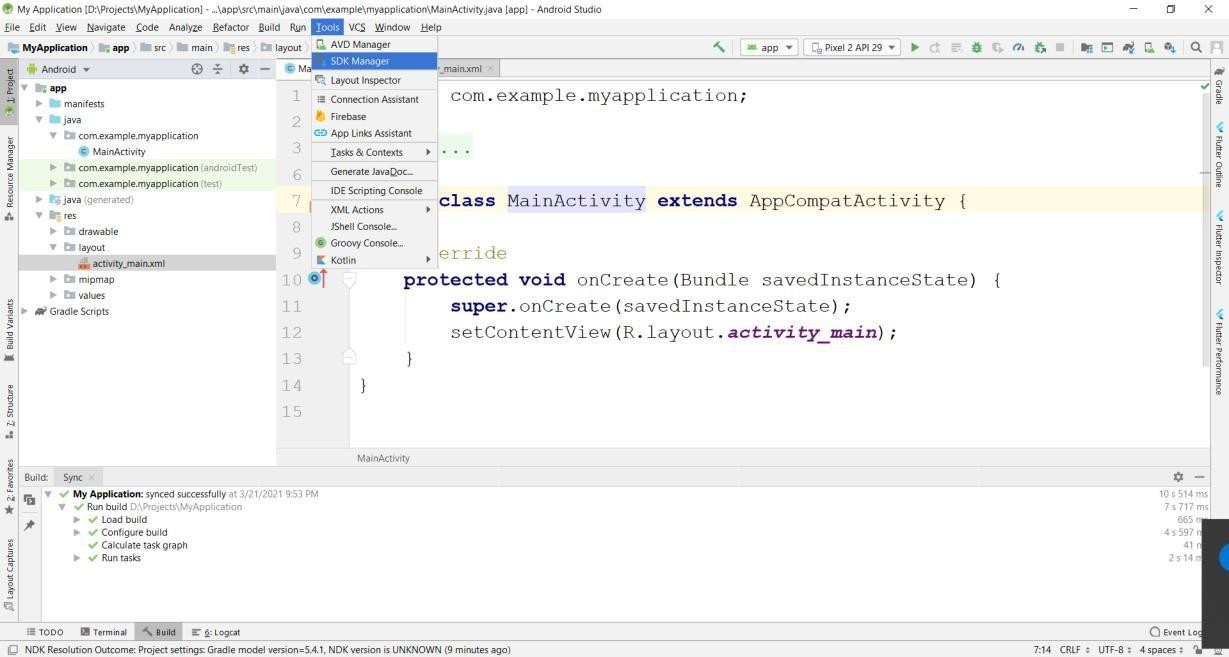
### Install Android Studio and Packages:

Download Android Studio from the below link:

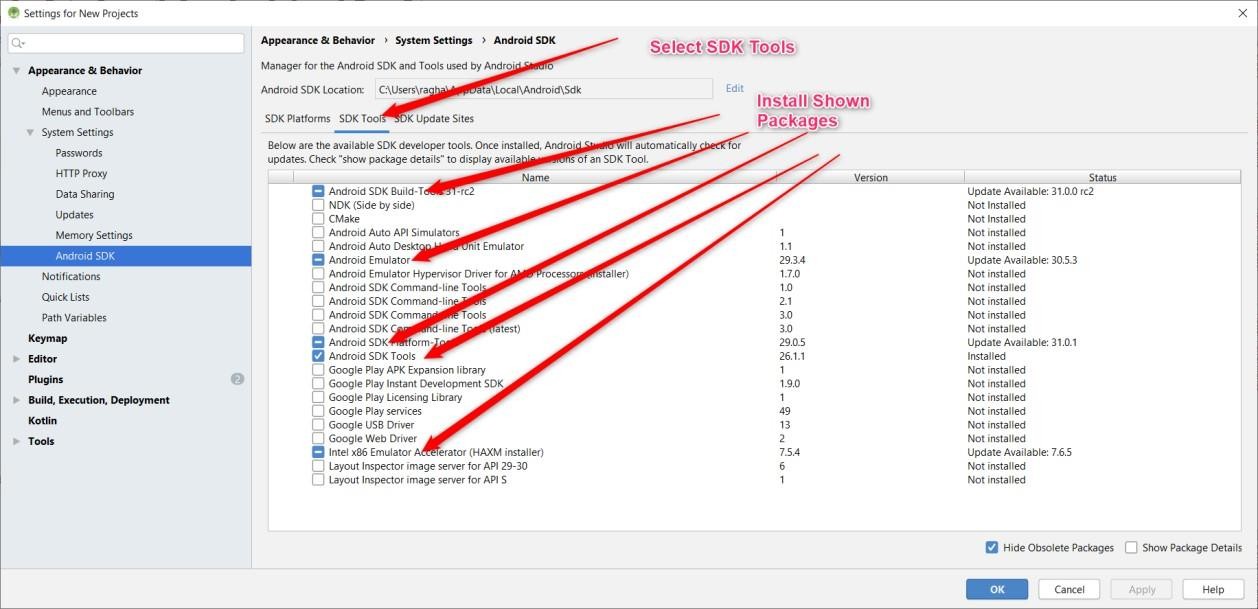
[*https://developer.android.com/studio*](https://developer.android.com/studio)

**Configuring Android SDK packages:**

Go to Tools -> SDK Manager

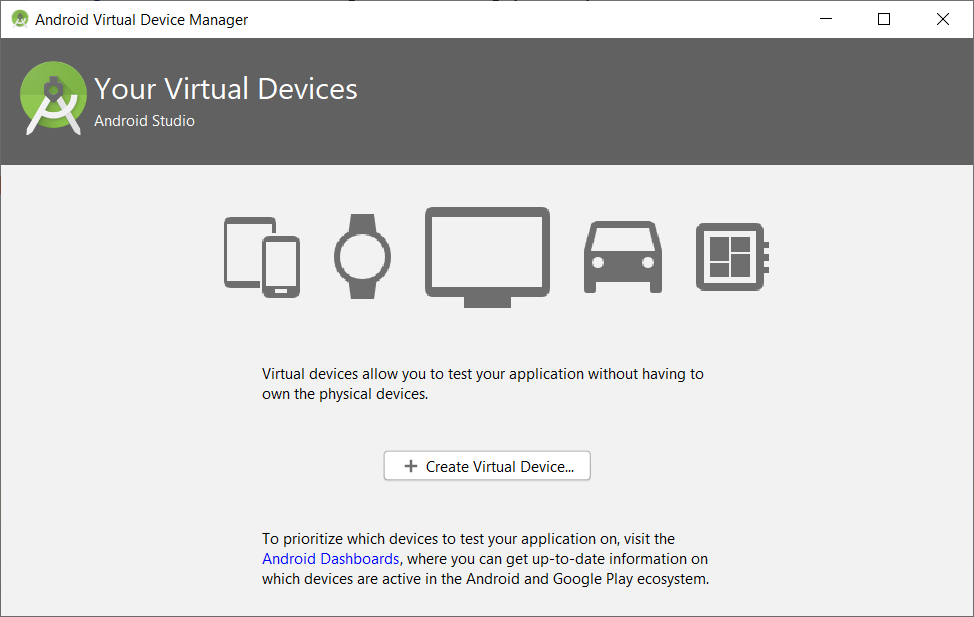
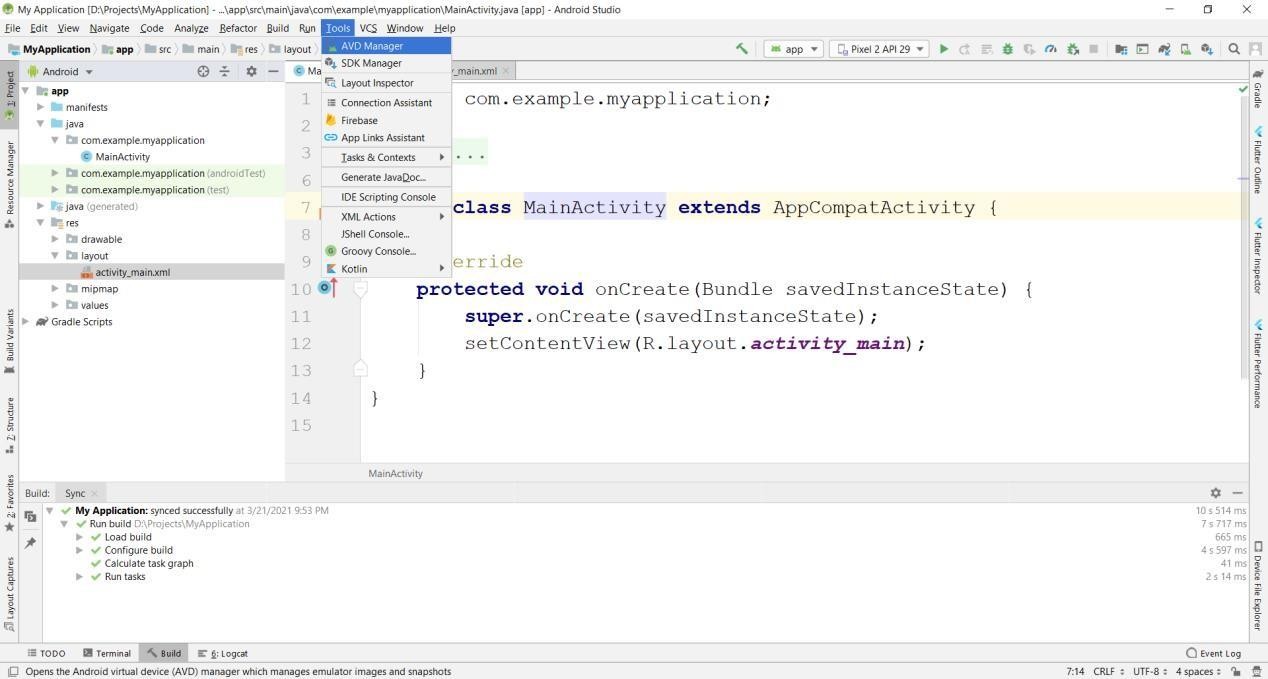


***Fig 3.1 – Configuring Android SDK Project***

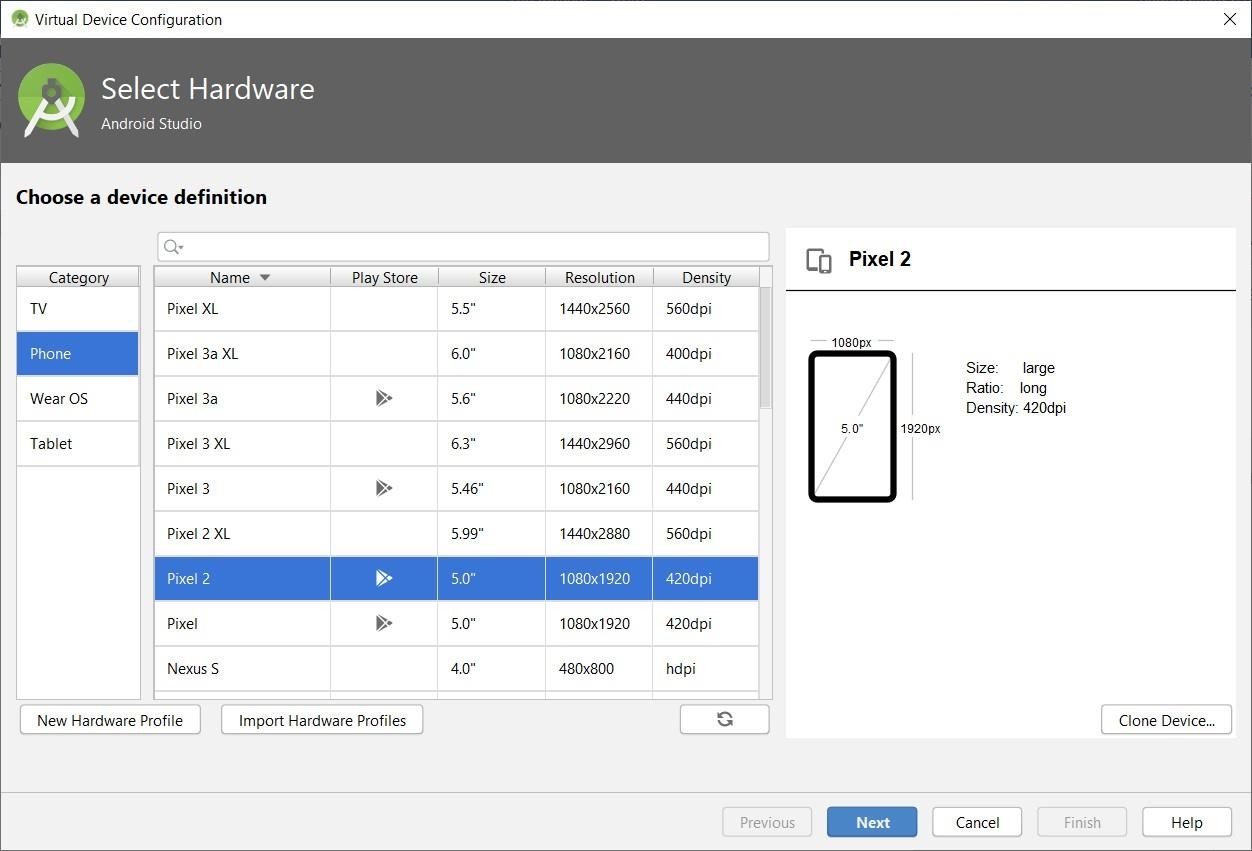


**Creating Emulator:**

Go to Tools -> Select AVD Manager

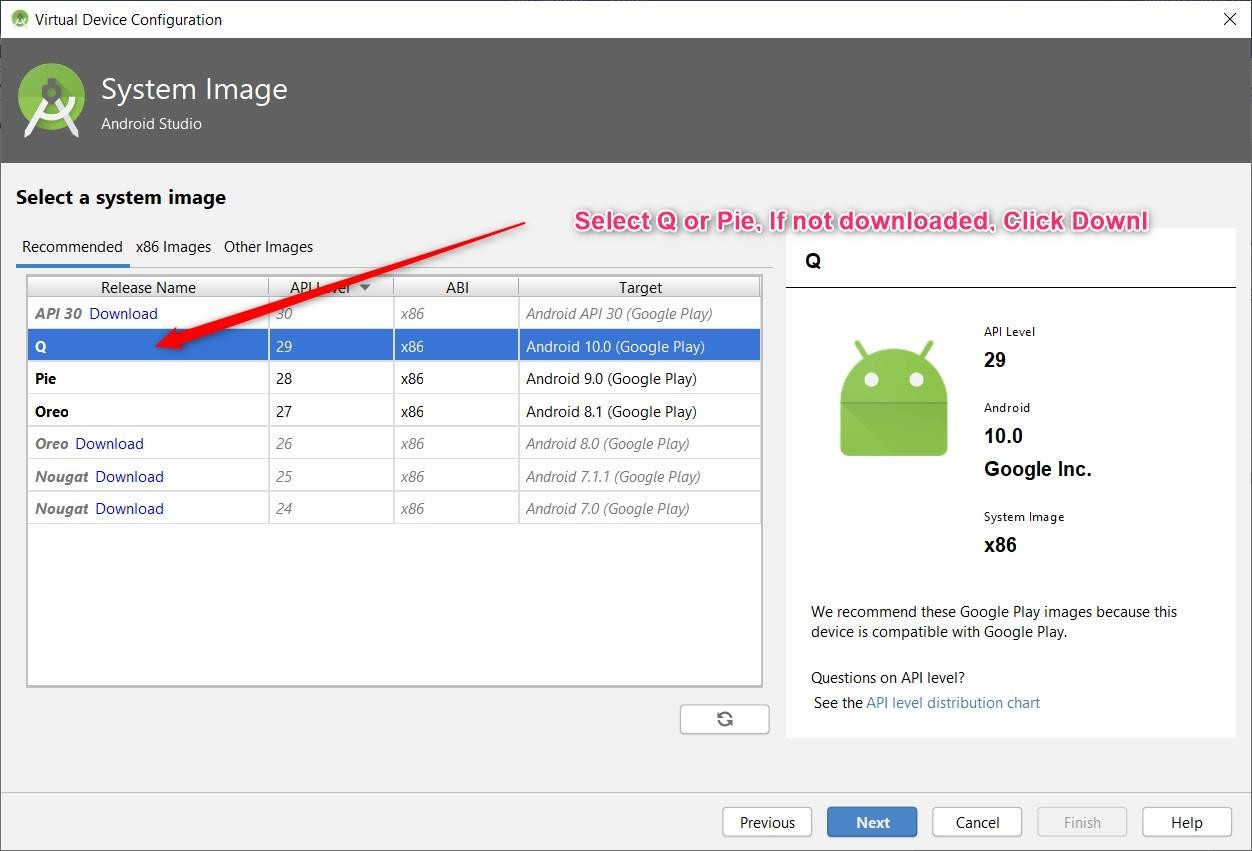


Select Create Virtual Device -> Select Phone -> Pixel 2 -> Press Next

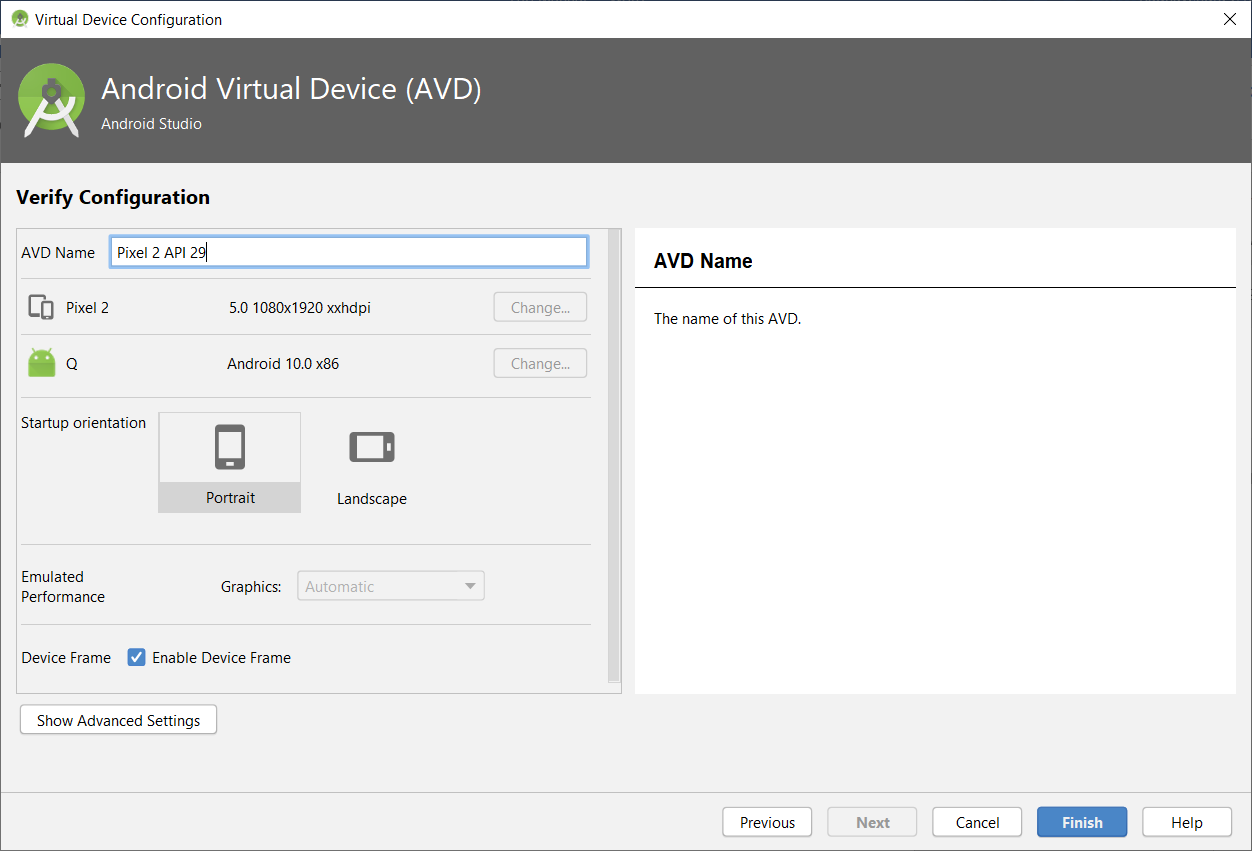


***Fig 3.2 – Creation of Virtual Device***

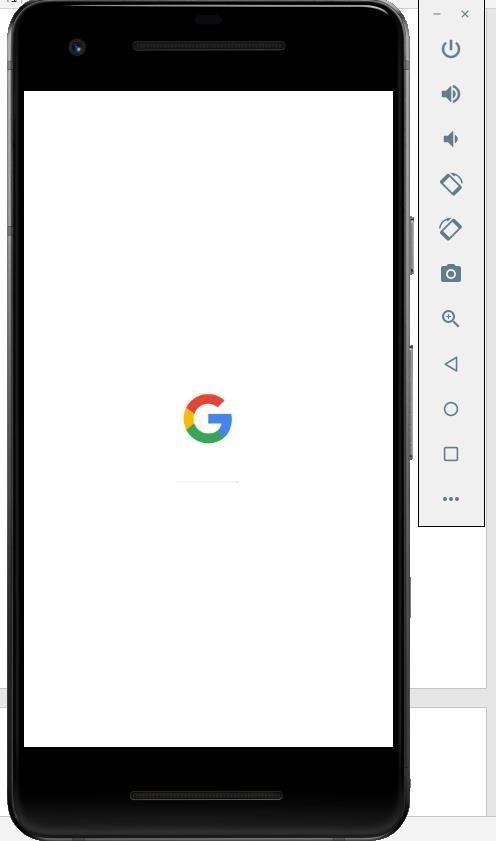
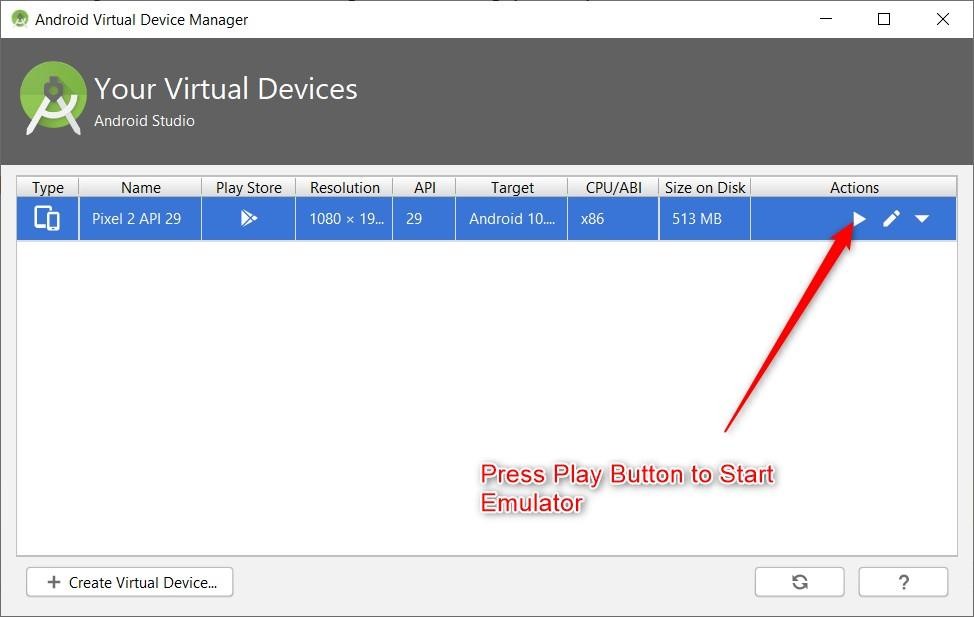
Select **Android Q**, if not already downloaded press download, after download completes Select **Q** and press **Next** button:



Enter AVD Name and Press Finish:

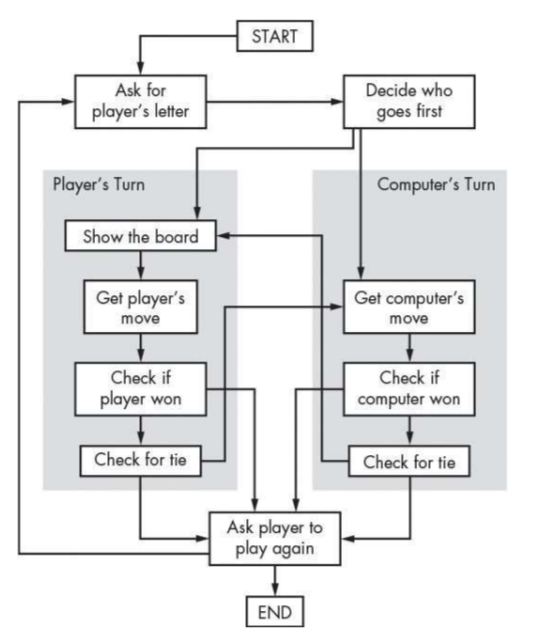


Press Play Button to Start Emulator:



***Fig 3.3 - Emulator***

**3.3 System Architecture**

****

***Fig 3.4 – System Architecture***

**Chapter 4**

**IMPLEMENTATION DETAILS**

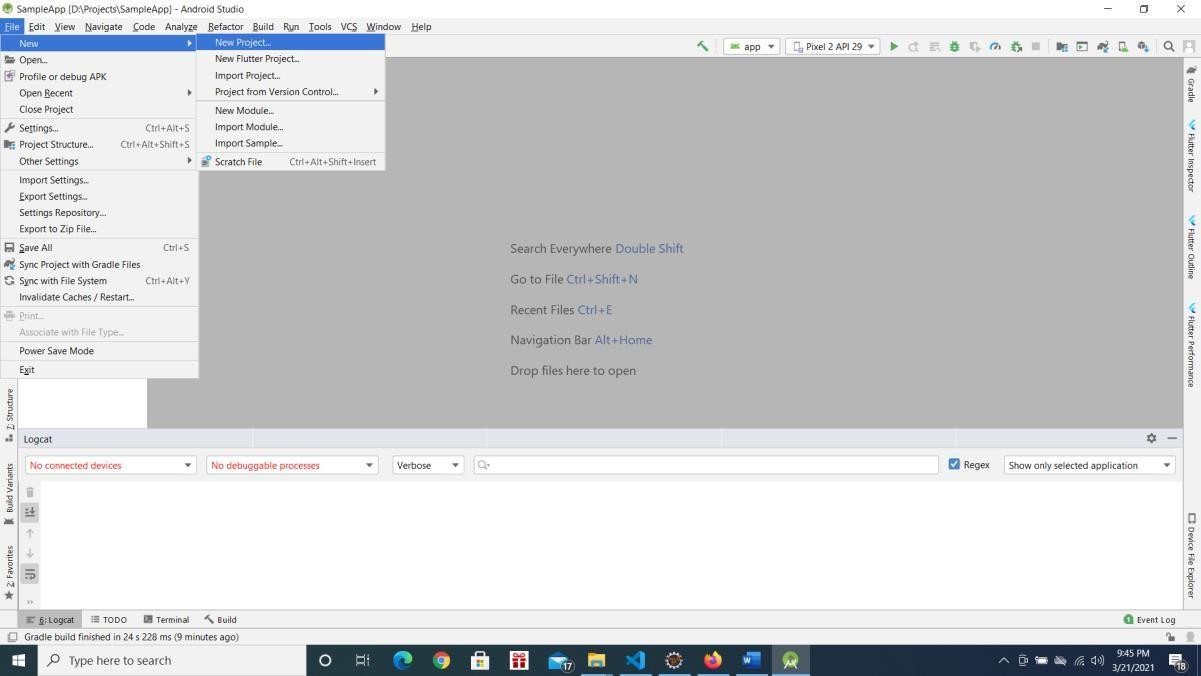
The implementation phase involves more than just writing code. Code also needs to be tested and debugged as well as compiled and built into a complete executable product. We usually need to utilize configuration management in order to keep track of different version of code. This is the stage of the project where the theoretical design is turned into a working system. If the implementation is not carefully planned and controlled, it can cause chaos and confusions. It is always a good idea to keep in mind that some characteristics that should be found in a good implementation like Readability- our code is written in MVC Architecture, JAVA to achieve the objective of the project that is to introduce a novel scheme of mechanism design for balancing the resource consumptions.

**4.1 Creation of New Project**

#### Creating a New Project in Android Studio:

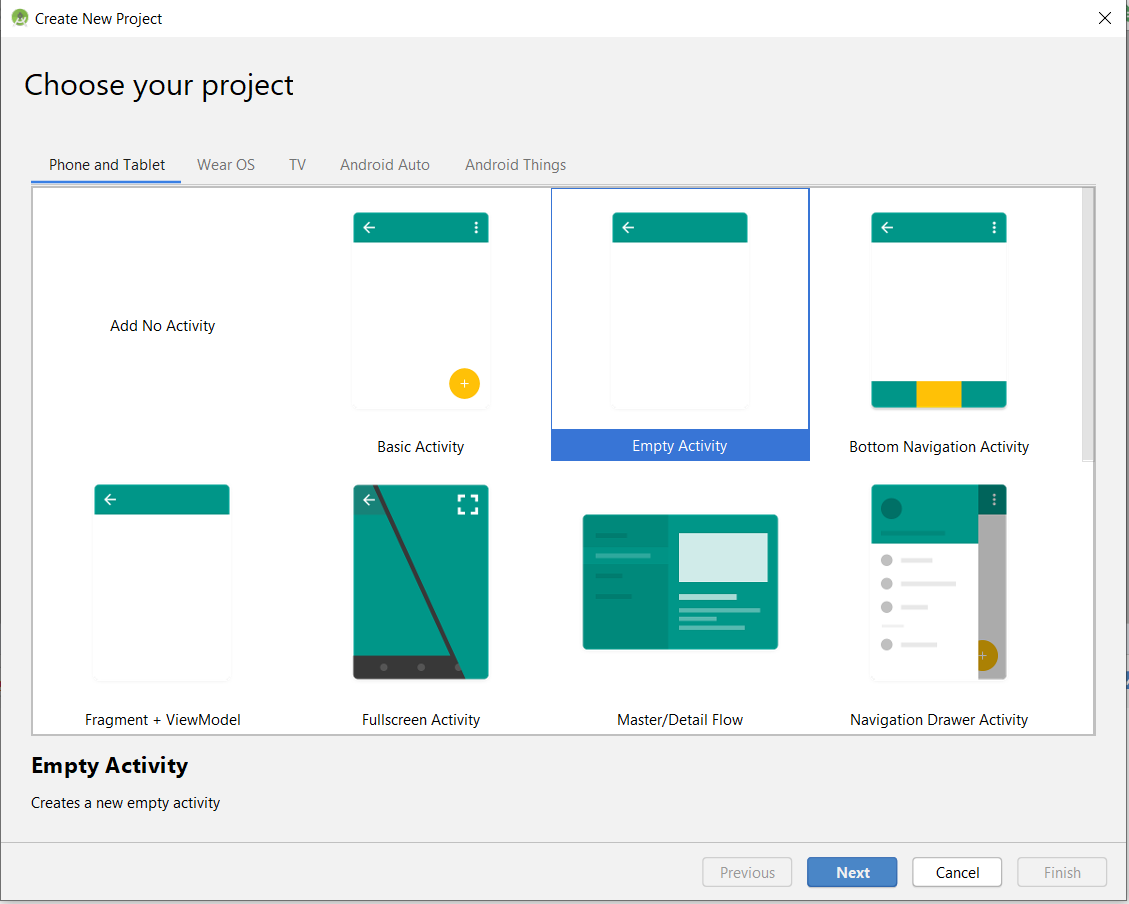
While creating a New Project for First Time, make sure Android Studio is connected to internet. It downloads the required packages from internet.

Go to File -> New -> New Project



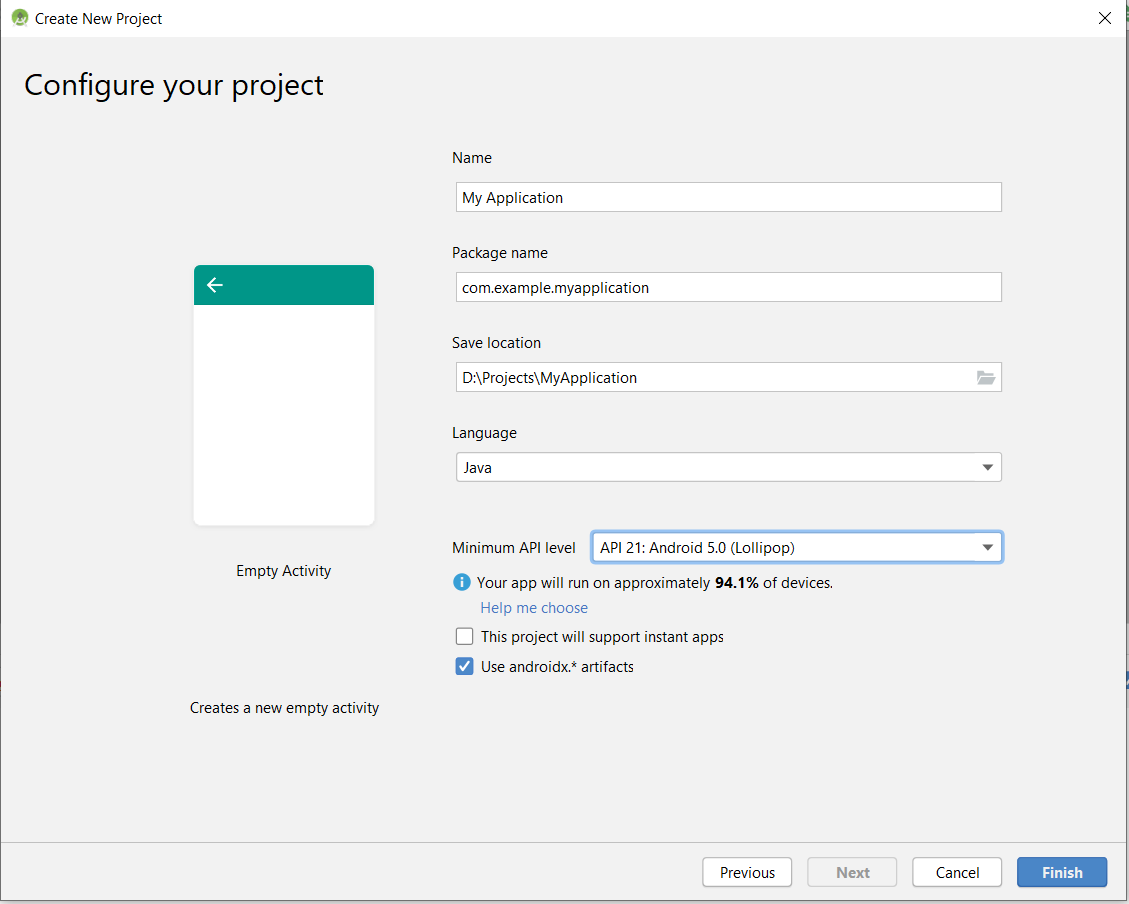
***Fig 4.1 – Creation of new project***

Choose Phone and Tablet -> Empty Activity -> Press Next

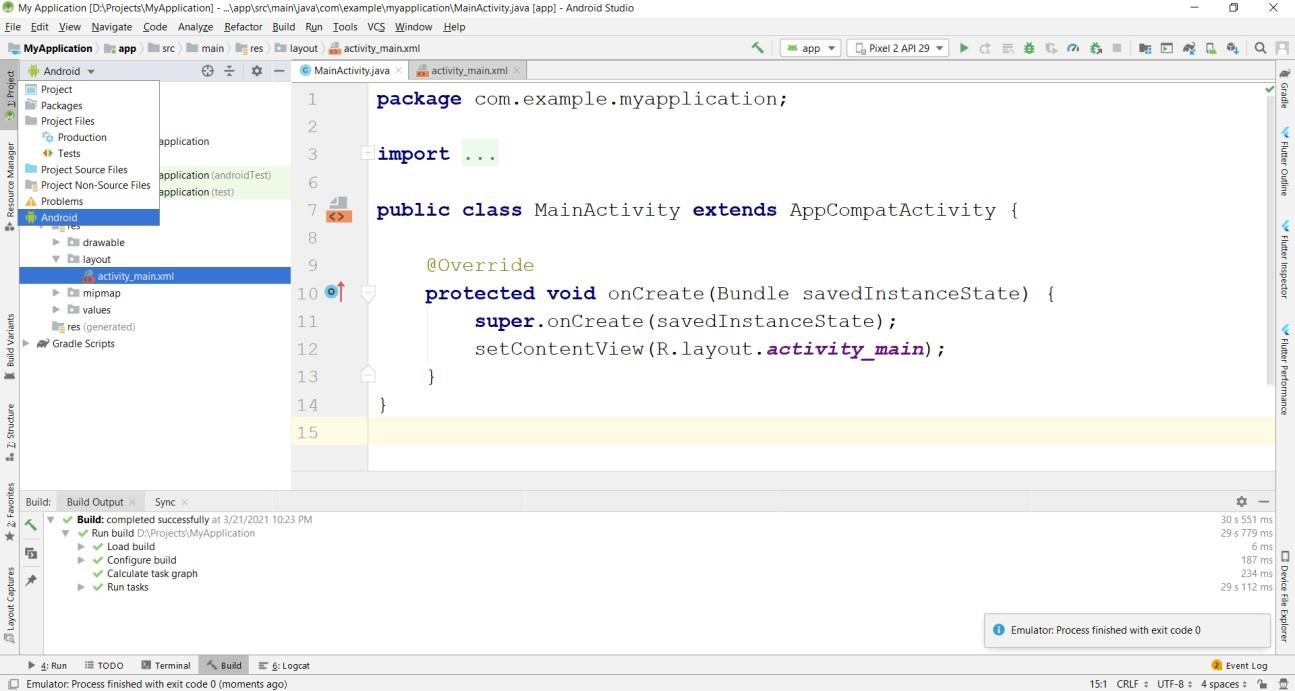


* In Configure your Project Screen, Enter below details and Press Finish Button.
* Enter Name of the Application -> This will be application name this will be visible with Home Screen Icon.
* Package Name -> Enter package name at least two identifier (eg: com.example). Best Practice is 3 or more identifier (eg: com.example.firstapp).
* Save Location -> Location where to save the Project Language -> Choose Java
* Minimum API Level -> Android 5.0
* Select Checkbox Use android.artifacts folder as below screenshot.

***Fig 4.2 – Configuring the Project***

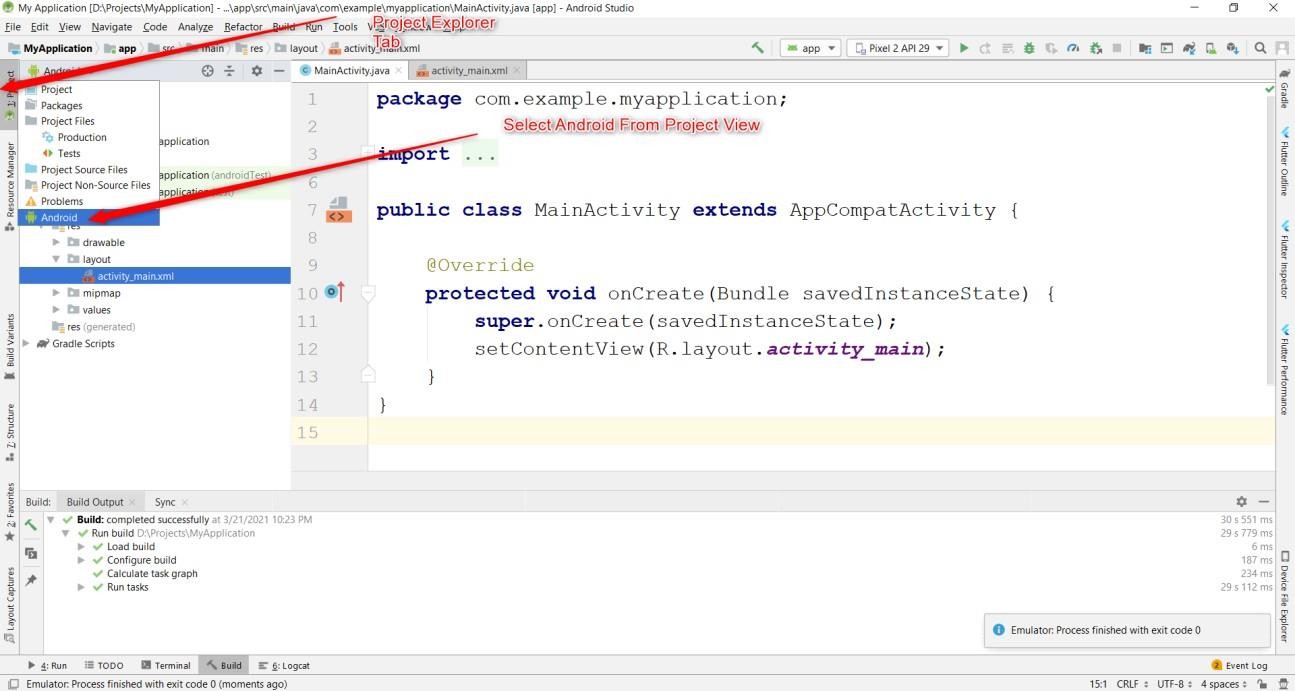


Android Project Structure:

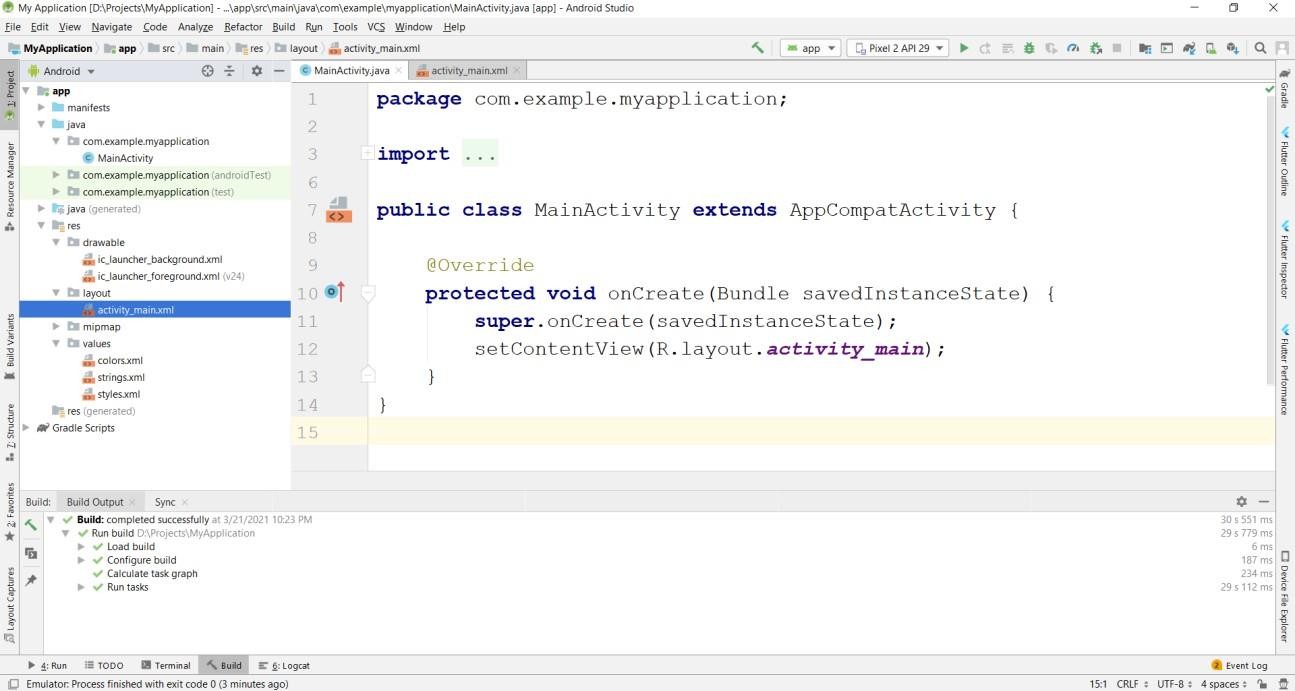


***Fig 4.3 – Android Project Structure***

Select Project Explorer and Select Android from Project View:



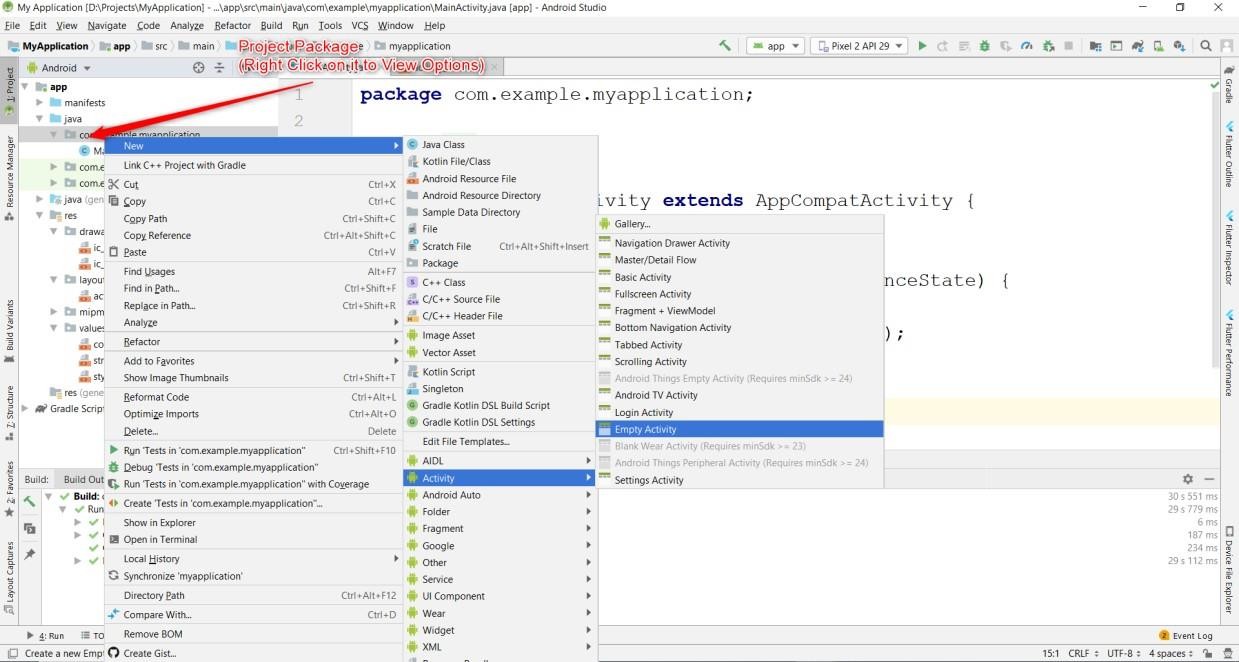
Basic View:



***Fig 4.4 – Basic View of IDE***

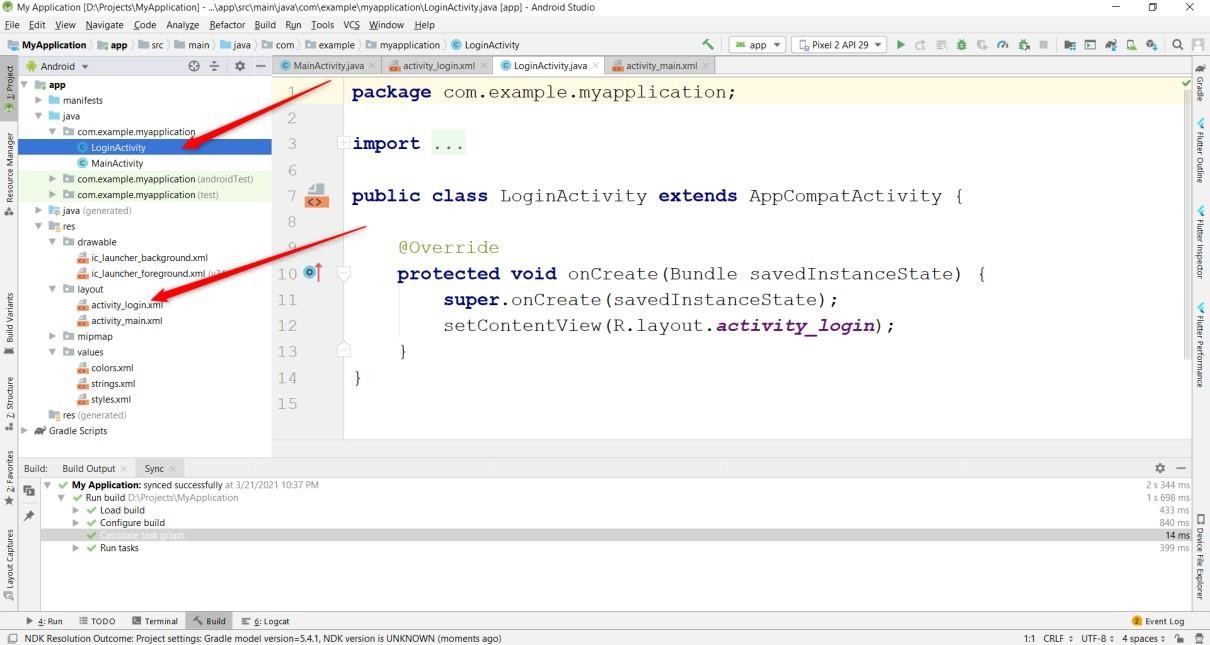
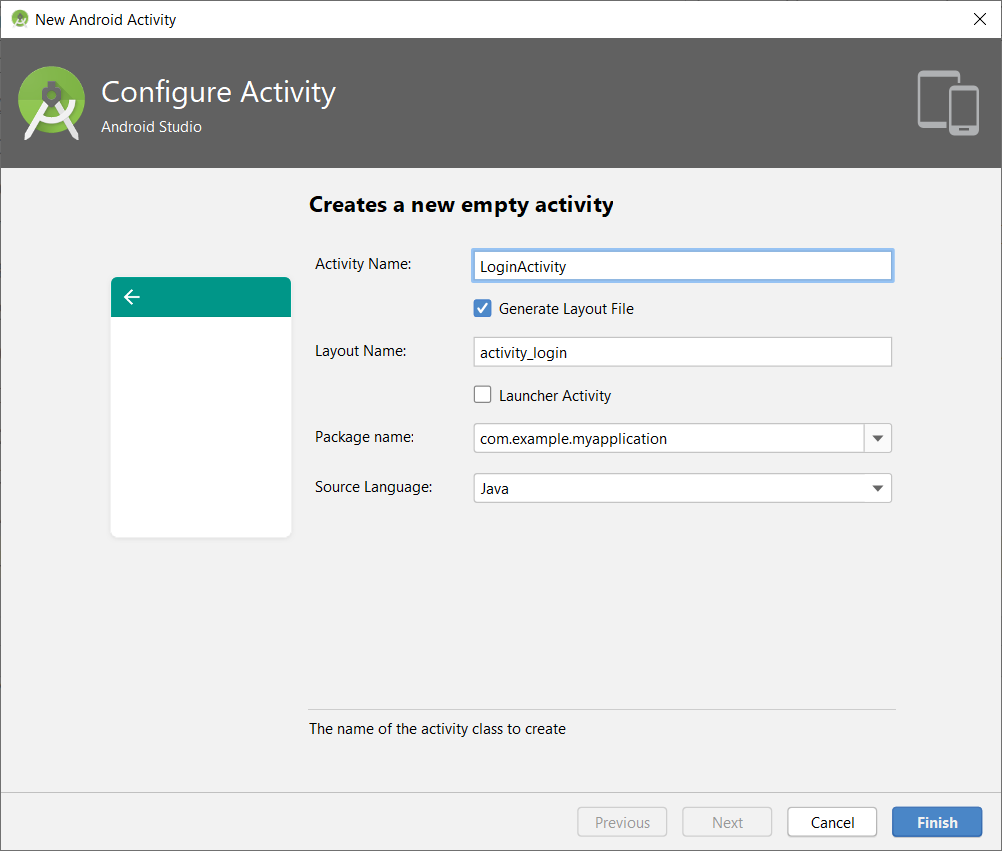
**Creating an Activity in Android Studio:**

Right Click on Package -> New -> Activity-> Empty Activity



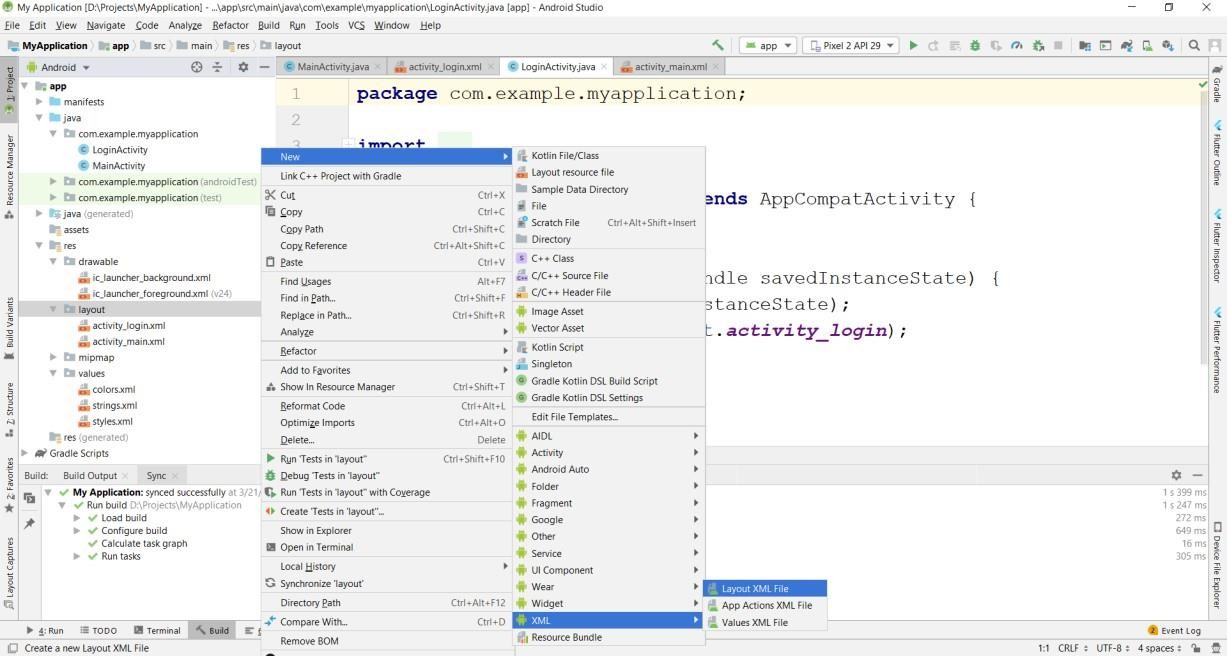
***Fig 4.5 – Creation of Android Activity***

Enter Activity Name and Press Finish:



#### Creating a Layout in Android Studio:

Right Click on Layout Folder -> New -> XML -> Layout XML File



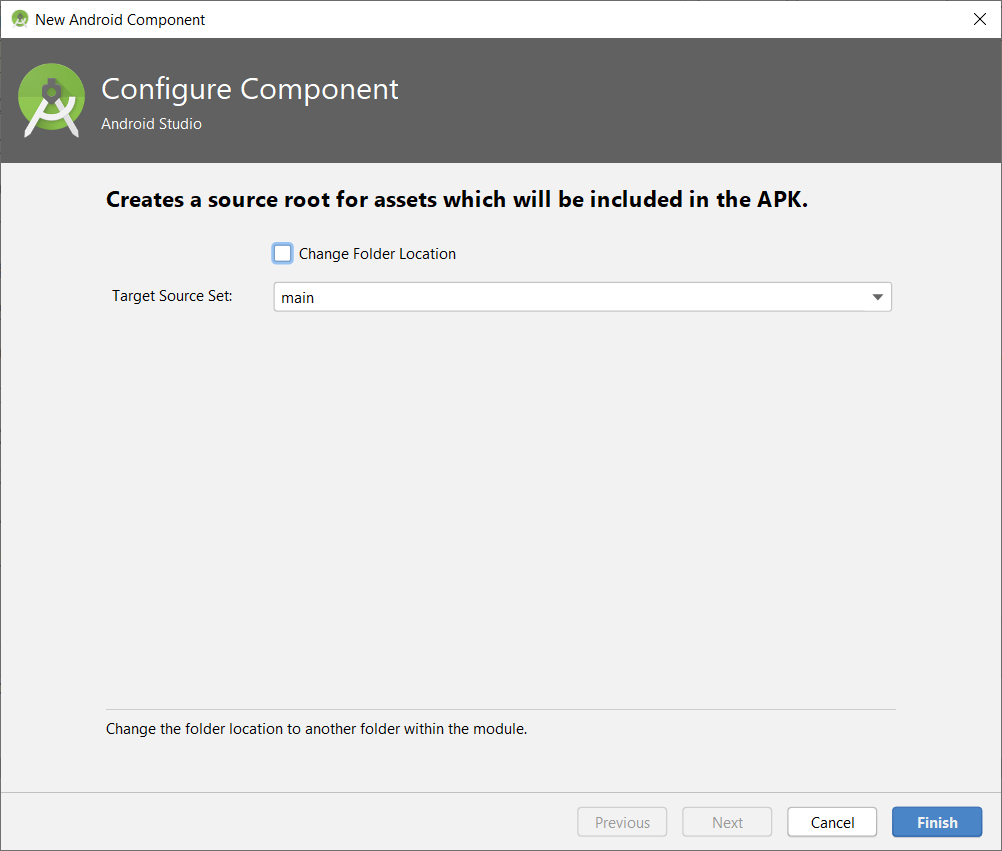
***Fig 4.6 – Creation of Layout***

Enter xml file name and press Finish:

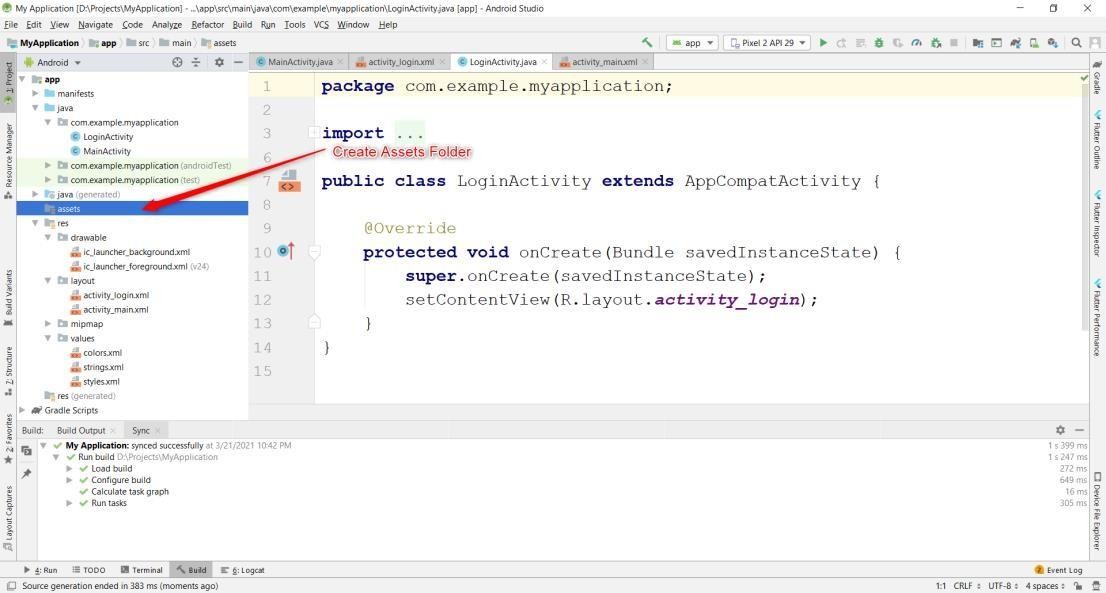
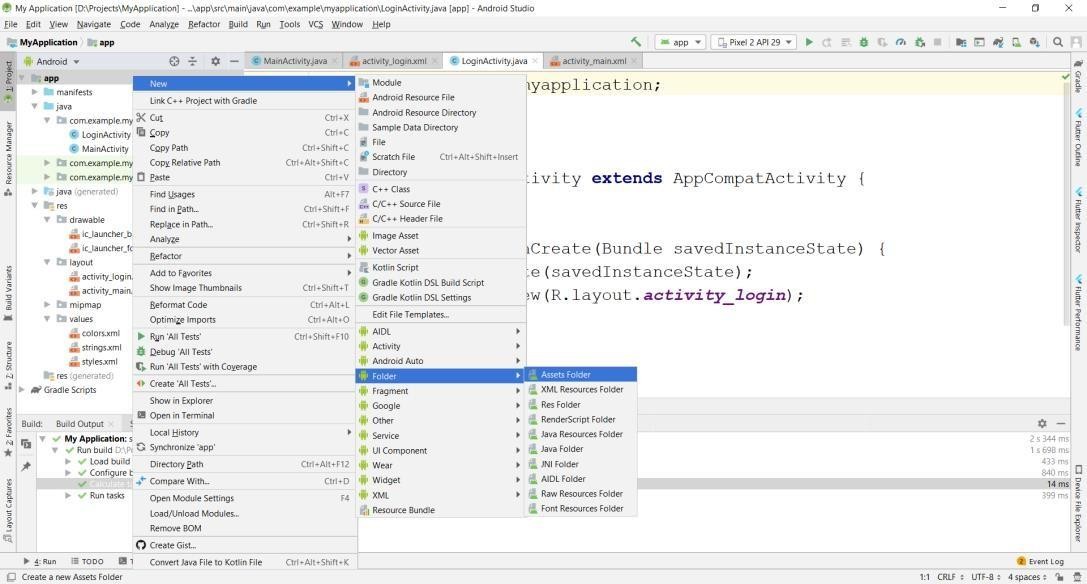


#### Creating Assets Folder in Android Studio:

Right Click on app folder -> New -> Folder -> Assets Folder -> Press Finish Button

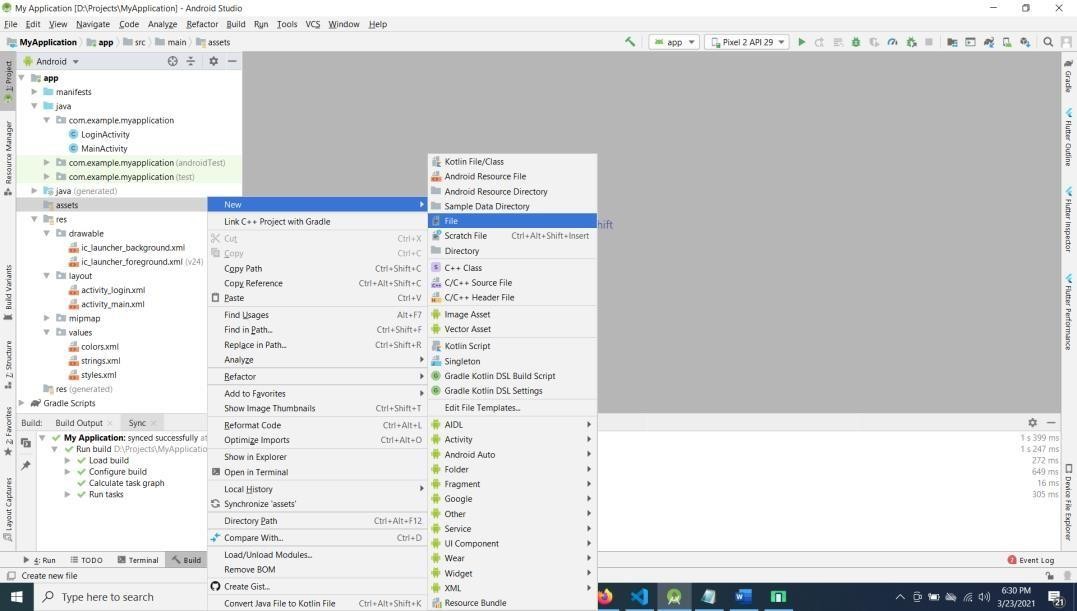


***Fig 4.7 – Creation of assets folder***



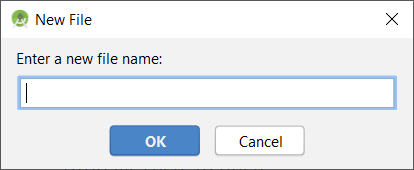
#### Creating File in assets Folder:

Right Click on assets folder -> New -> File



***Fig 4.8 – Creating File in assets folder***

Enter filename with extension (eg: abc.xml)



**4.2 Code Lines for the Application**

**4.2.1 mainactivity.java:**

package prajwal.tictactoe;

import android.content.Intent;

import android.os.Bundle;

import android.view.inputmethod.EditorInfo;

import android.widget.CheckBox;

import android.widget.EditText;

import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

public EditText plyr1;

public EditText plyr2;

public CharSequence player1 = "Player 1";

public CharSequence player2 = "Player 2";

public CharSequence cloneplayer2;

boolean player1ax = true;

boolean selectedSinglePlayer;

boolean easy = true;

boolean medium = false;

boolean hard = false;

boolean impossible = false;

public CheckBox p1x, p1o, p2x, p2o, singleplayer, twoplayer;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//apply the animation ( fade In ) to your LAyout

if (getIntent().getBooleanExtra("EXIT", false)) {

finish();

}

plyr1 = (EditText) findViewById(R.id.playerone);

plyr2 = (EditText) findViewById(R.id.playertwo);

p1x = (CheckBox) findViewById(R.id.player1x);

p1o = (CheckBox) findViewById(R.id.player1o);

p2x = (CheckBox) findViewById(R.id.player2x);

p2o = (CheckBox) findViewById(R.id.player2o);

singleplayer = (CheckBox) findViewById(R.id.splayer);

twoplayer = (CheckBox) findViewById(R.id.tplayer);

p1x.setOnClickListener(checkboxClickListener);

p1o.setOnClickListener(checkboxClickListener);

p2x.setOnClickListener(checkboxClickListener);

p2o.setOnClickListener(checkboxClickListener);

singleplayer.setOnClickListener(checkboxClickListener);

twoplayer.setOnClickListener(checkboxClickListener);

p1x.setChecked(true);

p2o.setChecked(true);

twoplayer.setChecked(true);

}

View.OnClickListener checkboxClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

boolean checked = ((CheckBox) view).isChecked();

if (checked) {

switch (view.getId()) {

case R.id.player1x:

p1o.setChecked(false);

p2x.setChecked(false);

p2o.setChecked(true);

player1ax = true;

break;

case R.id.player1o:

p1x.setChecked(false);

p2o.setChecked(false);

p2x.setChecked(true);

player1ax = false;

break;

case R.id.player2x:

p2o.setChecked(false);

p1x.setChecked(false);

p1o.setChecked(true);

player1ax = false;

break;

case R.id.player2o:

p2x.setChecked(false);

p1o.setChecked(false);

p1x.setChecked(true);

player1ax = true;

break;

case R.id.splayer:

twoplayer.setChecked(false);

selectedSinglePlayer = true;

cloneplayer2 = player2;

plyr2.setText("CPU");

plyr1.setImeOptions(EditorInfo.IME\_ACTION\_DONE);

plyr1.setImeActionLabel("DONE", EditorInfo.IME\_ACTION\_DONE);

break;

case R.id.tplayer:

singleplayer.setChecked(false);

selectedSinglePlayer = false;

plyr2.setText(cloneplayer2);

plyr1.setImeOptions(EditorInfo.IME\_ACTION\_NEXT);

plyr1.setImeActionLabel("NEXT", EditorInfo.IME\_ACTION\_NEXT);

break;

}

} else {

switch (view.getId()) {

case R.id.player1x:

p1o.setChecked(true);

p2x.setChecked(true);

p2o.setChecked(false);

player1ax = false;

break;

case R.id.player1o:

p1x.setChecked(true);

p2o.setChecked(true);

p2x.setChecked(false);

player1ax = true;

break;

case R.id.player2x:

p2o.setChecked(true);

p1x.setChecked(true);

p1o.setChecked(false);

player1ax = true;

break;

case R.id.player2o:

p2x.setChecked(true);

p1o.setChecked(true);

p1x.setChecked(false);

player1ax = false;

break;

case R.id.splayer:

twoplayer.setChecked(true);

selectedSinglePlayer = false;

plyr2.setText(cloneplayer2);

plyr1.setImeOptions(EditorInfo.IME\_ACTION\_NEXT);

plyr1.setImeActionLabel("NEXT", EditorInfo.IME\_ACTION\_NEXT);

break;

case R.id.tplayer:

singleplayer.setChecked(true);

selectedSinglePlayer = true;

plyr2.setText("CPU");

plyr1.setImeOptions(EditorInfo.IME\_ACTION\_DONE);

plyr1.setImeActionLabel("DONE", EditorInfo.IME\_ACTION\_DONE);

break;

}

}

}

};

public void startgame(View view) {

if (!selectedSinglePlayer){

if (plyr2.getText().toString().length() == 0) {

player2 = "player 2";

}

else{

player2=plyr2.getText().toString();

}

if (plyr1.getText().toString().length() == 0) {

player1 = "player 1";

}

else{

player1=plyr1.getText().toString();

}}

if (selectedSinglePlayer){

player2=plyr2.getText().toString();

if (plyr1.getText().toString().length() == 0) {

player1 = "YOU";

}

else{

player1=plyr1.getText().toString();

}}

CharSequence[] players = {player1, player2};

Intent i = new Intent(this, Afterstart.class);

i.putExtra("playersnames", players);

i.putExtra("player1ax", player1ax);

i.putExtra("selectedsingleplayer", selectedSinglePlayer);

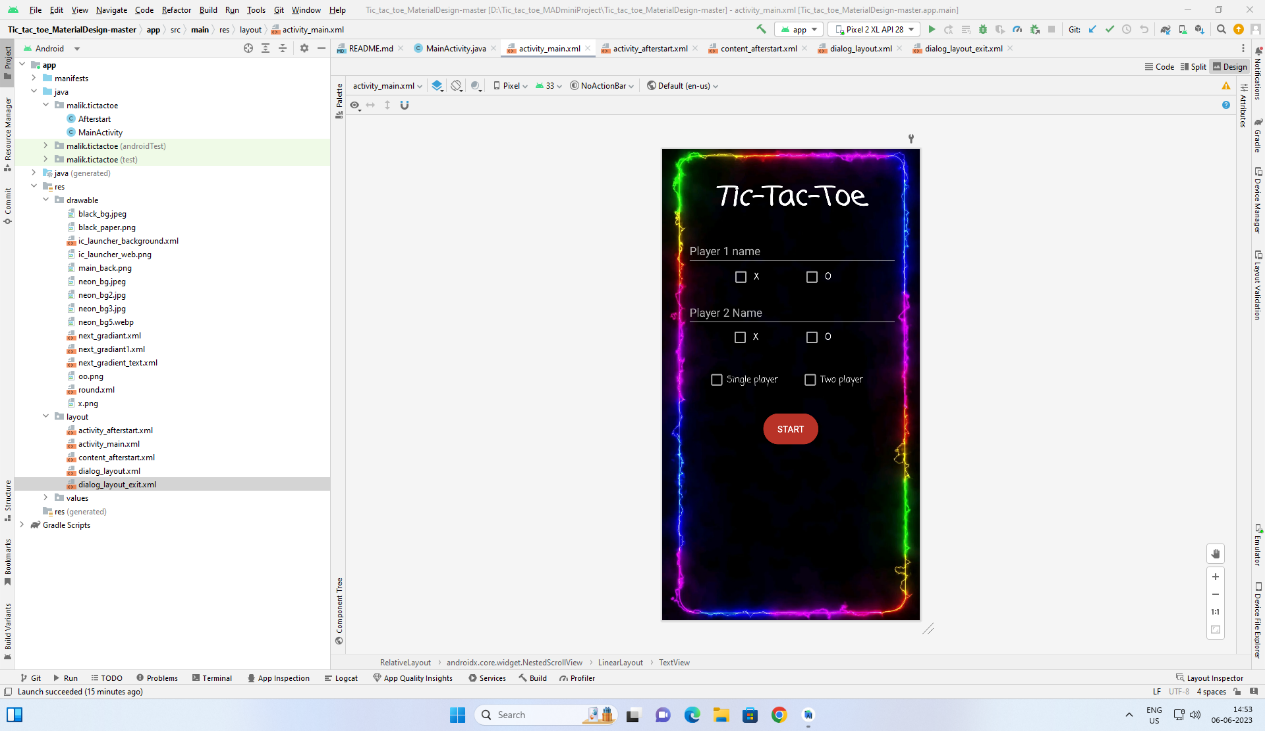
startActivity(i);

}

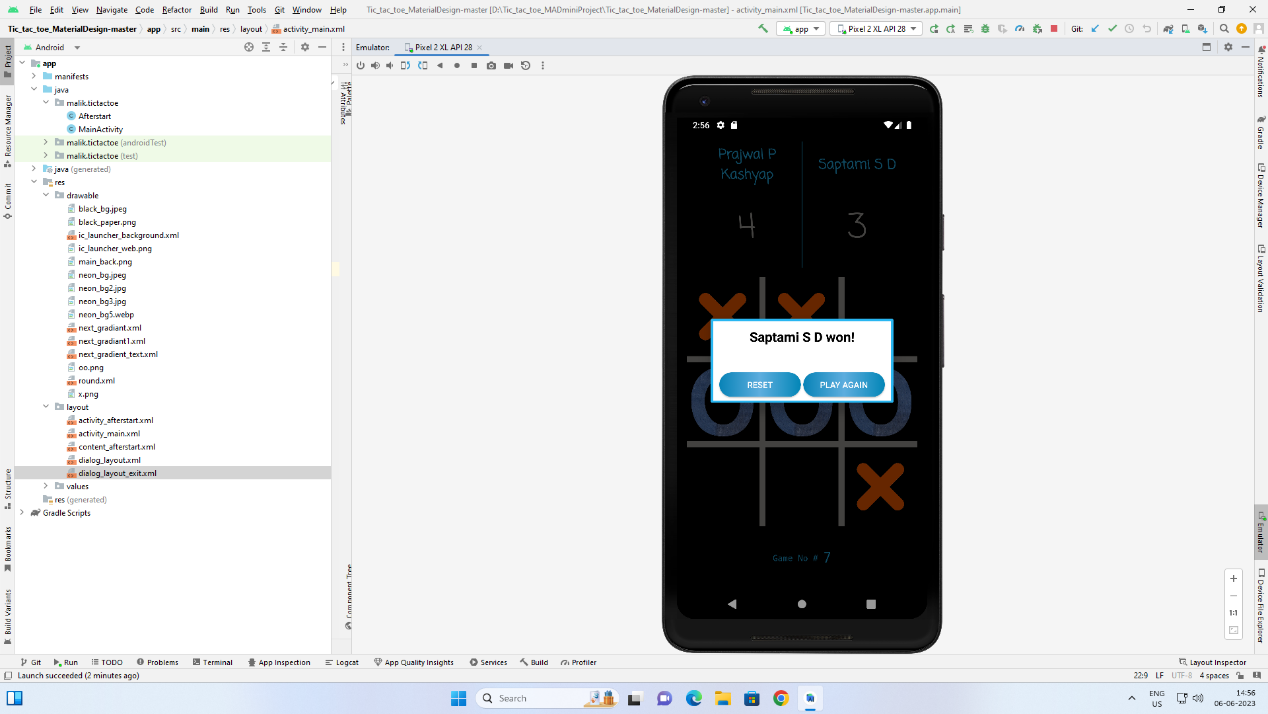
}

**Chapter 5**

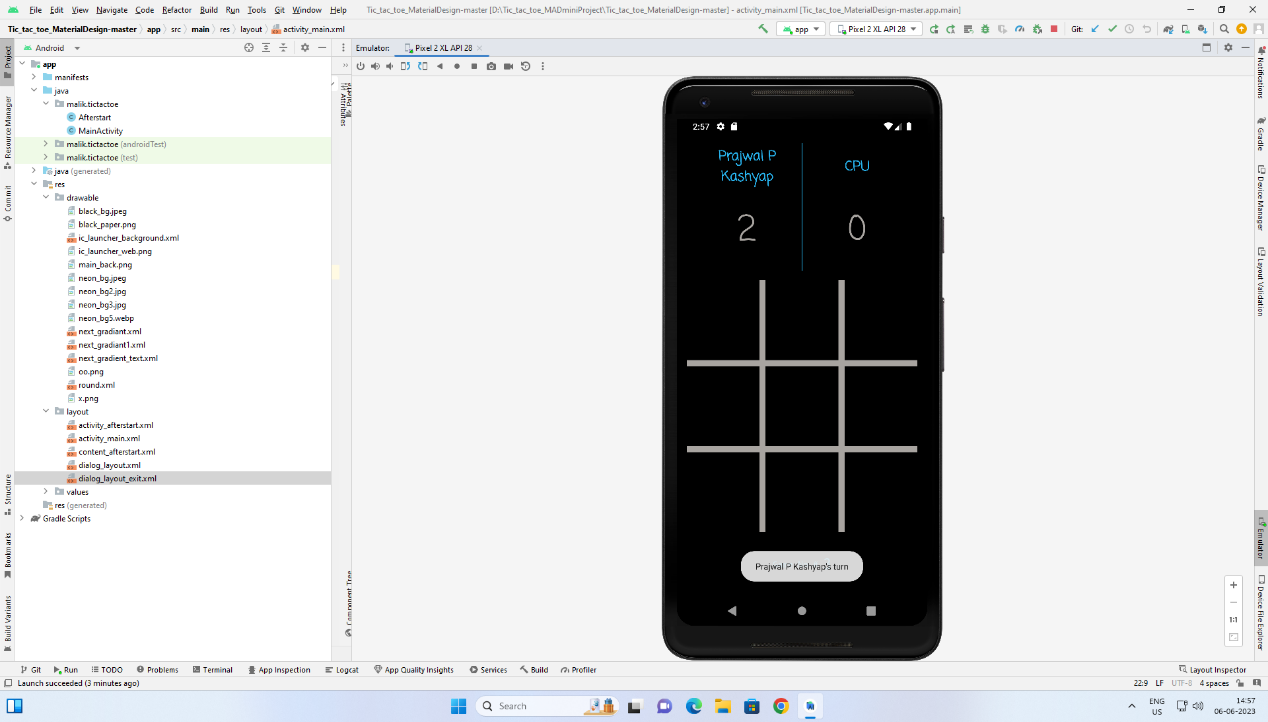
**RESULTS**

**5.1 Sample Screenshots**

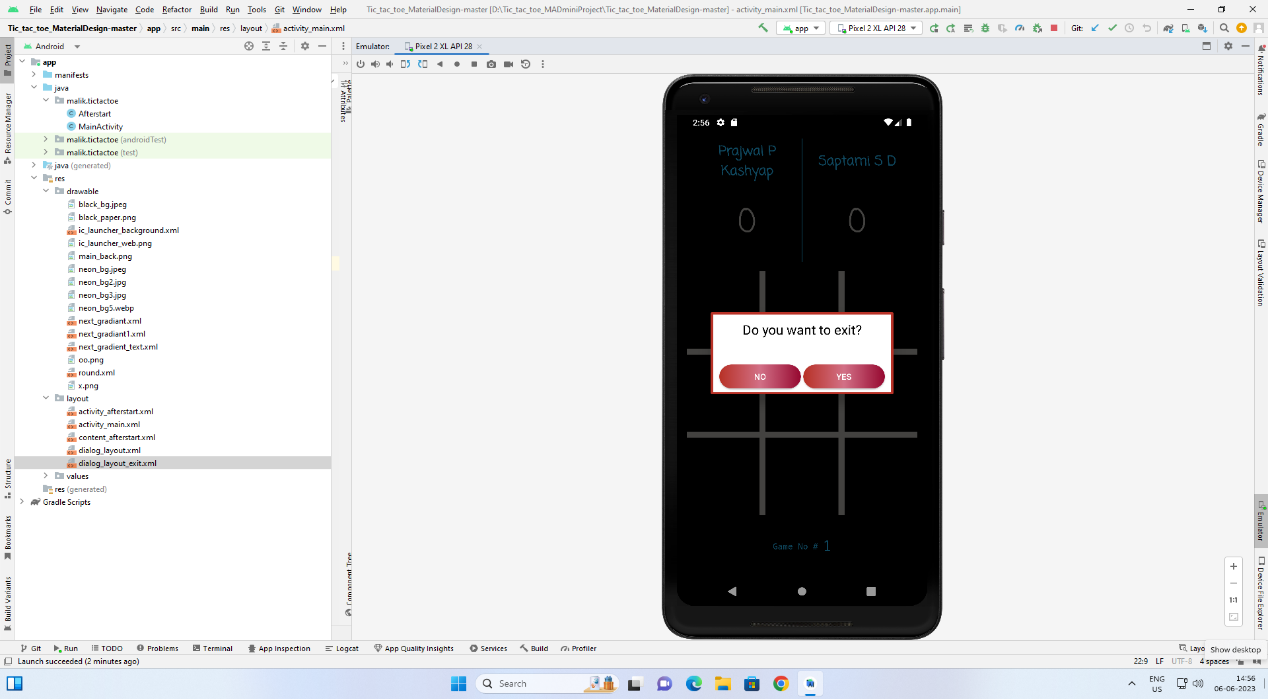
***Fig 5.1 – XML & Application Design***



***Fig 5.2 – Winner Output***

****

***Fig 5.3 – Game against CPU***



***Fig 5.4 – Exit Page***

**CONCLUSION**

Today one cannot afford to rely on the fallible human beings of be really wants to stand against today’s merciless competition where not to wise saying “to err is human” no longer valid, it’s outdated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much-sophisticated hard disk of the computer.

We have done our best to make the complicated process of “TIC-TAC-TOE GAME” as simple as possible using Structured & Modular technique and Menu oriented interface. We have tried to design the application in such a way that user may not have any difficulty in using this package & further expansion is possible without much effort. Even though I cannot claim that this work to be entirely exhaustive. As every game has some limitations so my project is not exceptional, but I will try to sort out them very shortly and deliver a defective less product to client. I am confident that this software package can be readily used by non-programming personal avoiding human handled chance of error.

**BIBLIOGRAPHY**

1. Danial C. Hanson, “Android Application Development and Implementaion – 3D Tic-Tac-Toe”; *3D Tic-Tac-Toe:* pp. 1 - 4, 2010.
2. Lalija Saroja Thota | Manal Elsayeed | Naseema Shaik | Tayf Abdullah Ghawa, “Implementation of Tic-Tac-Toe Game in LabVIEW”: pp. vol 12 numb 2 – June 2014.
3. Computer science With Java - Class XI By : Sumita Arora
4. [www.videolan.org](http://www.videolan.org)
5. [www.wikipedia.com](http://www.wikipedia.com)
6. [www.google.com](http://www.google.com)
7. <www.youtube.com>
8. [www.w3resource.com](www.w3resource.com%20)
9. <https://en.wikipedia.org/wiki/E_(mathematical_constant)>