What is Android?

- **ANDROID** is a Linux-based operating system designed for touch screen mobile devices such as smartphones and tablet PC.

- Now a days **ANDROID** is one of the most popular mobile OS. Developed by Google and later the Open Handset Alliance (OHA).

- **ANDROID** has its own virtual machine i.e. DVM(Dalvik Virtual Machine), which is used for executing the android application.

- **ANDROID** platform is a platform that provides tools and technologies which can used to develop and build mobile Applications.

- **ANDROID** applications are usually developed in the Java language using the Android Software Development Kit.
Versions of Android

- **Alpha** - Android 1.0 (API level- 1)
- **Beta** - Android 1.1 (API level- 2)
- **Cupcake** - Android 1.5 (API level- 3)
- **Donut** - Android 1.6 (API level- 4)
- **Eclairs** - Android 2.0-2.1 (API level- 5 to 7)
- **Froyo** - Android 2.2-2.2.3 (API level- 8)
- **Gingerbread** - Android 2.3-2.3.7 (API level- 9 to 10)
- **Honeycomb** - Android 3.0-3.2.6 (API level- 11 to 13)
- **IceCreamSandwitch** - Android 4.0-4.0.4 (API level- 14 to 15)
- **Jellybean** - Android 4.1-4.3.1 (API level- 16 to 18)
- **KitKat** - Android 4.4-4.4.4 (API level- 19)
- **Lollipop** - Android 5.0-5.1.1 (API level- 21 to 22)
- **Marshmallow** - Android 6.0-6.0.1 (API level- 23)
- **Nougat** - Android 7.0-7.0.1 (API level- 24 to 25)
Types of Android Apps

• **Social** - (WhatsApp, Facebook, LinkedIn, etc…)

• **Entertainment** (Digital Music, Adele, BigTV, Hot Star, etc…)

• **Games** (Grand Theft Auto Theft, Flick Soccer, Temple Run, Candy Crush, etc…)

• **Reading** (BBC News, Pulse News, The Guardian, etc…)

• **LifeStyle**
Android Auto

• Android Auto is a smartphone projection standard developed by Google to allow mobile devices running the Android operating system (version 5.0 "Lollipop" and later) to be operated in automobiles through the dashboard's head unit.

• Android Auto was designed with safety in mind. With a simple and intuitive interface, integrated steering wheel controls, and powerful new voice actions, it's designed to minimize distraction so you can stay focused on the road.

• Android Auto is Google's solution for bringing the power of your smartphone to the relatively dumb nature of vehicle infotainment.

• The standard offers drivers control over GPS mapping/navigation, music playback, SMS, telephony, and web search.
• both touchscreen and button-controlled head unit displays will be supported, although hands-free operation through voice commands is emphasized to ensure safe driving.

• The aim of Android Auto is to extend the functionality of an Android mobile device in an automobile to the dashboard's head unit.

• Android Auto is Google's solution for bringing the power of your smartphone to the relatively dumb nature of vehicle infotainment.

• Basically you're using your phone to broadcast a new user interface onto your car's touchscreen, thus bringing the full power of the latest phones to your car.

• The standard offers drivers control over GPS mapping/navigation, music playback, SMS, telephony, and web search.
What You Need??

- To use Android Auto, you need an Android Auto compatible vehicle or aftermarket radio and an Android phone running 5.0 (Lollipop) or higher.

- When you connect your Android phone to the compatible vehicle or radio, Android Auto will display applications on the vehicle’s screen.

- For more information about Android Auto compatibility, please visit support.google.com/androidauto
\textit{have on our phones even when we drive.}

\textit{but it can be hard to use those apps.}
from your phone during your drive to use

less distracting to use apps and content
phone running android 5.0 lollipop or

during your drive google maps will lead
higher if you're using your phone screen
you USB cable then follow the setup guide
You can start Android Application Development on Operating System like Windows, Mac OS, Linux.

All the required tools to develop Android applications are freely available and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

- Java JDK7 or JDK8
- Java Runtime Environment(JRE)
- Android Studio
Launch *Android Studio.exe*, Make sure before launch Android Studio, Our Machine should required installed Java JDK. To install Java JDK, take a references of [Android environment setup](#).
Once you launched Android Studio, it's time to mention JDK7 path or later version in android studio installer.
Initiate JDK to android SDK
Need to check the components, which are required to create applications, below the image has selected **Android Studio, Android SDK, Android Virtual Machine** and **performance (Intel chip)**.
Need to specify the location of local machine path for Android studio and Android SDK.
Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.
At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.
After done all above steps perfectly, you must get finish button and it going to be open android studio project with Welcome to android studio message as shown below
You can start your application development by calling start a new android studio project. In a new installation frame should ask Application name, package information and location of the project.

![Android Studio project creation frame](image)

Please enter an application name (shown in launcher)
After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK.
The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.
At the final stage it going to be open development tool to write the application code.
Create Android Virtual Device

After Click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking **Create new Virtual device** button.
HelloWorld Example

Before Writing a Hello word code, you must know about XML tags. To write hello word code, you should redirect to `App>res>layout>Activity_main.xml`
String 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 similar types of strings go into this file. This file is responsible for their textual content.

```xml
<resources>

  <string name="app_name">HelloWorld</string>  
  <string name="hello_world">Hello world!</string>  
  <string name="menu_settings">Settings</string>  
  <string name="title_activity_main">MainActivity</string>

</resources>
```
**activity_main.xml** (layout file)

To show hello word, we need to call text view with layout (about text view and layout, you must take references at **Relative Layout** and **Text View**).

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <TextView android:text="@string/hello_world"
        android:layout_width="550dp"
        android:layout_height="wrap_content" />
</RelativeLayout>
```
MainActivity.java

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.

```java
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```
Need to run the program by clicking **Run>Run App** or else need to call **shift+f10** key. Finally, result should be placed at Virtual devices as shown below.
Application Components

- Application components are the essential building blocks of an Android application.

- These components are loosely coupled by the application manifest file `AndroidManifest.xml` that describes each component of the application and how they interact.

- There are following four main components that can be used within an Android application:

  - **Activities**: They dictate the UI and handle the user interaction to the smartphone screen
  - **Services**: They handle background processing associated with an application.
  - **Broadcast Receivers**: They handle communication between Android OS and applications.
  - **Content Providers**: They handle data and database management issues.
Activity

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of `Activity` class as follows:

```java
public class MainActivity extends Activity {
}
```
Other components of Android Application

➢ Resources
➢ Manifest
➢ Layout
➢ Intents
➢ Fragment
➢ Views
Android Resources

**Resources** like static content that your code uses, such as bitmaps, colors, layout definitions, user interface strings, animation instructions, and more. These resources are always maintained separately in various sub-directories under `res/` directory of the project.

You should place each type of resource in a specific subdirectory of your project's `res/` directory.

```
MyProject/
    src/
    MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            activity_main.xml
            info.xml
        values/
            strings.xml
```
The `res/` directory contains all the resources in various subdirectories. Here we have an image resource, two layout resources, and a string resource file.

**anim/** :-  
XML files that define property animations. They are saved in `res/anim/` folder and accessed from the `R.anim` class.

**color/** :-  
XML files that define a state list of colors. They are saved in `res/color/` and accessed from the `R.color` class.

**drawable/** :-  
Image files like .png, .jpg, .gif or XML files that are compiled into bitmaps, state lists, shapes, animation drawables. They are saved in `res/drawable/` and accessed from the `R.drawable` class.

**layout/** :-  
XML files that define a user interface layout. They are saved in `res/layout/` and accessed from the `R.layout` class.

**menu/** :-  
XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. They are saved in `res/menu/` and accessed from the `R.menu` class.
values/ :-

XML files that contain simple values, such as strings, integers, and colors. For example, here are some filename conventions for resources you can create in this directory:

✓ arrays.xml for resource arrays, and accessed from the R.array class.
✓ integers.xml for resource integers, and accessed from the R.integer class.
✓ bools.xml for resource boolean, and accessed from the R.bool class.
✓ colors.xml for color values, and accessed from the R.color class.
✓ dimens.xml for dimension values, and accessed from the R.dimen class.
✓ strings.xml for string values, and accessed from the R.string class.
✓ styles.xml for styles, and accessed from the R.style class.

xml/ :-

Arbitrary XML files that can be read at runtime by calling Resources.getXML(). You can save various configuration files here which will be used at run time.
Manifest

Whatever component you develop as a part of your application, you must declare all its components in a manifest file called `AndroidManifest.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.
Android Example 1
Step 6: -

activity_main.xml is layout file available in res/layout directory, that is referenced by application while building interface.

```xml
<LinearLayout
    android:layout_margin="10dp"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/swami"
        android:adjustViewBounds="false"/>

    <TextView
        android:layout_margin="10dp"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Swami Jnanananda"
        android:textAppearance="?android:textAppearanceLarge"/>

    <TextView
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Indian Yogi and Nuclear Physicist"
        android:textColor="@android:color/background_dark"/>

    <Button
        android:id="@+id/submit"
        android:layout_margin="10dp"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"/>

</LinearLayout>
```

To add Imageview in activity add image to res/drawable directory
Step 7: -

To perform Action intent to the activity create another Activity, While we click on “Submit” button then it should goes to next screen

src/main/java >> right click on ‘package name’ >> New>> Activity>> (Choose any Activity)
For Ex. ‘Empty Activity’
Step 8: -
Write a code to give action to the Button in MainActivity.java

```java
package in.ennovator.manodemanooore;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Manode extends AppCompatActivity {
    Button submit;

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

        submit = (Button) findViewById(R.id.submit);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Manode.this, MainActivity.class));
            }
        });
    }
}
```

By `onCreate()` method we link up the xml file of Activity

`onClickListener()` method provide the action to button

`startActivity()` provides intent to the next Screen
Step 8: 
Now the App is ready to Run. Connect your Android Device and click on “Run” ( ) then setup Run Configuration. Select Your Device from “Select Deployment Target” dialog box and click “Ok”
Android Example 2
To add images to Android project, we have to add image first to **app/src/main/res/drawable** folder.

Copy the image from computer and paste it to **app/src/main/res/drawable** folder.

In Android studio drag option ImageView and select image from drawable.
Android Example 3
Step 1 :-

Open Android Studio and Create new application. In app/res/layout/activity_main.xml, This activity is for entering name, age and select male/female you should create Textiew, EditText to enter name and age, RadioButton to select option male/female.

Create one submit Button after clicking on that it should goes to next screen and shows the content which are entered by users.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>

<TextView
android:id="@+id/nameText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
android:layout_marginLeft="18dp"
android:layout_marginTop="24dp"
android:text="@string/name" />
```
<TextView
android:id="@+id/genderText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/gridLayout1"
android:layout_below="@+id/age"
android:layout_marginTop="19dp"
android:text="@string/gender" />

<GridLayout
android:id="@+id/gridLayout1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/ageText"
android:layout_below="@+id/age"
android:layout_marginTop="44dp" >
<RadioGroup
android:id="@+id/gender"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<RadioButton
android:id="@+id/femaleRadio"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:checked="true"
android:text="@string/female" />

<RadioButton
android:id="@+id/maleRadio"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:checked="true"
android:text="@string/male" />
</RadioGroup>

<GridLayout>
<Button
android:id="@+id/submit"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignRight="@+id/gridLayout1"
android:layout_below="@+id/gridLayout1"
android:layout_marginTop="21dp"
android:text="@string/submit" />

</RelativeLayout>
Step 2: Create second Activity.

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:columnCount="3">
  <TextView
      android:id="@+id/nameText"
      android:layout_column="1"
      android:layout_gravity="left"
      android:layout_row="1"
      android:text="@string/name" />
</GridLayout>
```
<TextView
android:id="@+id/nameValue"
android:layout_column="2"
android:layout_gravity="left"
android:layout_row="1"
android:text=""
android:textAppearance="?android:attr/textAppearanceMedium" />

<TextView
android:id="@+id/ageText"
android:layout_column="1"
android:layout_gravity="left"
android:layout_row="3"
android:text="@string/age" />

<TextView
android:id="@+id/ageValue"
android:layout_column="2"
android:layout_gravity="left"
android:layout_row="3"
android:text=""
android:textAppearance="?android:attr/textAppearanceMedium" />
<TextView
    android:id="@+id/genderText"
    android:layout_column="1"
    android:layout_gravity="left"
    android:layout_row="5"
    android:text="@string/gender" />

<TextView
    android:id="@+id/genderValue"
    android:layout_column="2"
    android:layout_gravity="left"
    android:layout_row="5"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium" />
<Space
android:layout_width="1dp"
android:layout_height="10dp"
android:layout_column="0"
android:layout_gravity="fill_horizontal"
android:layout_row="4" />

</GridLayout>
Step 3 :-

In MainActivity.java

• In onCreate() method, Initialize TextView, EditText, RadioButton and Button.

```java
button = (Button) findViewById(R.id.submit);
button.setOnClickListener(this);
name = (EditText) findViewById(R.id.name);
age = (EditText) findViewById(R.id.age);
genderRadioGroup = (RadioGroup) findViewById(R.id.gender);
```
To give action to the button means while clicking it should go to next Activity and should display name, age and gender.

```java
@Override
public void onClick(View v) {
    Intent intent = new Intent(getApplicationContext(),
        DisplayActivity.class);

    // Create a bundle object
    Bundle b = new Bundle();

    // Inserts a String value into the mapping of this Bundle
    b.putString("name", name.getText().toString());
    b.putString("age", age.getText().toString());
    int id = genderRadioGroup.getCheckedRadioButtonId();
    RadioButton radioButton = (RadioButton) findViewById(id);
    b.putString("gender", radioButton.getText().toString());

    // Add the bundle to the intent.
    intent.putExtras(b);

    // Start the DisplayActivity
    startActivity(intent);
}
```
Step 4 :-

In Display.java, you should Initialize all three TextViews

```java
Bundle b = getIntent().getExtras();
TextView name = (TextView) findViewById(R.id.nameValue);
TextView age = (TextView) findViewById(R.id.ageValue);
TextView gender = (TextView) findViewById(R.id.genderValue);

name.setText(b.getCharSequence("name"));
age.setText(b.getCharSequence("age"));
gender.setText(b.getCharSequence("gender"));
```
In previous example we added data(name, age, gender) to the next screen, but it’s used for only one single user and only to display the entered data. It is not storing the data. To store the entered data of many users we need database.

Database is the application where we should store data like text, images etc. and retrieve it to our Android application.

Here we are using Firebase database which is very easy to integrate in android studio.

Firebase is the cloud database provide by Google. To work with firebase we need google account.
Step 1 :-

Create new Project in Android Studio, to integrate Firebase to Android project Open [www.firebase.com](http://www.firebase.com) and Login to Legacy Console with google account.
✓ Now Add some libraries to build.gradle(app) to store and retrieve data from the database.

```gradle
compile 'com.google.firebase:firebase-database:10.0.1'
compile 'com.google.firebase:firebase-core:10.0.1'
```

✓ After adding Sync project and let complete the process of gradle building.

✓ In `activity_main.xml`

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="in.ennovator.hello.MainActivity">
```
<Button
    android:text="Submit"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="63dp"
    android:id="@+id/submit"
    android:textColor="@color/colorAccent"
    android:background="@android:color/holo_green_dark"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
In `MainActivity.java`

```java
package in.ennovator.hello;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText name;
    private EditText designation;
    private EditText location;
    private Button submit;
    private DatabaseReference databaseReference;
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    databaseReference = FirebaseDatabase.getInstance().getReference();
    name = (EditText) findViewById(R.id.name);
    designation = (EditText) findViewById(R.id.designation);
    location = (EditText) findViewById(R.id.location);
    submit = (Button) findViewById(R.id.submit);
    submit.setOnClickListener(this);
}

@Override
public void onClick(View view) {
    final String mName = name.getText().toString().trim();
    final String mDesignation = designation.getText().toString().trim();
    final String mLocation = location.getText().toString().trim();
if (!TextUtils.isEmpty(mName) && !TextUtils.isEmpty(mDesignation) && !TextUtils.isEmpty(mLocation)) {

    DatabaseReference data = databaseReference.push();

    data.child("Name").setValue(mName);
    data.child("Designation").setValue(mDesignation);
    data.child("Location").setValue(mLocation);

    Toast.makeText(getApplicationContext(), "Registered Successfully", Toast.LENGTH_LONG).show();
}
}
Website links to learn

- www.androidstudiotutorials.com
- www.stackoverflow.com
- www.Github.com
- www.androidhive.com
- www.android.arsenal.com
- www.developer.android.com
Thank You