



Android

MAD Syllabus

Introduction: About Android, Pre-requisites to learn Android, Dalvik Virtual Machine & .apk file extension, Android API levels (versions & version names)

Android Java Basics: Getting started with Android development, project folder structure, simple programming, running project, generating build/APK of the app from Android Studio

First application: Creating Android Project, Android Virtual Device Creation, Set up debugging environment, Workspace set up for development, Launching emulator, debugging on mobile devices.

Basic UI design: Basics about Views, Layouts, Drawable Resources, Input controls, Input Events, Toasts.

More UI Components: Layouts - GridView and ListView, Action bar, Adapters, Menus: Option menu, context menu, sub menu, Pickers - Date and Time, Spinners.

Activity and Fragment: Activity, Fragment, Activity Lifecycle and Fragment Lifecycle.

Objectives

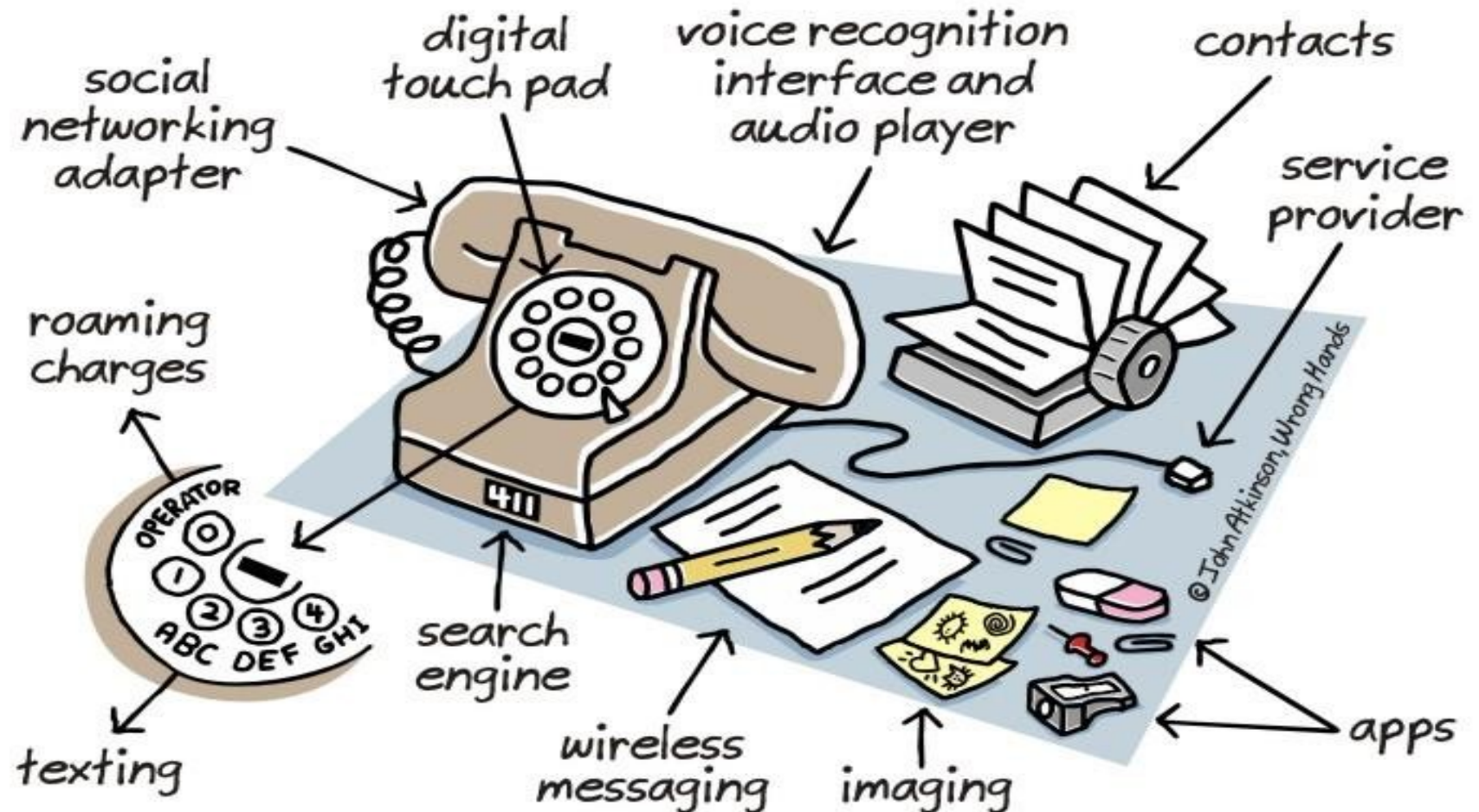


- **What is Android?**
 - **Definition**
 - **Architecture**
- **Why Android**
 - **Features of Android**
- **Installation of Android**
- **Different Android Versions**
- **Android Activity Lifecycle**
- **Android Project Structure**
- **My First Program-Addition of Two Numbers**
- **Google Map Navigation**

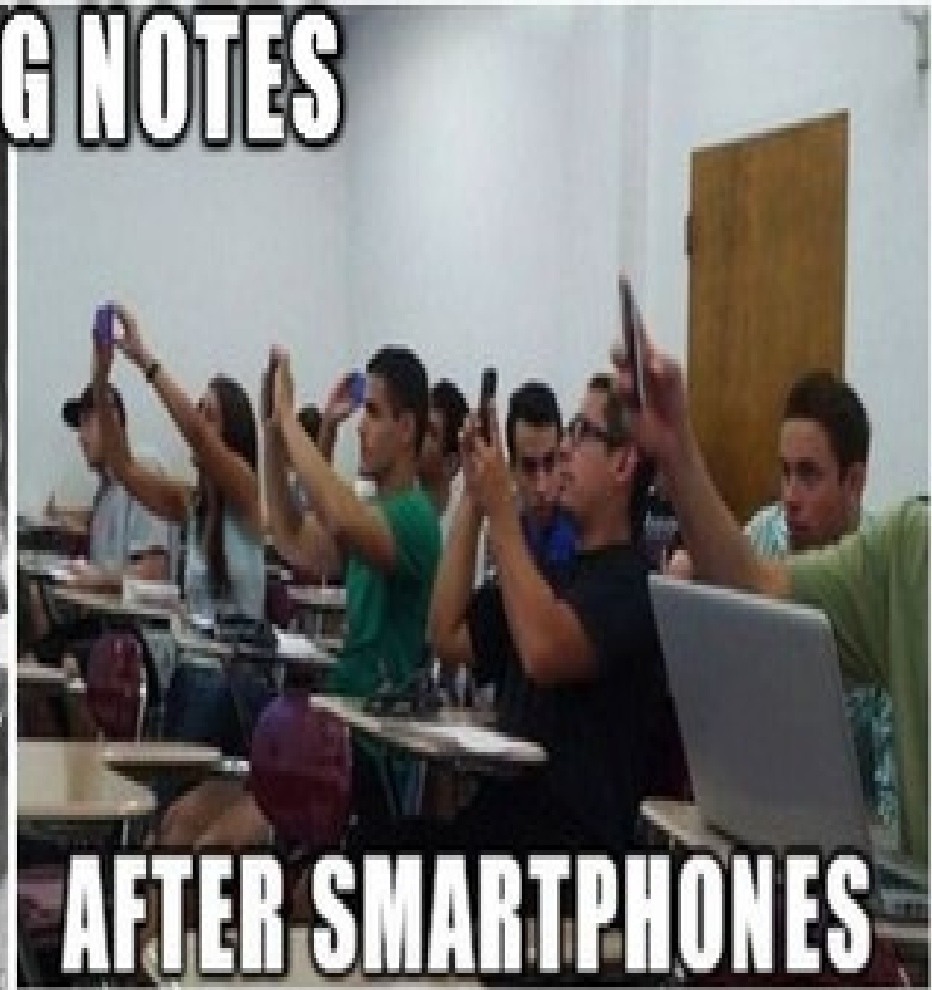
Before SmartPhone



vintage smartphone



Before SmartPhone





The Players

- **Android** – Open source mobile OS developed by the **Open Handset Alliance(2007)** led by Google. Based on Linux 2.6 kernel. The open handset alliance(OHA) is a business alliance of 84 firm to develop open standard for mobile devices. Member firms include HTC, Sony, Dell, Intel, Motorola, Google, Samsung Electronics, LG Electronics, **T-Mobile G1(First Android Phone)** .
- **iOS** – **Apple's** proprietary mobile OS, iPhone, iPod Touch, iPad. Derived from OS X, very UNIX like
- **Symbian** – acquired by **Nokia** 2008
- **Windows Phone 7** – **Microsoft** – Kin, discontinued 6 weeks after initial launch
- **Blackberry OS** – **RIM** (Research in Motion), proprietary OS

Mobile Operating Systems

❖ Android	Google	11	.apk	(70.92%)
❖ iOS	Apple	4.3	.app	(26.53%)
❖ Tizen	Samsung n Intel	4.0.0.7	.tpk	(0.22%)

- Symbian(.sis) By Nokia
- RIM's BlackBerry(.cod)
- Windows mobile(.cab) by Microsoft
- J2ME(.jar)



Android?

- **What?**
- **Why?**
- **How?**

What is Android?



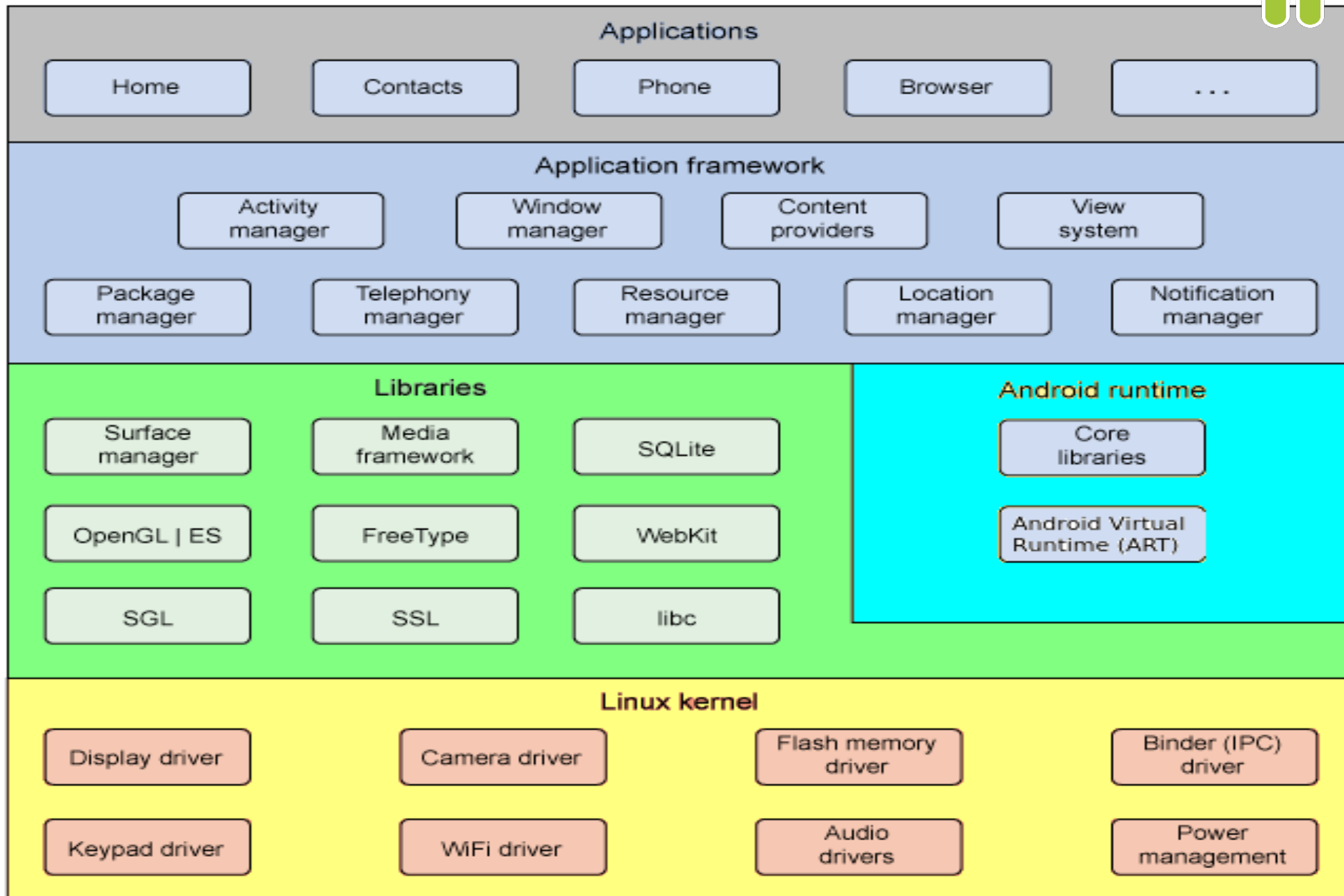
- Android is an **open source operating system**, created by Google specifically for **use on mobile devices** (cell phones and tablets)
- **Linux based (2.6 kernel)**
- Supports **Bluetooth, Wi-Fi, and 3G and 4G networking**



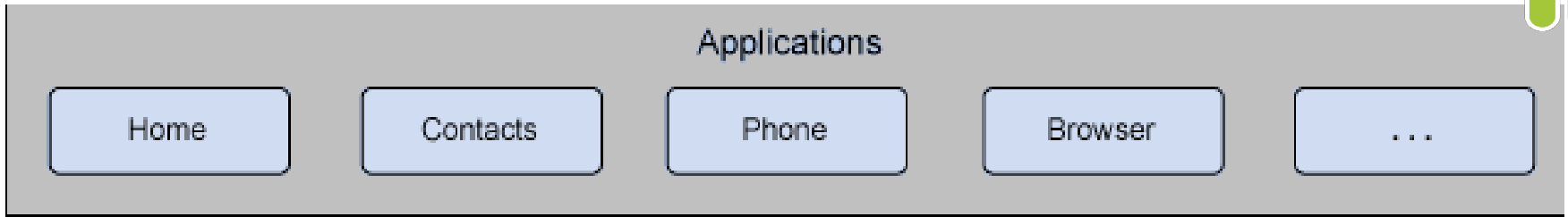
What is Android?

- Initially founded by **Andy Rubin** in October 2003 and later acquired by Google on August 17, 2005. **Android** is a free Linux based platform and is
- **An open software stack with an operating system**
- **Applications**
- **Middleware (As middle-ware is the interface between the applications and operating system.)**

Android Platform Architecture

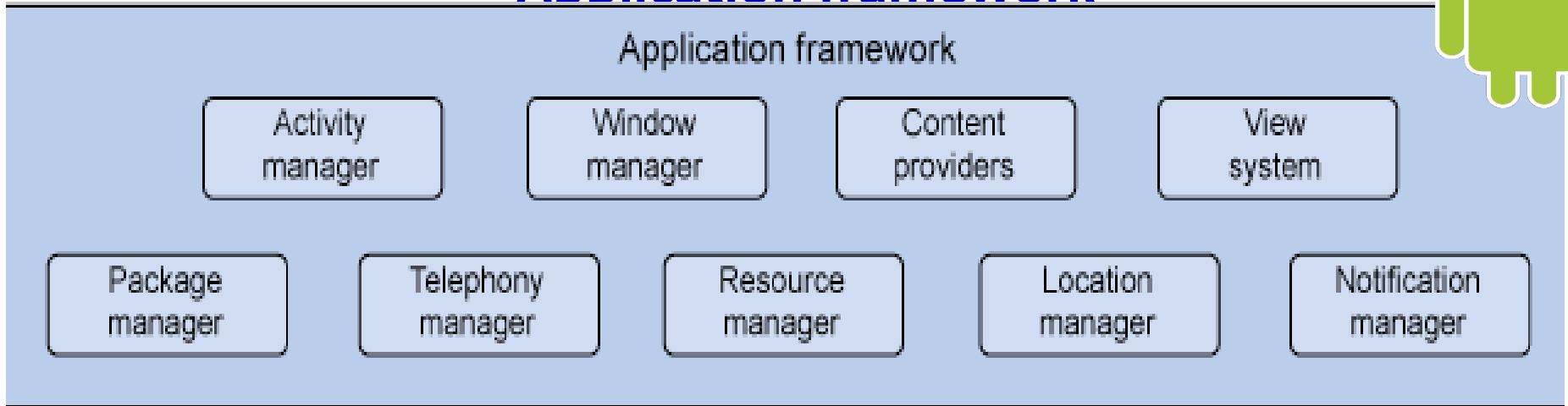


Android applications



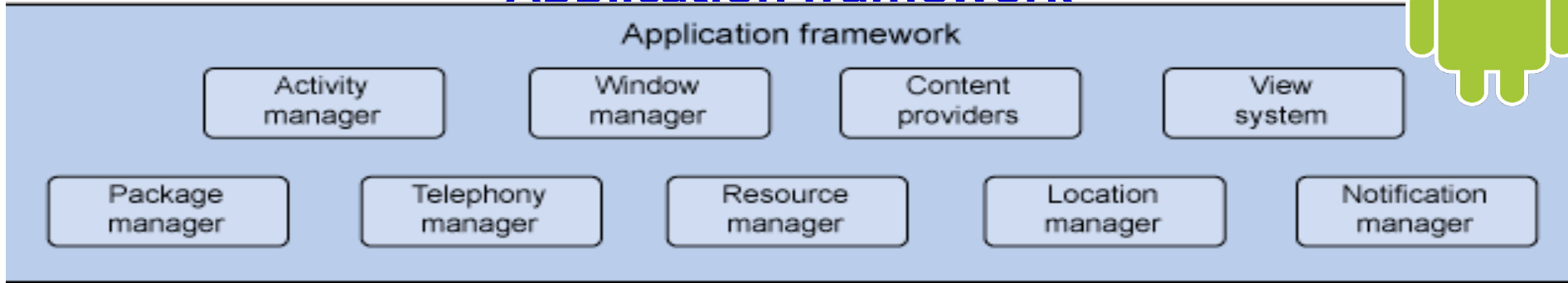
- This is the topmost layer in the Android platform stack and is comprised of applications that are **built-in (developed by the Android team) or any other third party applications** that have been installed on the device.
- Applications that you develop are also installed in this layer. Typical applications include: **Camera, Alarm, Clock, Calculator, Contacts, Calendar, Media Player, and so forth.**

Application framework



- The Applications Framework Layer, sometimes referred to as the Java API Framework, provides a set of **higher-level Java classes** which you can make use of when developing an Android app.
- **Package Manager:** An API which **handles installing, uninstalling and updating of Android applications.**

Application framework



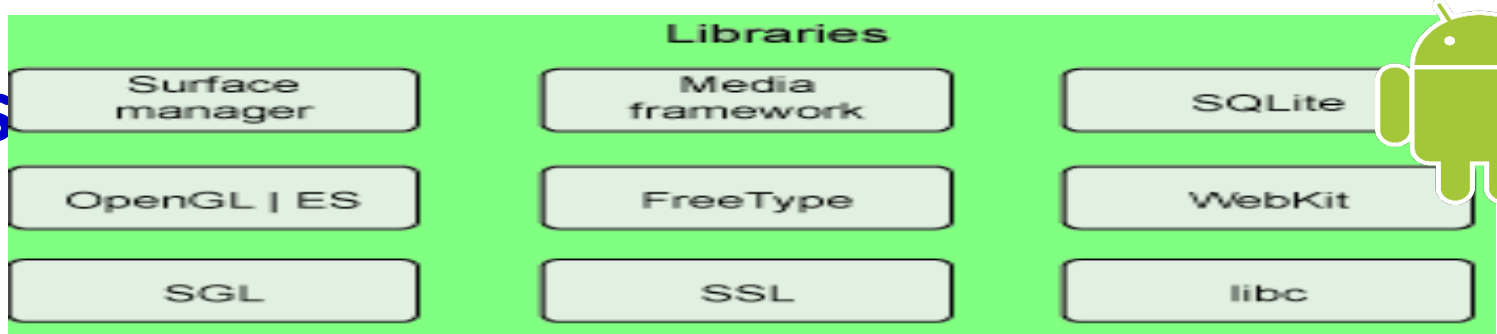
- **Telephony Manager:** The Telephony Manager is **responsible for voice calls**. `TelephonyManager` class provides information about the telephony **services such as subscriber id, sim serial number, phone network type** etc. Moreover, you can determine the phone state etc.
- **View System:** Contains **User Interface building blocks** used to build an application's UI, including lists, grids, texts, boxes, buttons, etc.
- **Window Manager:** **Responsible for keeping track of the order of the windows**, which windows are visible, and how they are laid out on the UI.

Libraries (Middleware)



- On the top of linux kernel, there are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.
- The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

Libraries



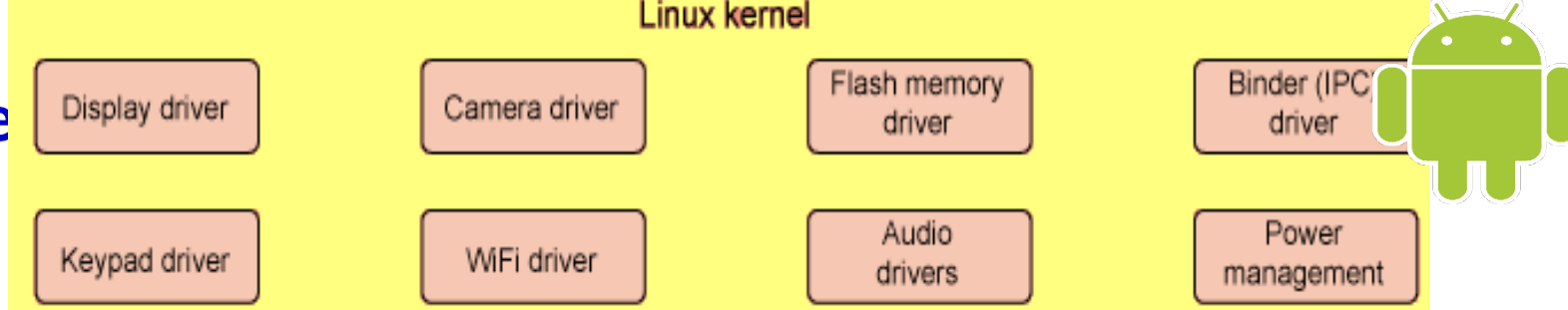
- Contains C/C++ libraries used by components of Android Systems.
- Few features include,
 - **SQLite Library used for data storage** and light in terms of mobile memory footprints and task execution.
 - **WebKit Library mainly provides Web Browsing engine** and a lot more related features.
 - **The surface manager library is responsible for rendering windows** and drawing surfaces of various apps on the screen.
 - **The media framework library provides media codecs for audio and video.**
 - The **OpenGL (Open Graphics Library) and SGL(Scalable Graphics Library)** are the graphics libraries for 3D and 2D rendering, respectively.
 - The **FreeType Library is used for rendering fonts.**

Android Runtime (Middleware)



- Designed to **run apps in a constrained environment** that has limited muscle power **in terms of battery, processing and memory.**
- Contains set of core libraries that enables developers to write Android Apps using Java Programming.

Linux Kernel

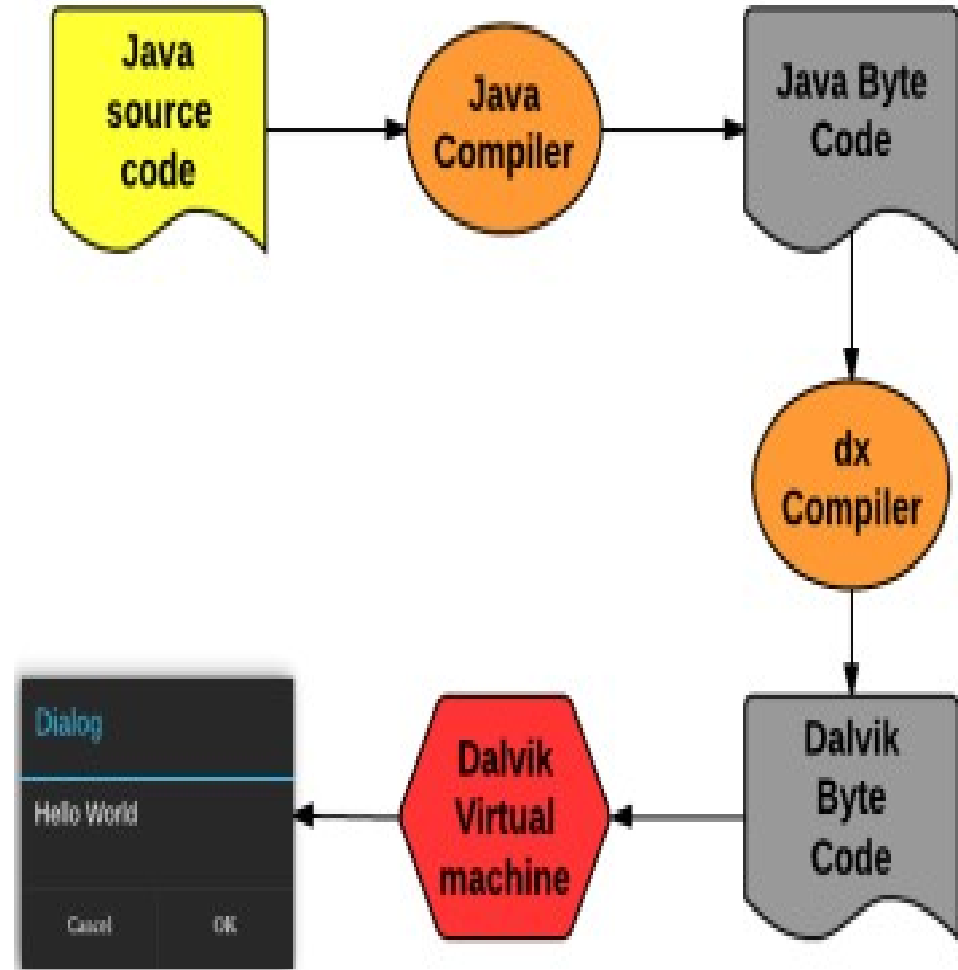


- The Android platform is **built on top of the Linux 2.6** Kernel with a few architectural changes. Note that the term *kernel* refers to the core of any operating system.
- The Linux Kernel provides **support for memory management, security management, network stack, process management, and device management.**
- The **Linux Kernel contains a list of device drivers** that facilitate the communication of an Android device with other peripheral devices

Dalvik Virtual Machine | DVM



- The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. **It optimizes the virtual machine for memory, battery life and performance.**
- The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. **Multiple class files are converted into one dex file.**





"How many calories do you burn by downloading diet apps?"



Why Android?

- **Android is free and Open source:**

- Android, since the day it was launched, has been available **free of cost** and Google made it clear that it will be free in future as well.
- With **many brains working on the system**, newer and newer ideas were incorporated, which in turn helped in making Android a preferred choice.

- **Open for customisation:**

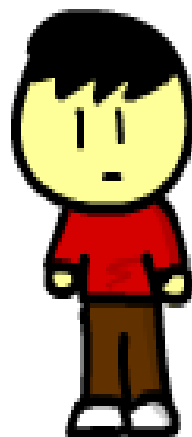
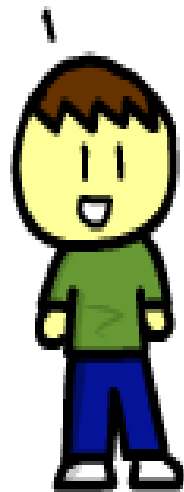
- Unlike Windows or any other mobile operating system, **device manufacturers are free to modify** Android as per their needs.



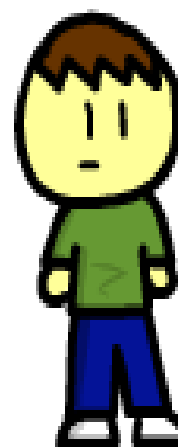
Why Android?

- **Large number of applications:**
 - Have a wide variety of applications to choose from and can customize their phones for a personal experience.
- **Language support:**
 - Even people who are unfamiliar with English use smartphones. **Android supports its interface in 70 languages.**
- **Support for Devices:**
 - **TV, Watches, Mobiles, Glass, Cars, Tabs**

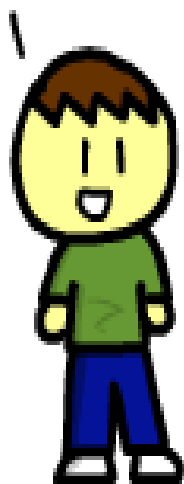
HEY MAN! HAPPY BIRTHDAY!



OH! GEE THANKS. HOW DID YOU KNOW?
I DIDNT TELL ANYBODY



OH...HAHA THATS JUST HOW GOOD OF A FRIEND I AM



facebook

MAKING YOU A BETTER FRIEND SINCE 2004



How to Install and Setup Android Studio

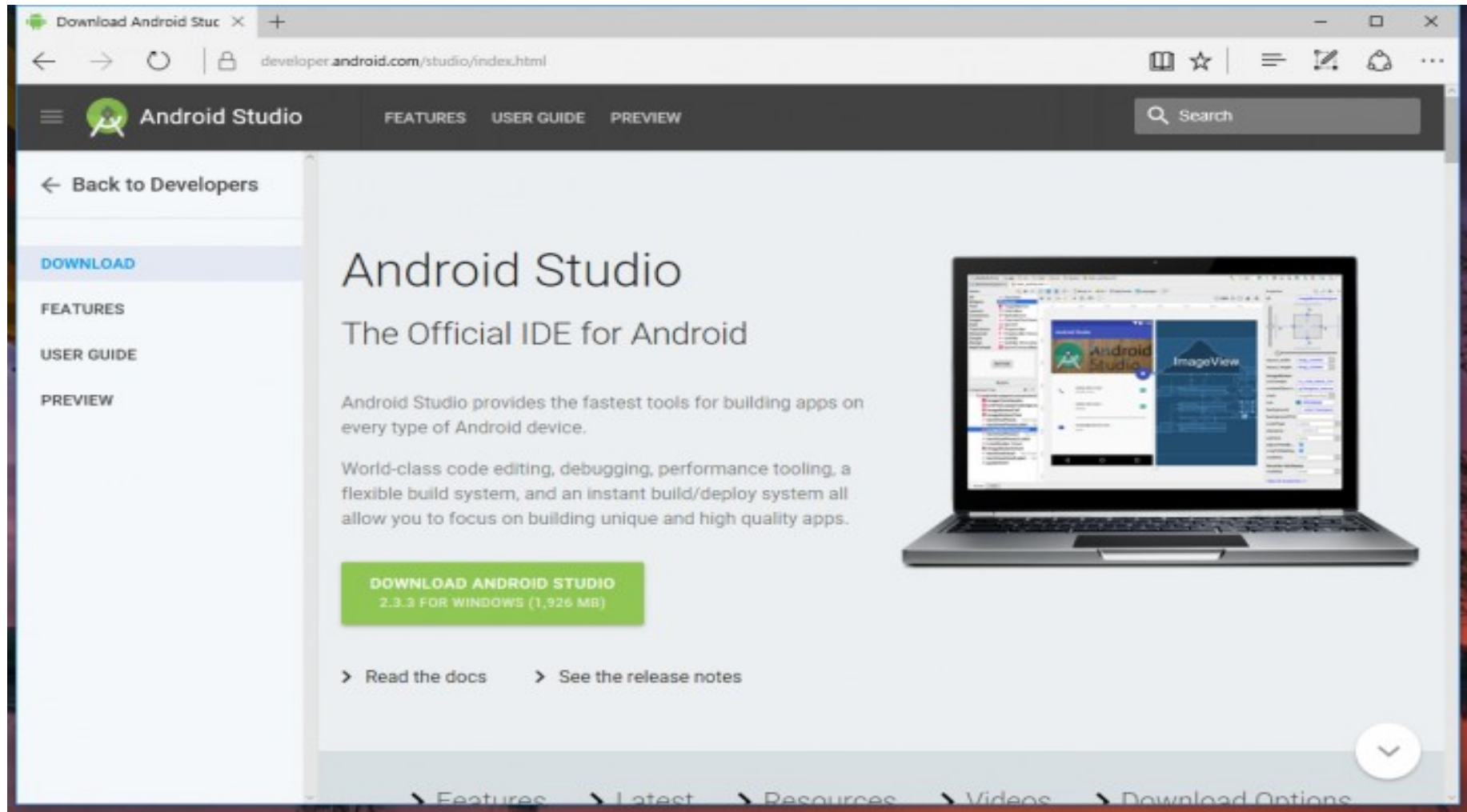
- **Step 1** - [Download Java Platform \(JDK\)](https://www.oracle.com/technetwork/java/javase/downloads/index.html) and install it on your computer. Once completed, then proceed to the next step. Java is important for the functioning of Android Studio.

The screenshot shows the Oracle Java SE Downloads page in a web browser. The browser's address bar displays the URL [oracle.com/technetwork/java/javase/downloads/index.html](https://www.oracle.com/technetwork/java/javase/downloads/index.html). The page features the Oracle logo and a navigation menu. A sidebar on the left lists various Java products, including Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The main content area is titled "Java SE Downloads" and includes tabs for Overview, Downloads, Documentation, Community, Technologies, and Training. It displays two download options: "Java Platform (JDK) 8u131" and "NetBeans with JDK 8", each with a "DOWNLOAD" button. Below these, there is a section for "Java Platform, Standard Edition" detailing the "Java SE 8u131" release, which includes important security fixes and bug fixes. A highlighted box contains an "Important planned change for MD5-signed JARs", stating that starting with the April Critical Patch Update releases, all JRE versions will treat JARs signed with MD5 as unsigned. At the bottom, there are links for "Installation Instructions", "Release Notes", "Oracle License", "Java SE Products", and "Third Party Licenses", along with "JDK" and "Server JRE" download buttons.



How to Install and Setup Android Studio

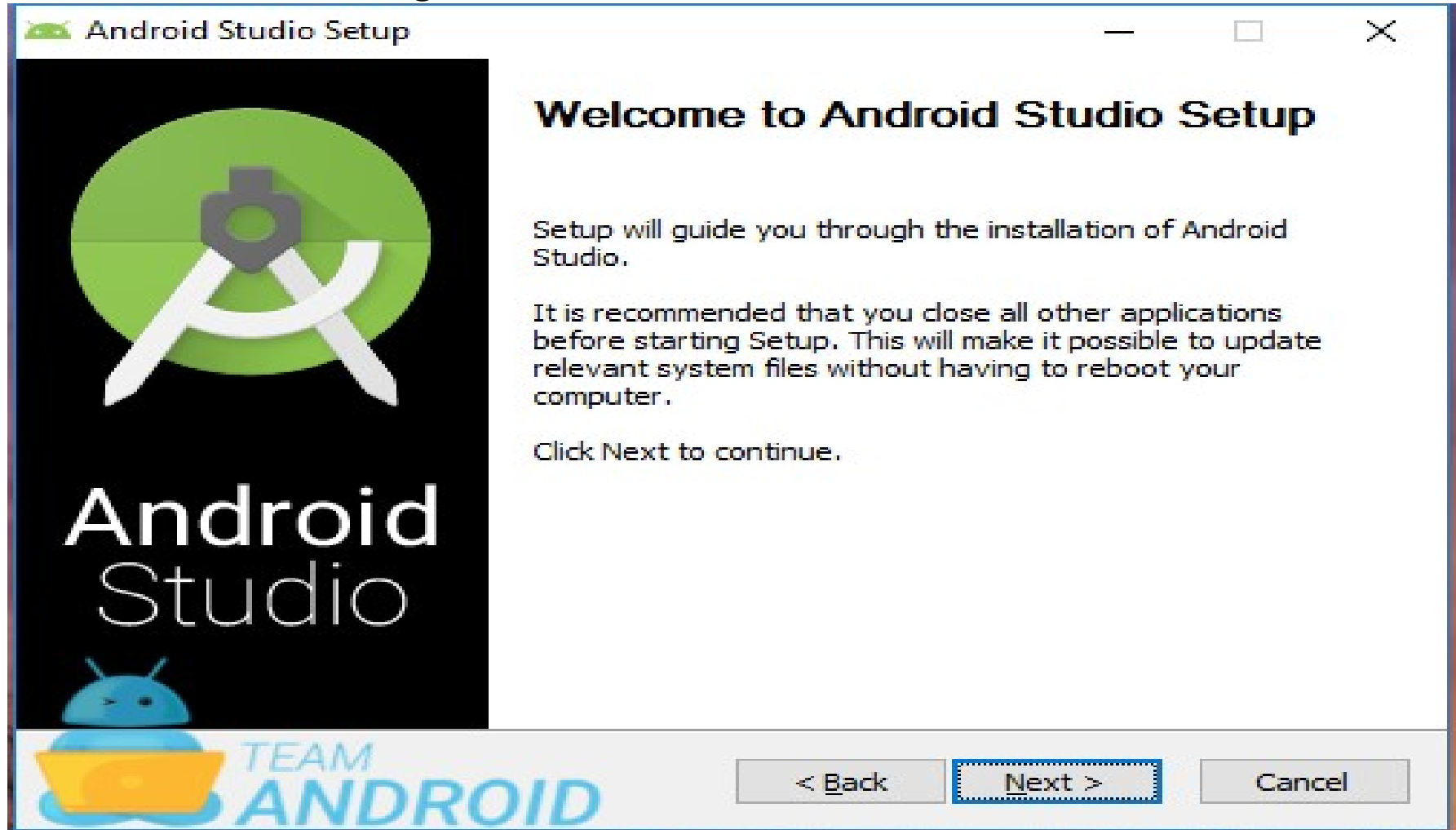
- **Step 2** - [Download Android Studio](https://developer.android.com/studio/index.html) (android-studio-bundle-XX.XXXXXX-windows.exe) from the Android Developers website. This may take a while to download as the entire set up is about 2GB in size.





How to Install and Setup Android Studio

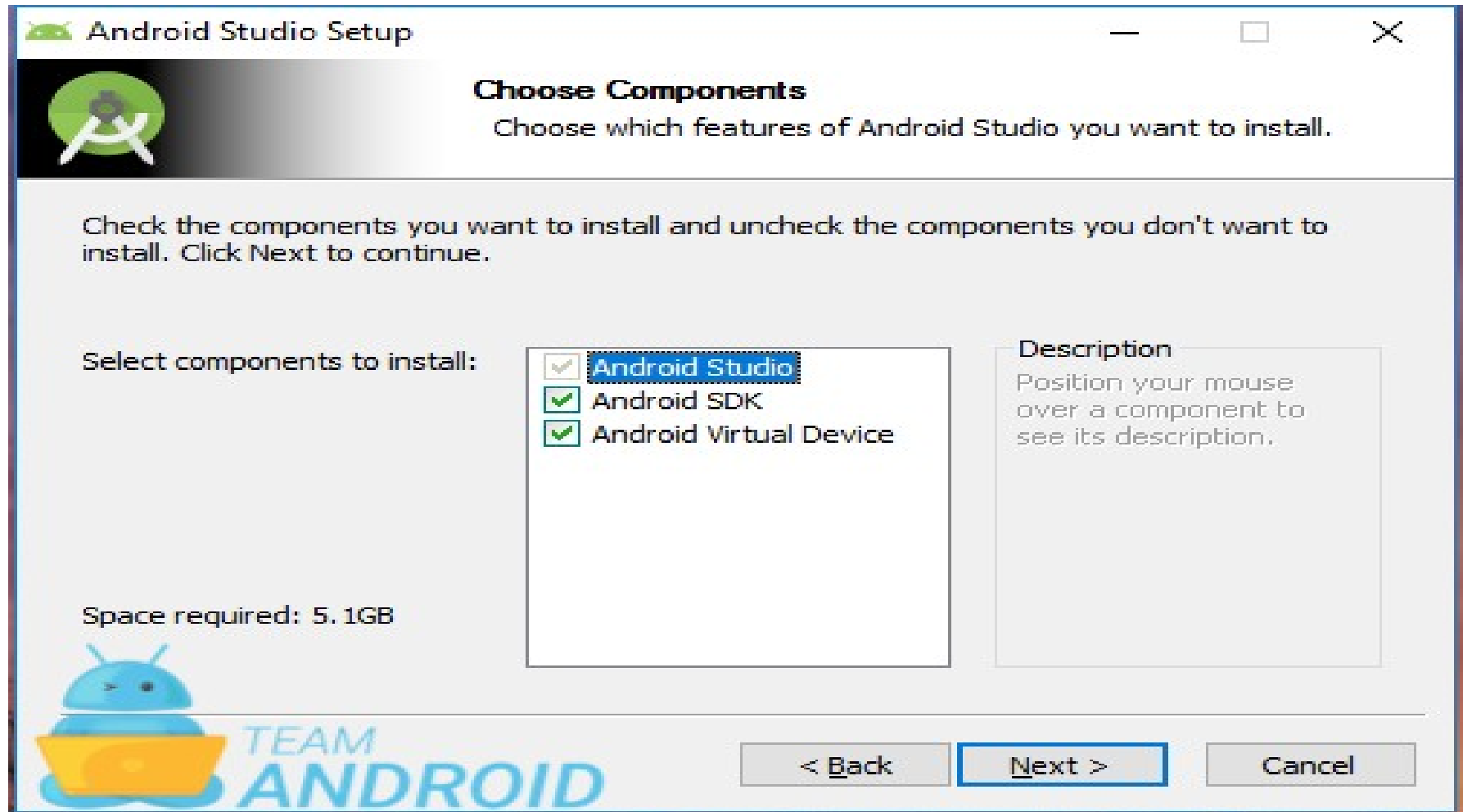
- **Step 3** - Run the EXE setup file you just downloaded. You should be greeted with a similar setup wizard screen as shown below. Click *Next* to begin!





How to Install and Setup Android Studio

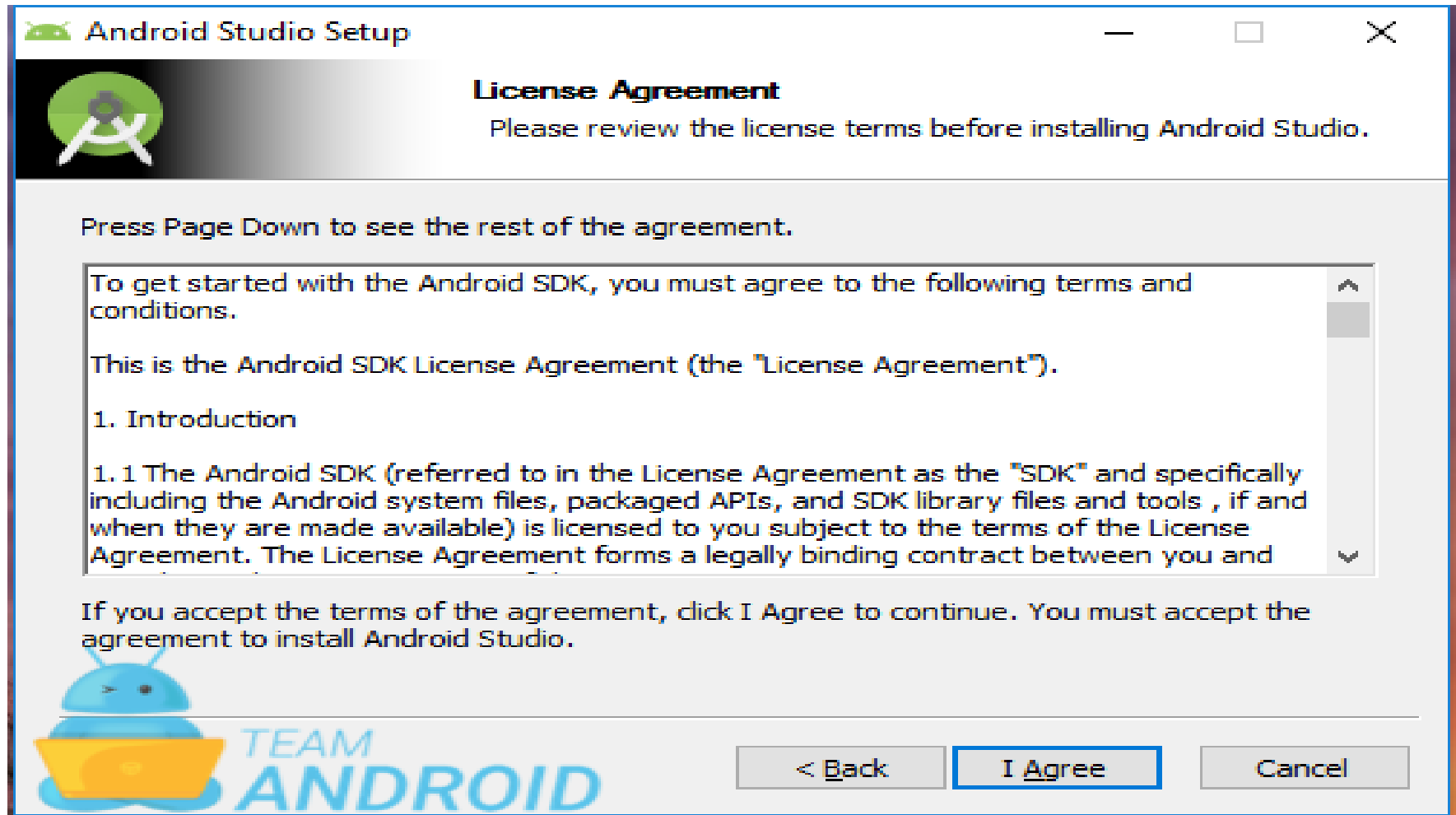
- **Step 4** – Keep the default components selected for installation. Click *Next*.





How to Install and Setup Android Studio

- **Step 5** – We are sure you would not want to read through the entire license agreement. Click *I Agree*.





How to Install and Setup Android Studio

- **Step 6** - This is where you select the installation location for Android Studio and Android SDK. You may select another location / drive that has the required space available. Click *Next* to continue.

Android Studio Setup

Configuration Settings
Install Locations

Android Studio Installation Location

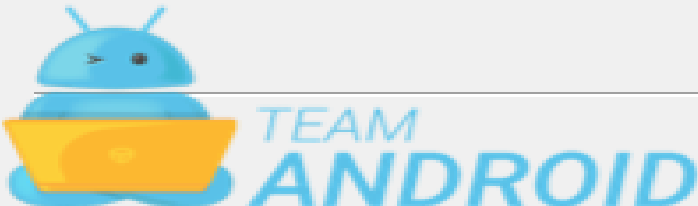
The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

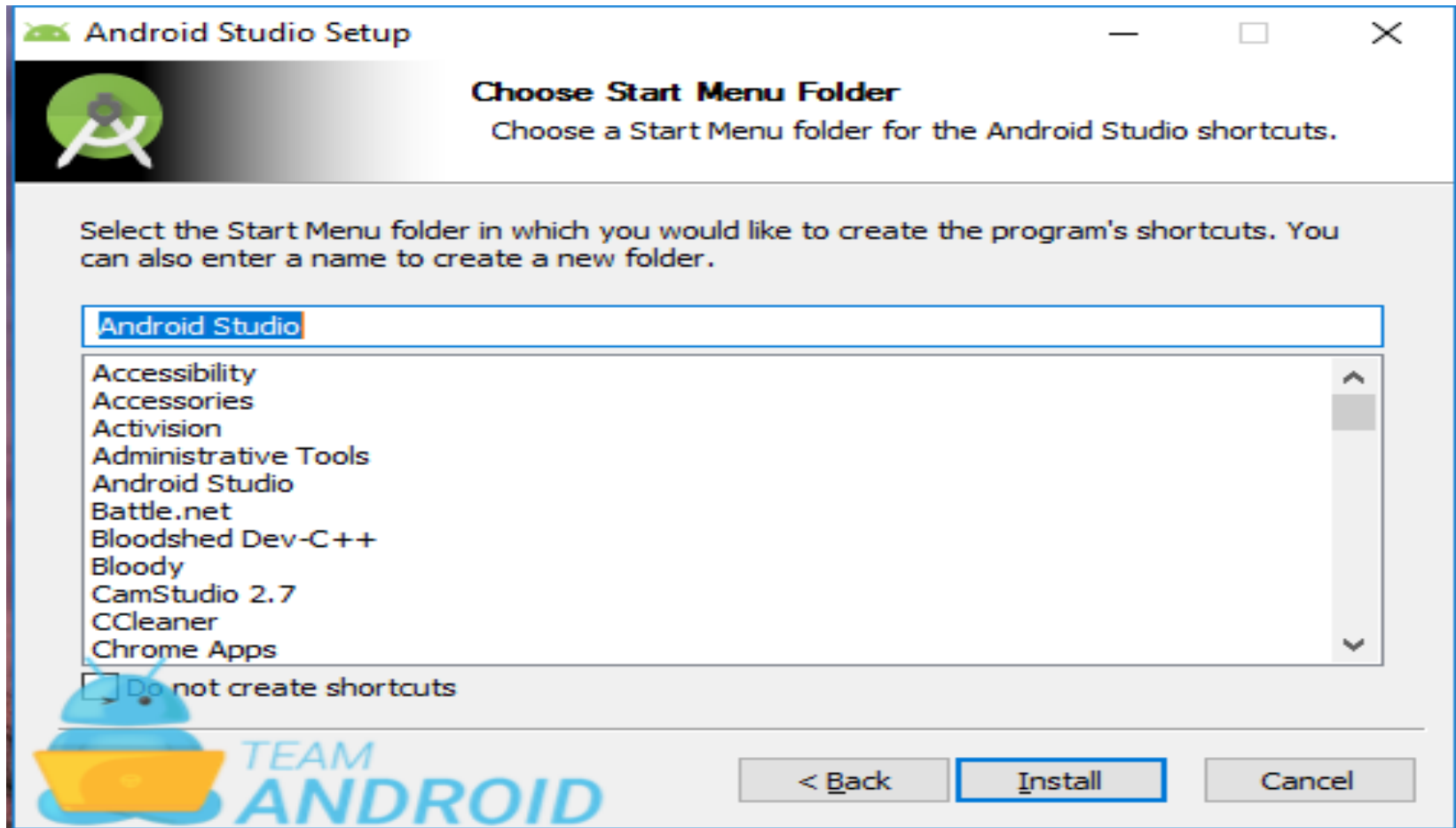
C:\Users\Haris\AppData\Local\Android\sdk





How to Install and Setup Android Studio

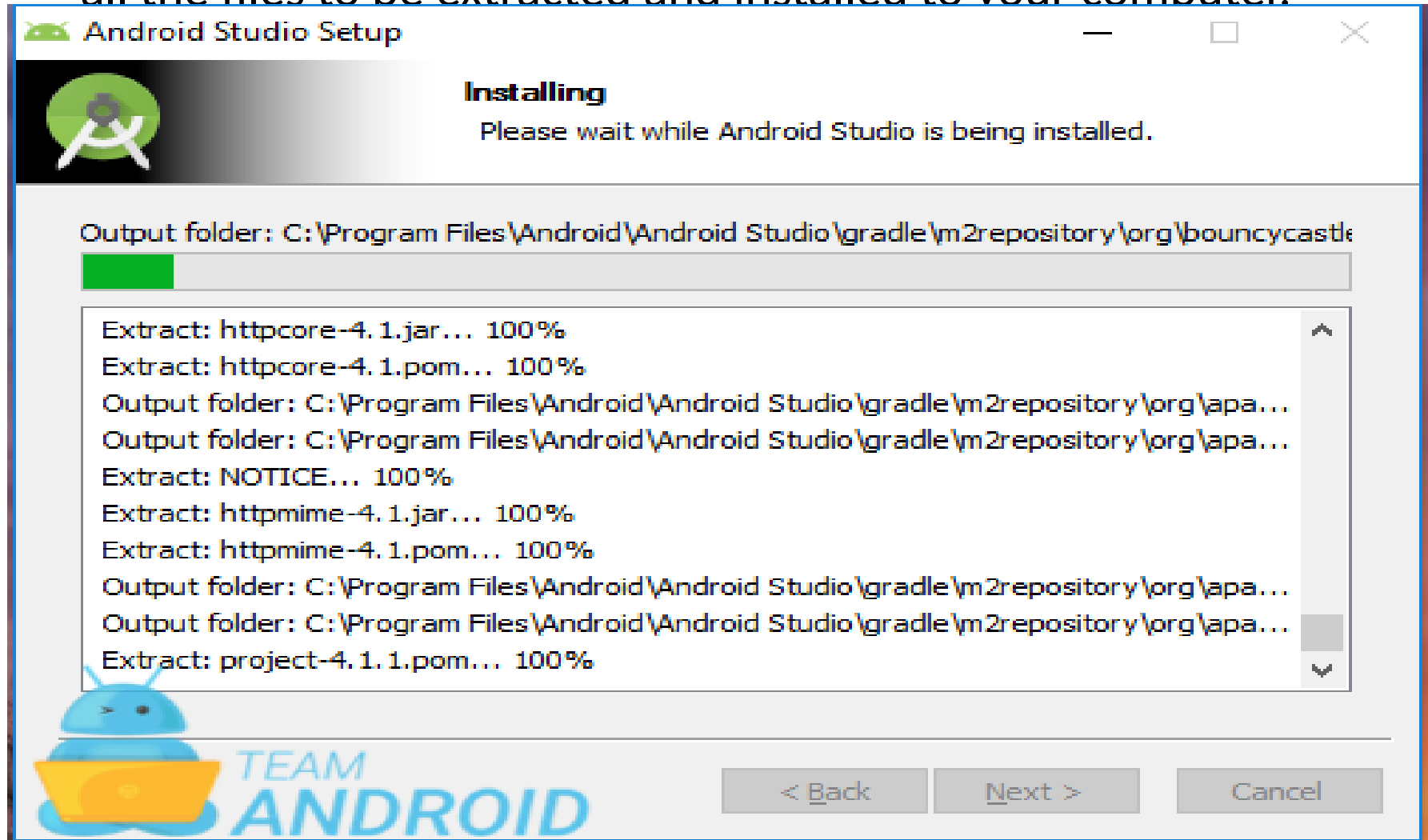
- **Step 7** – Yes, you would want to have to a Start menu folder. Just click *Install* to continue.





How to Install and Setup Android Studio

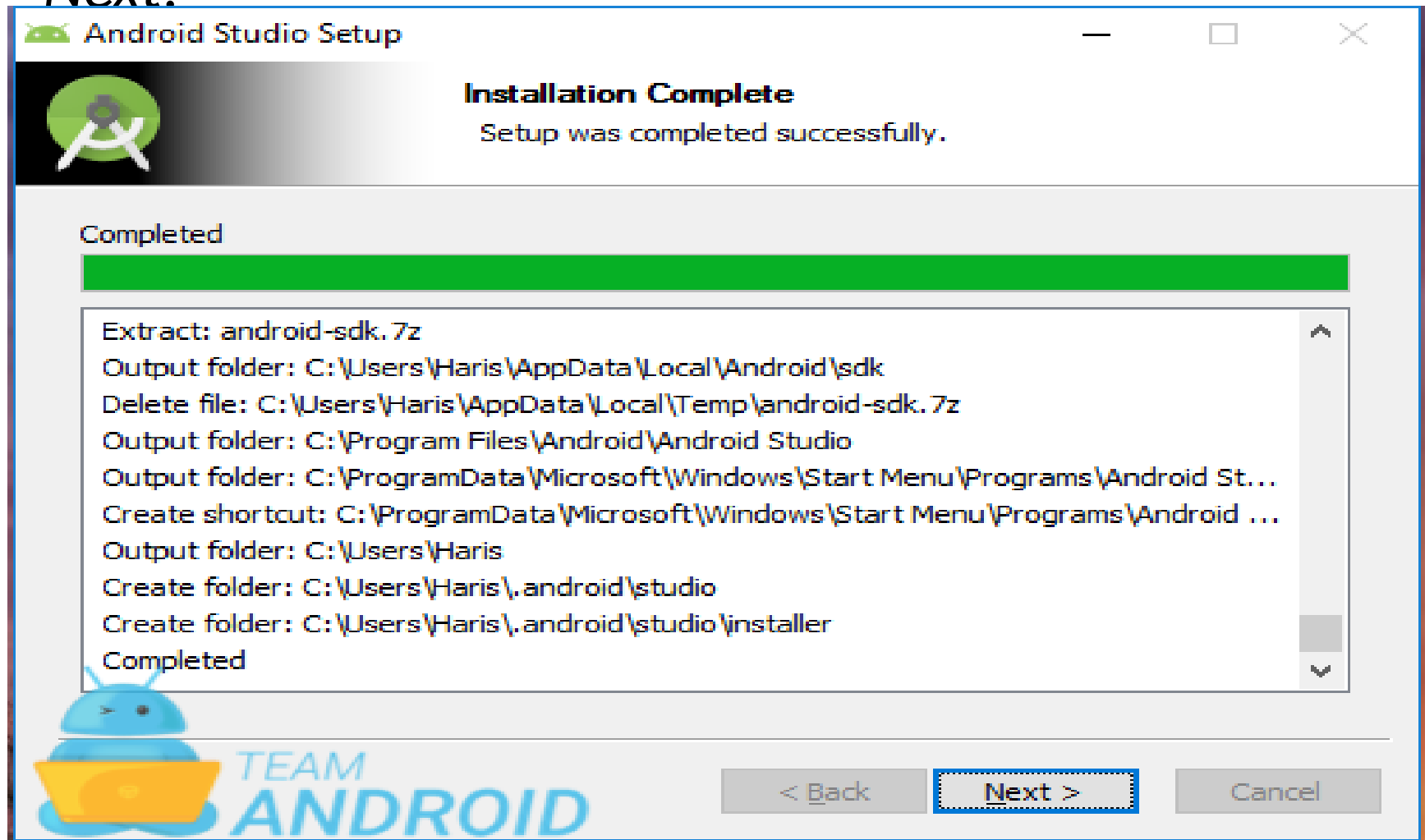
- **Step 8** - Installation should begin now. This may take a while for all the files to be extracted and installed to your computer.





How to Install and Setup Android Studio

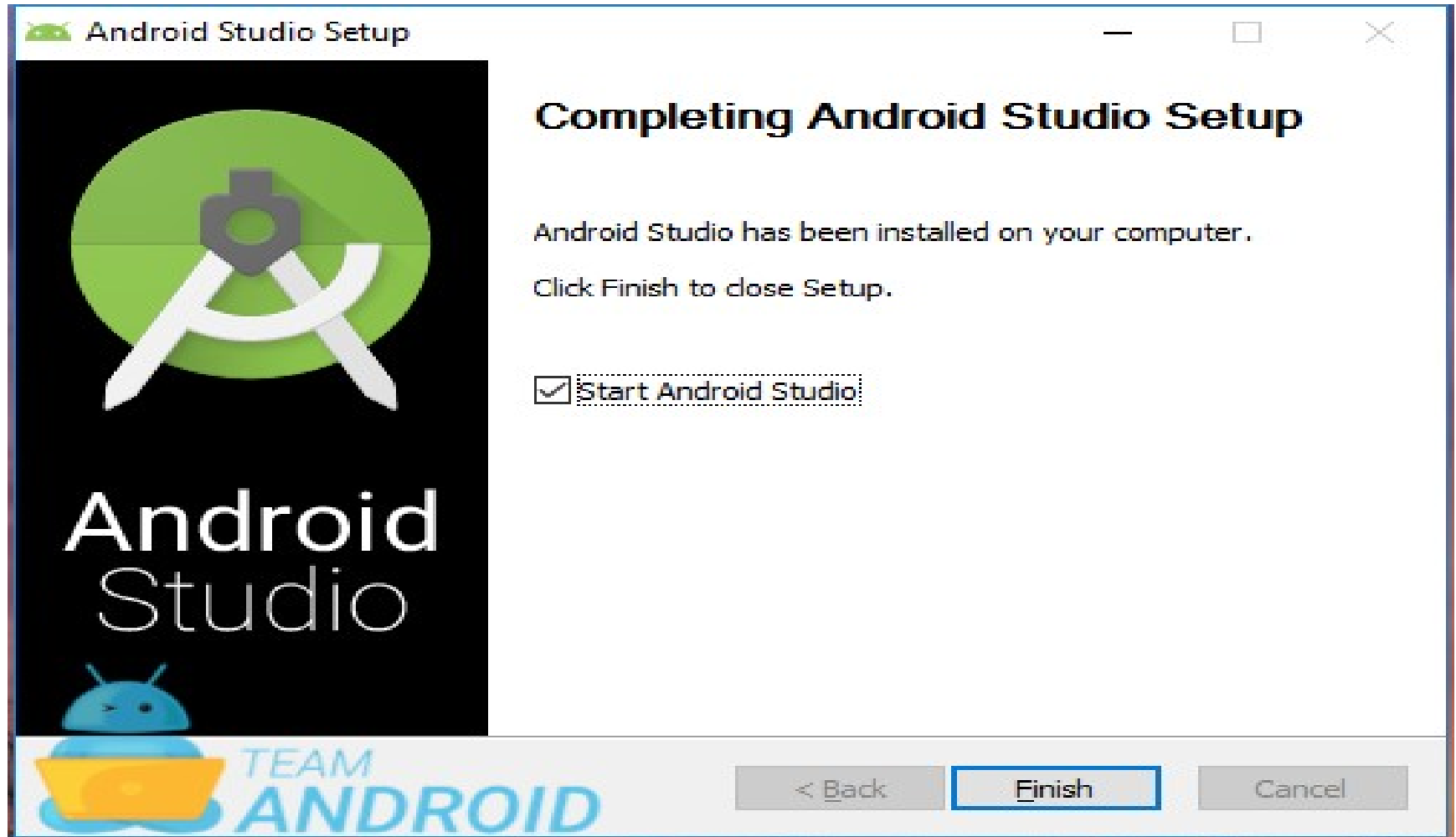
- **Step 9** – Once the installation is completed, click *Next*.





How to Install and Setup Android Studio

- **Step 10** – Yep, this is it. You are all set to launch Android Studio for the first time. Click *Finish* to proceed.





How to Install and Setup Android Studio

- download android studio from

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

- Extract the archive file into an appropriate location for your applications, eg: /opt.
- **sudo unzip android-studio-ide-145.3537739-linux.zip -d /opt**
- To launch Android Studio, navigate to the
- **] cd /opt/android-studio/bin directory**
- Execute **] ./studio.sh.**

Android Versions

- ❖ 2001 - Alpha - Beta
- ❖ Android 1.5 - Cupcake
- ❖ Android 1.6 - Donut
- ❖ Android 2.1 - Éclair
- ❖ Android 2.2 - Froyo
- ❖ Android 2.4 - Gingerbread
- ❖ Android 3.2 - Honeycomb
- ❖ Android 4.0 - Ice Cream
- ❖ Android 4.1 - Jelly Bean
- ❖ Android 4.4 - KitKat
- ❖ Android 5 - Lollipop
- ❖ Android 6 - Marshmallow
- ❖ Android 7 - Nougat
- ❖ Android 8 - Oreo
- ❖ Android 9 - Pie
- ❖ Android 10- Q
- ❖ 2020 Android 11- Guess



Alpha



Beta



Cupcake



Donut



Eclair



Froyo



Gingerbread

Android Nougat



Honeycomb



Ice Cream Sandwich



Jelly Bean



KitKat



Lollipop



Marshmallow



Nougat



Alpha and Beta.

- software did include a suite of early Google apps like Gmail, Maps, Calendar and YouTube, all of which were integrated into the operating system

Cupcake

- **Bluetooth** support
- **Soft-keyboard** with text-prediction
- **Video recording** was added to the camera along with the ability to directly upload videos to YouTube.
- The browser got a speed improvement along with **copy-paste support.**



Donut

- **Google Maps Navigation** was added with turn by turn satellite navigation support.
- Donut **included universal search feature** which allowed us to pinpoint apps on the phone or searching the web.

Eclair

- Support for **multiple Google accounts** was added.
- Support for searching within text messages. **speech-to-text function.**

Froyo

- Settings joined contacts and email in **backing up to Google's servers** allowing you to automatically restore everything on a new device.
- Enhanced **Bluetooth** compatibility with docks and **car speakers**.
- **Portable WiFi hotspot** to share the device's 3G connection with other gadgets.

Gingerbread



- **Front facing camera** support for **video calling**.
- Download manager for keeping an eye on your downloads.



Ice Cream Sandwich

- **Hardware buttons are dropped** in favor of on-screen buttons.

Jellybean

- **Face recognition for unlocking** the phone.

KitKat

- New dialer with **Caller ID feature**.
- **Emoji keyboard** for emoticons.

Lollipop

- **Enhanced battery life** with new Battery Saver mode.



Marshmallow

- Official **fingerprint support** for devices.
- Support for mobile payments via Android Pay.

Nougat

- **Multi Window** for using two apps at the same time.

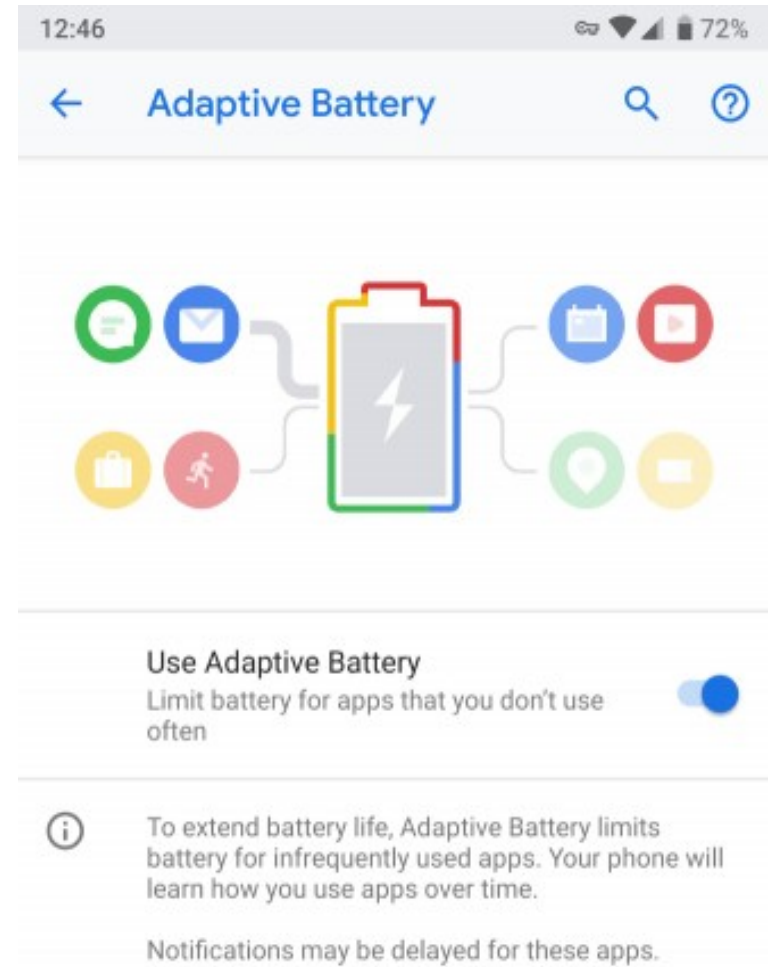
Oreo

- **Faster boot time:** On Pixel devices, you can now experience up to twice as fast boot times compared to Nougat.
- **AutoFill** for filling and remembering passwords within apps.



Pie

- **Adaptive Battery** goes further by learning about the apps and services you use most often, then adjusting what you don't use as much to use less battery.



MY LAUNDRY
JUST SENT ME
A FRIEND
REQUEST.



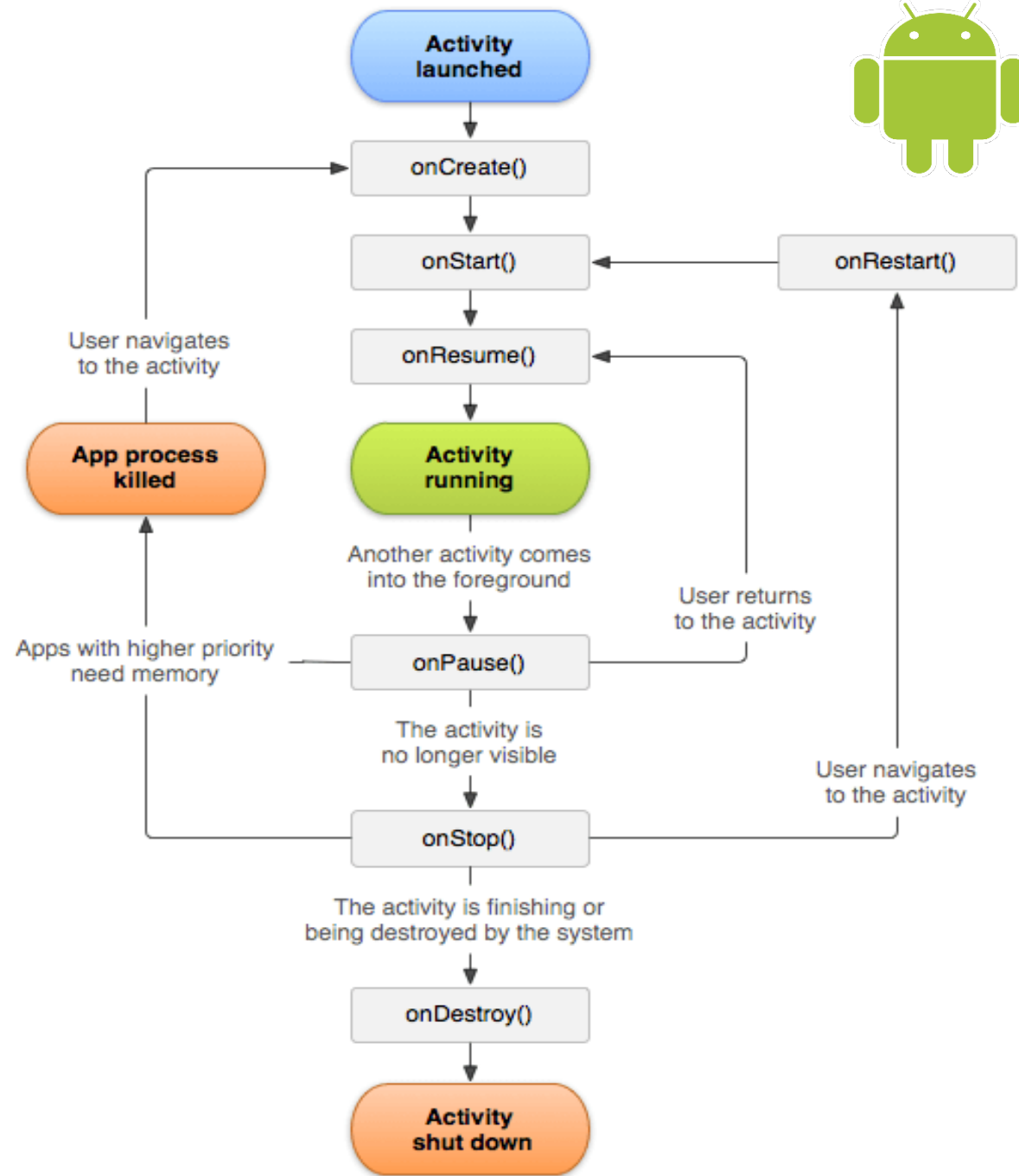
CARTOONSTOCK
.com

Search ID: Jsh121018

STAHLER.
10/18

Activity Life Cycle

- An **activity** is the **single screen in android**. It is like window or frame of Java.
- By the help of activity, you can place all your UI components or widgets in a single screen.





Activity Life Cycle

- **onCreate** : Called when the **activity is first created**. This is where you should do all of your normal static set up: create views, bind data to lists, etc.
- **onStart**: Called when the **activity is becoming visible to the user**. Followed by `onResume()` if the activity comes to the foreground, or `onStop()` if it becomes hidden.
- **onResume** : Called when the **activity will start interacting with the user**. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by `onPause()`.



Activity Life Cycle

- **onPause** : Called as part of the activity lifecycle when an **activity is going into the background, but has not (yet) been killed**. The counterpart to onResume(). When activity B is launched in front of activity A, this callback will be invoked on A.
- **onStop**: Called **when you are no longer visible to the user**. You will next receive either onStart(), onDestroy(), or nothing, depending on later user activity.
- **onRestart**: Called after your activity has been stopped, **prior to it being started again**. Always followed by onStart()
- **onDestroy** : The final call you receive before your activity is destroyed. This can happen either **because the activity is finishing (someone called finish() on it, or because the system is temporarily destroying this instance of the activity to save space**



System Permissions

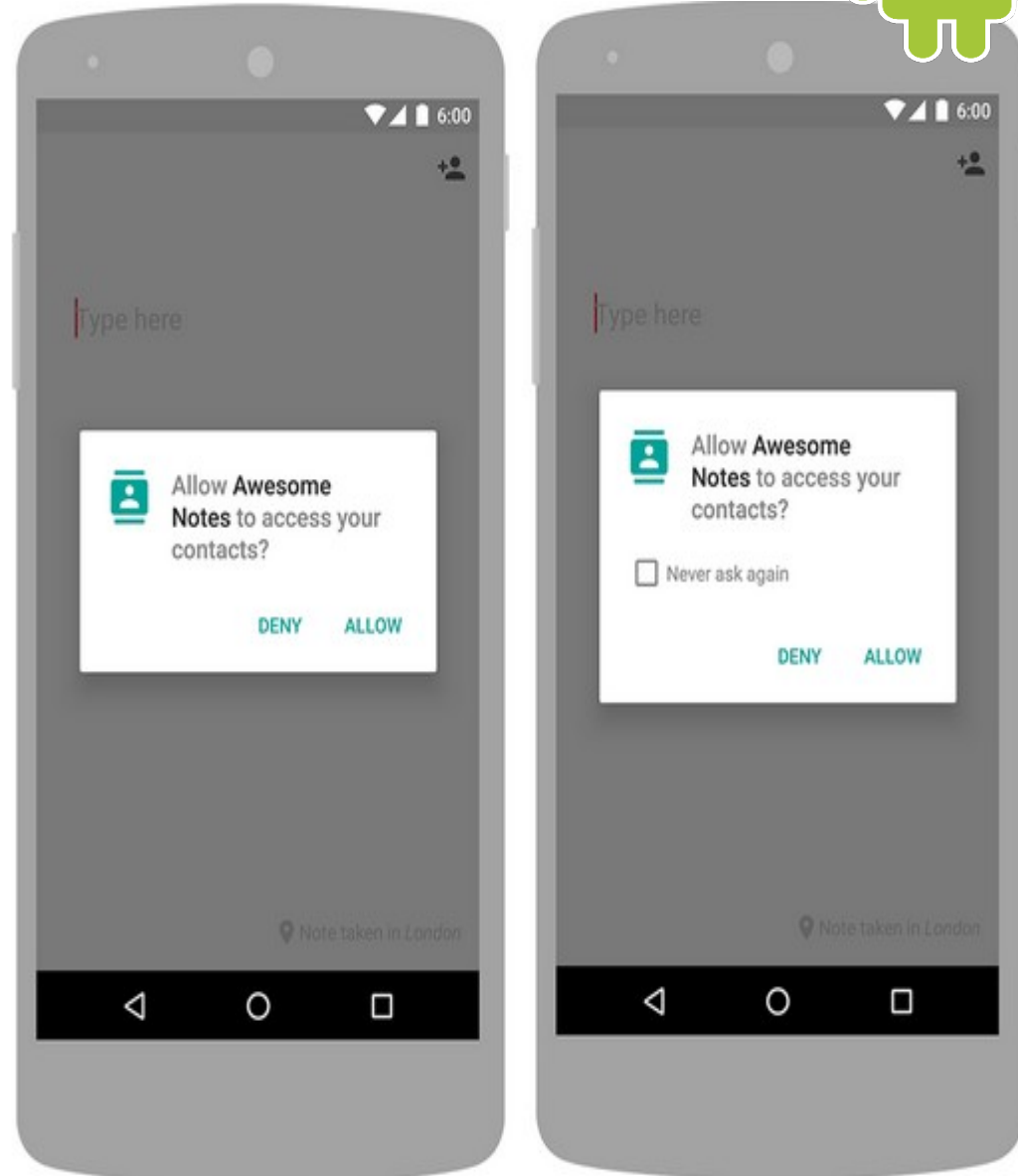
- The purpose of a **permission** is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.snazzyapp">
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <application ...>
    ...
  </application>
</manifest>
```



System Permissions

- The purpose of a *permission* is to protect the privacy of an Android user. **Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet).** Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.



System Permissions



<manifest

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
package="com.example.snazzyapp">
```

```
<uses-permission
```

```
android:name="android.permission.SEND_SMS"/
```

```
>
```

```
<application ...>
```

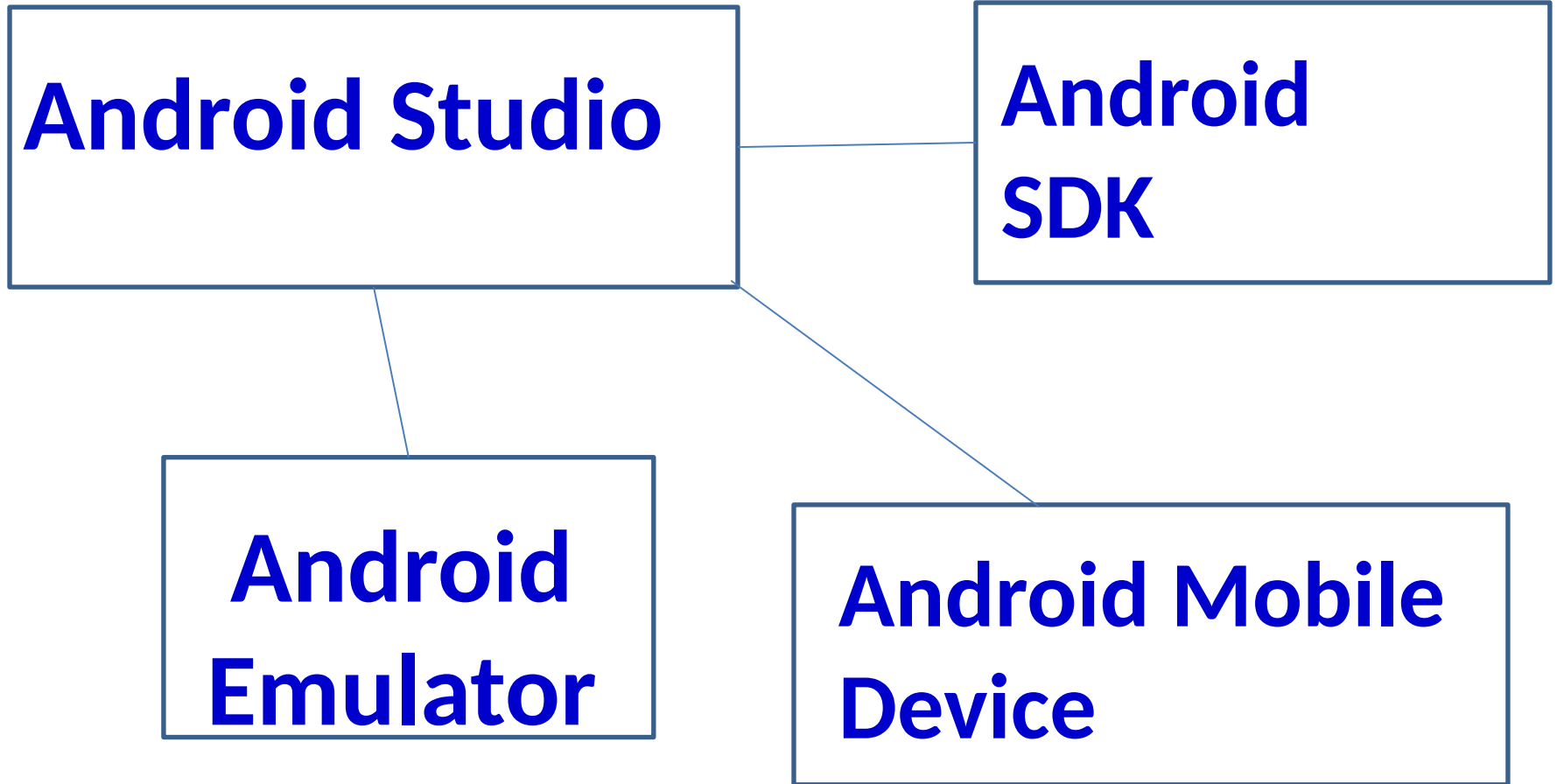
```
...
```

```
</application>
```

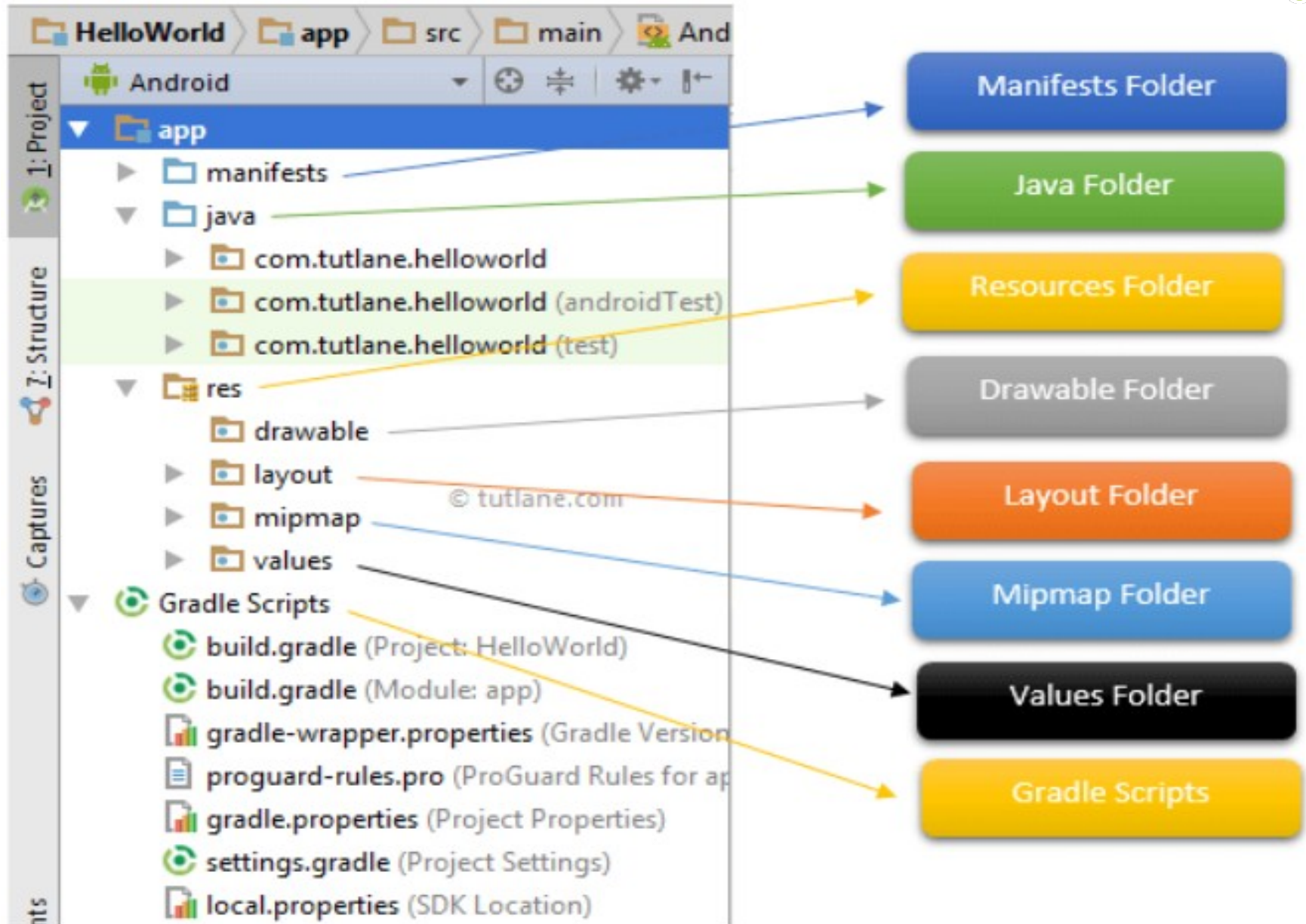
```
</manifest>
```



Android Application Development



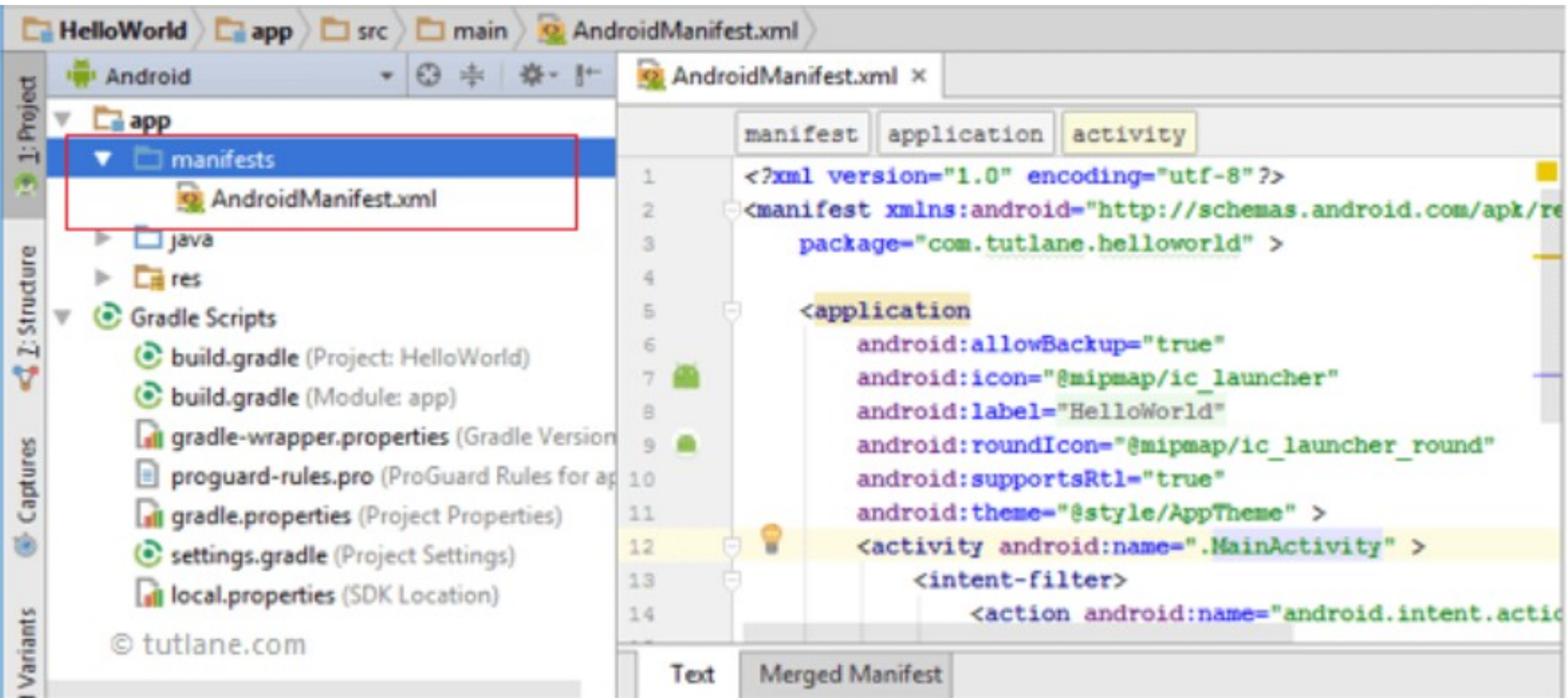
Android Application Structure



Manifests Folder



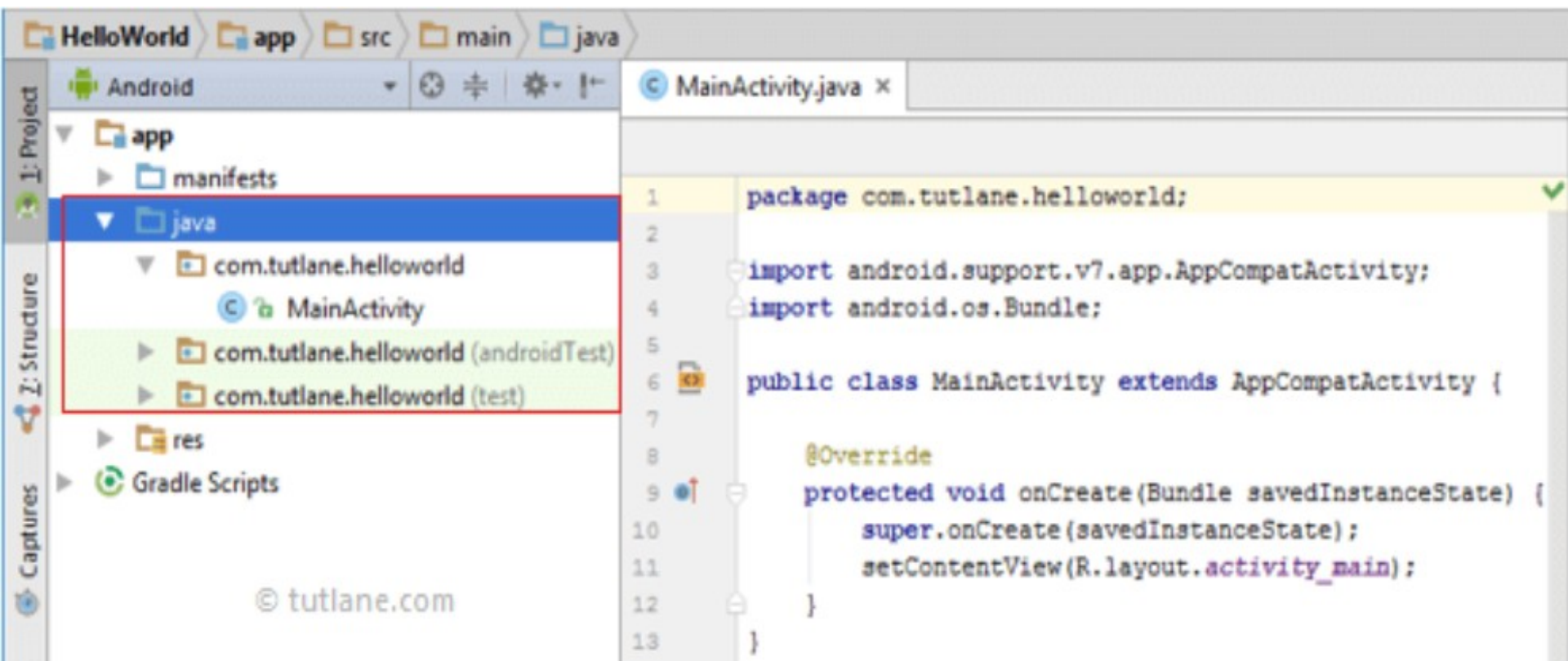
- This folder will contain a manifest file (AndroidManifest.xml) for our android application. **This manifest file will contain information about our application such as android version, access permissions, metadata, etc.** of our application and its components. The manifest file will act as an **intermediate between android OS and our application.**





Java Folder

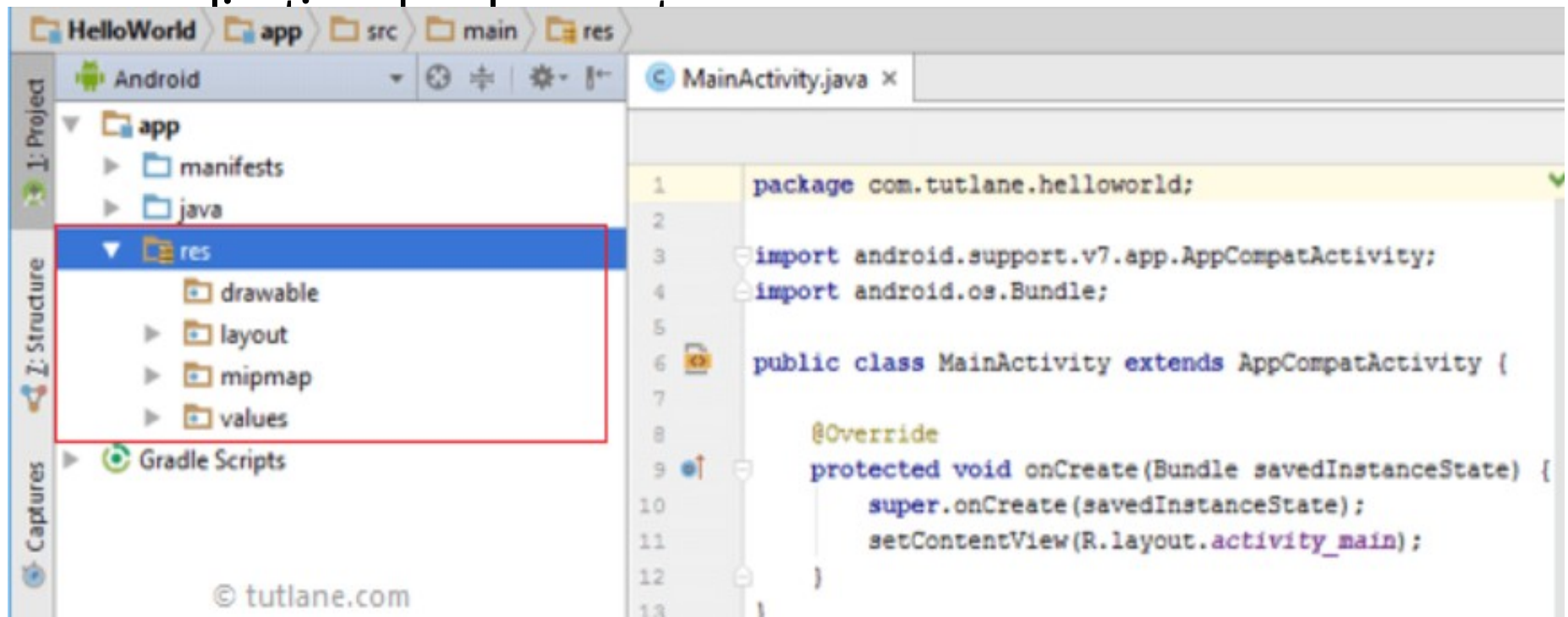
- This folder will contain all the java source code (.java) files which we'll create during the application development, including JUnit test code. Whenever we create any new project / application, by default the class file **MainActivity.java** will create automatically.



res (Resources) Folder



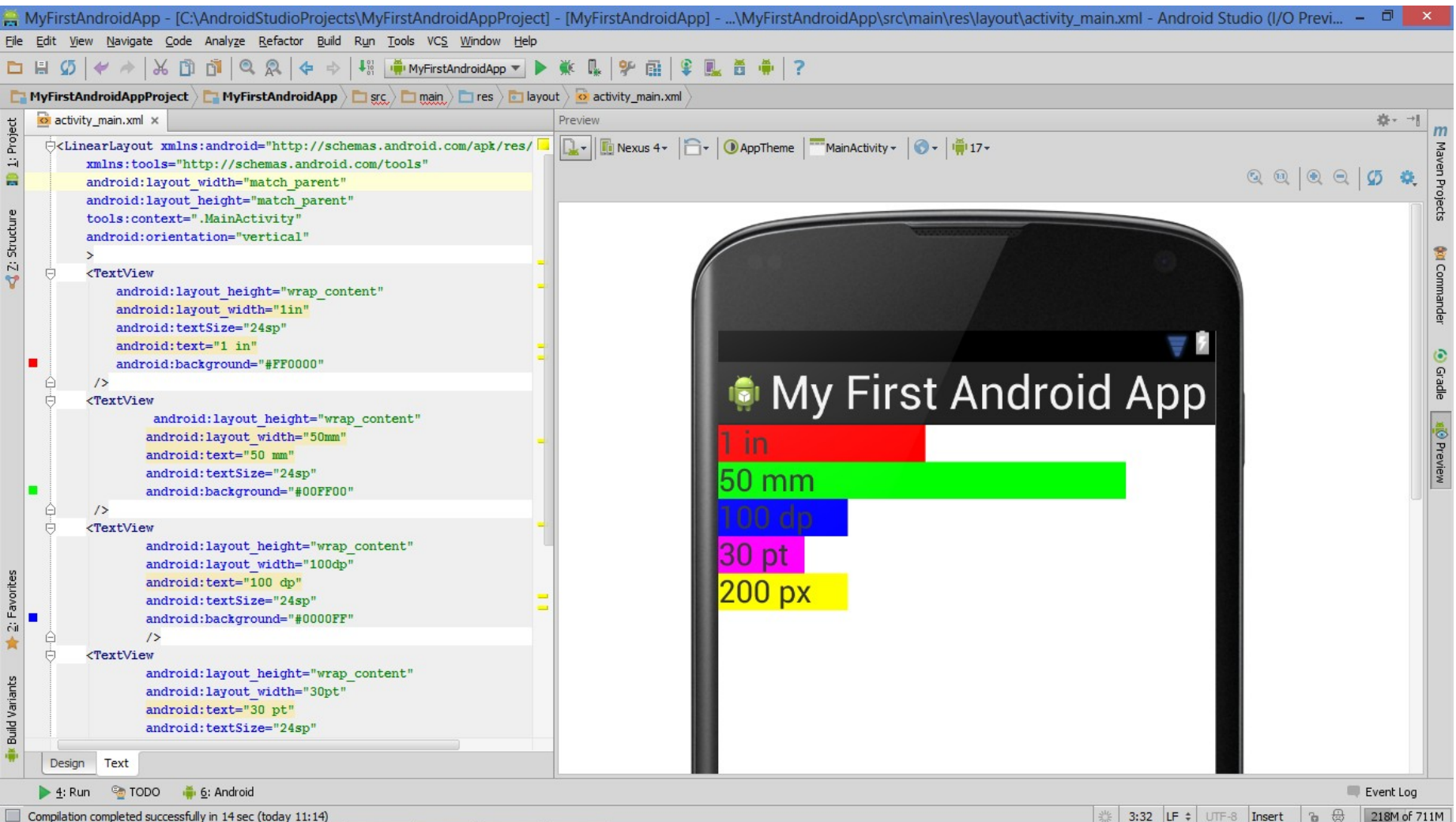
- It's an important folder which will contain all non-code resources, such as bitmap images, UI strings, XML layouts like as show below.
- **Drawable Folder (res/drawable):** It will contains the different type of images as per the requirement of application. It's a best practice to add all the images in **drawable** folder other than app / launcher icons for the



res (Resources) Folder



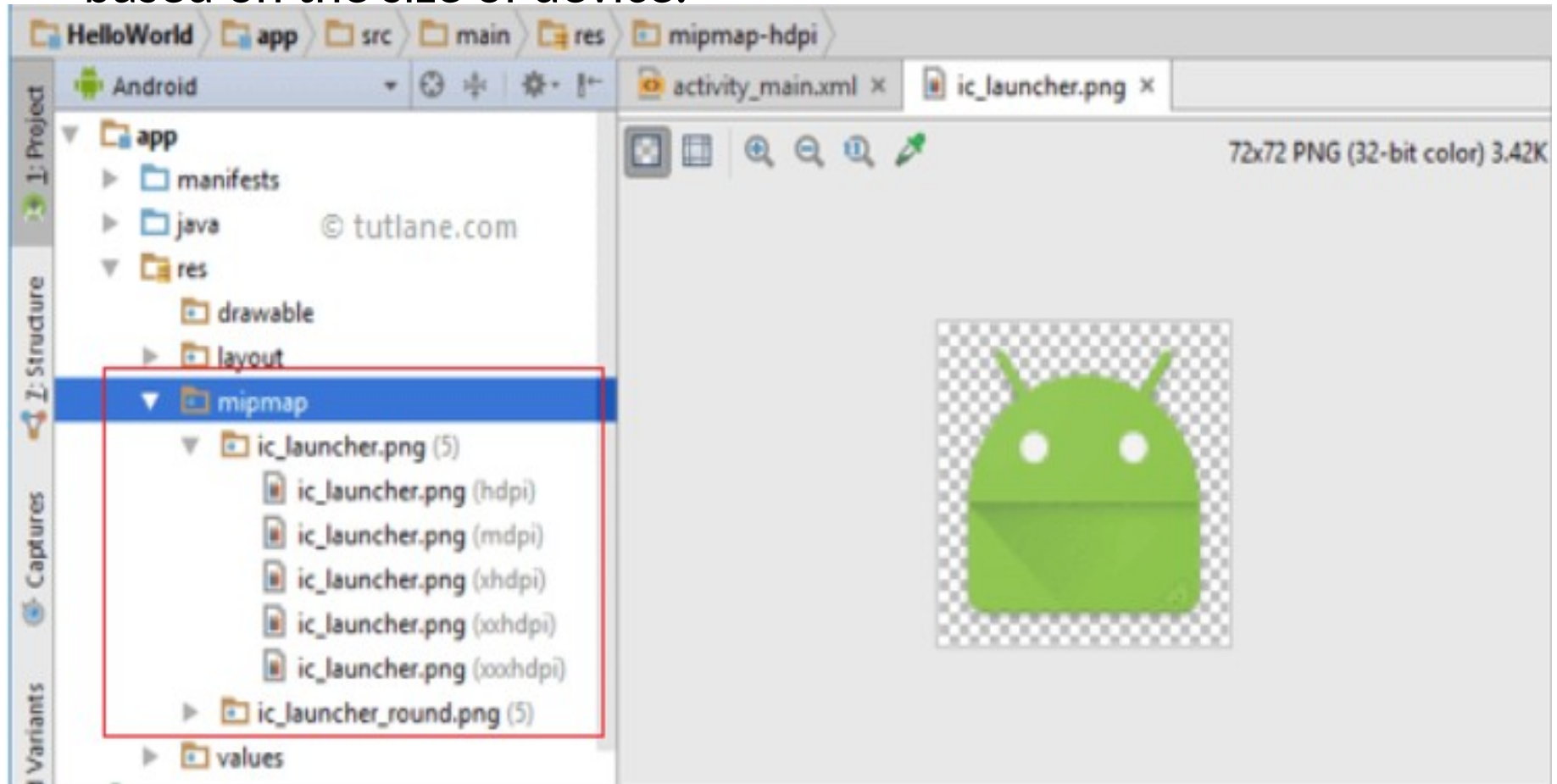
- **Layout Folder (res/layout):** This folder will contain all XML layout files which we used to define the user Interface of our application. Following is the structure of layout folder in android application.



res (Resources) Folder



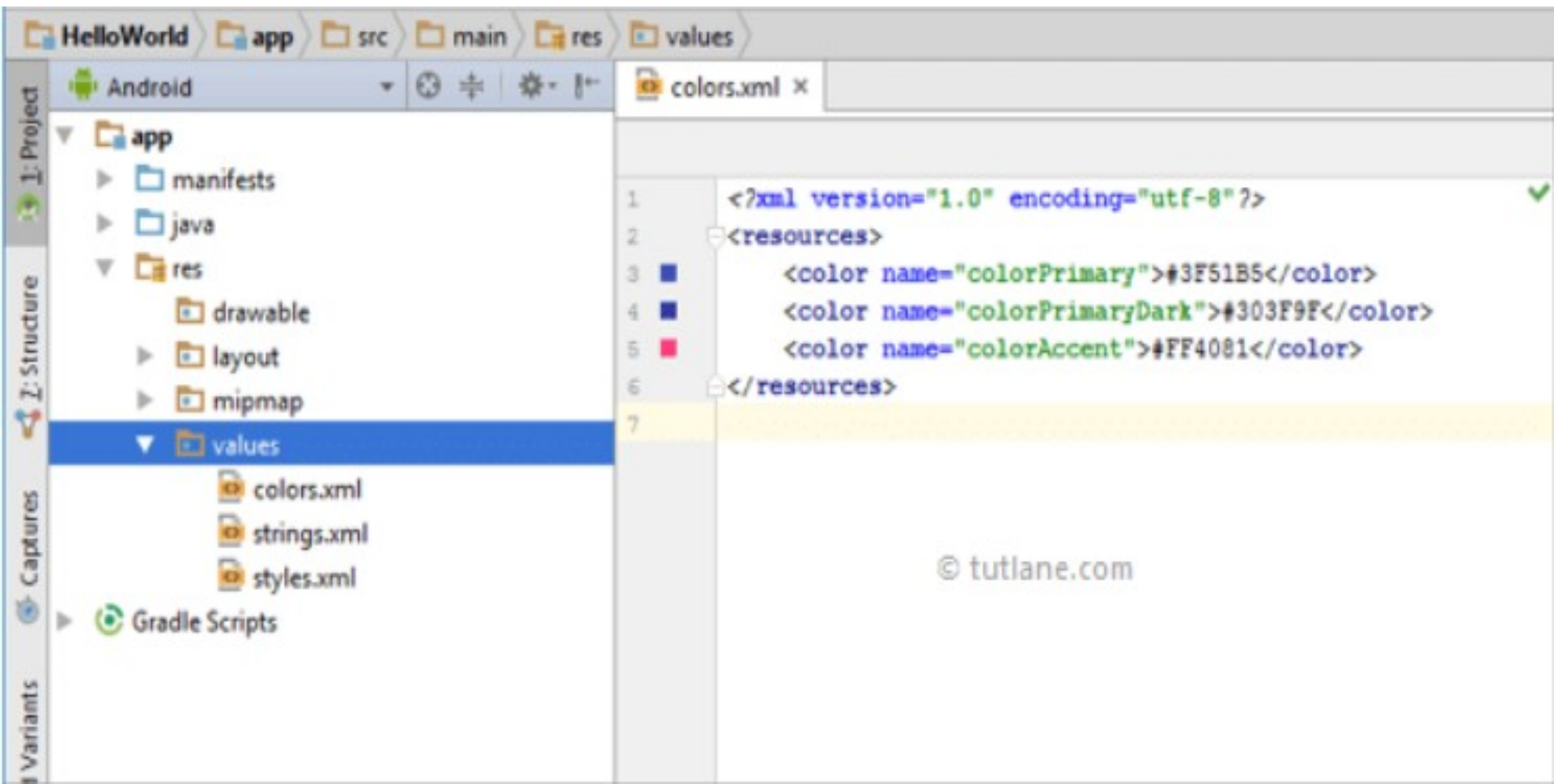
- **Mipmap Folder (res/mipmap):** This folder will contain app / launcher icons which are used to show on the home screen. It will contain different density type of icons such as hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi, to use different icons based on the size of device.



res (Resources) Folder

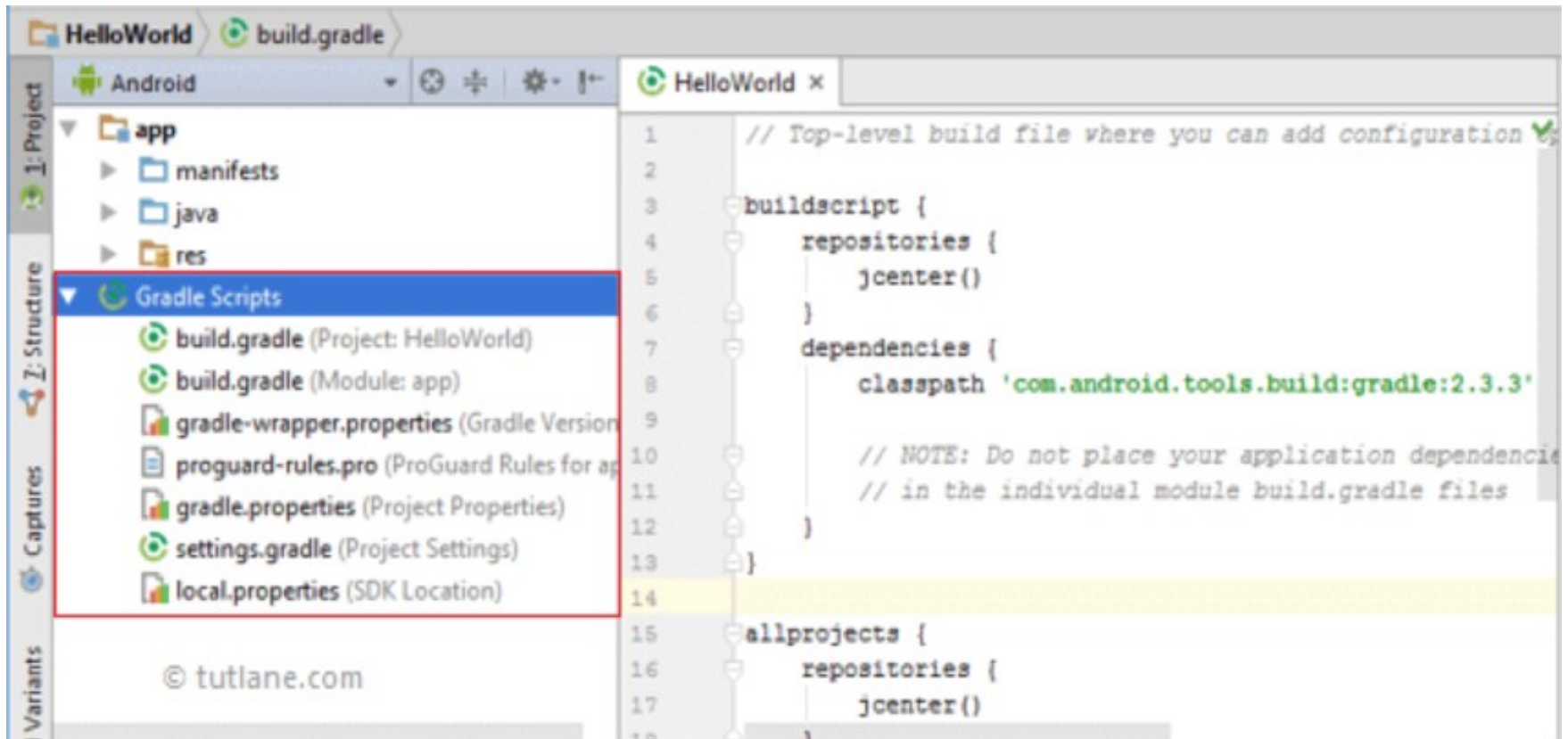


- **Values Folder (res/values):** This folder will contain a various XML files, such as strings, colors, styles definitions and static array of strings or integers. Following is the structure of **values** folder in android application.





- In android, Gradle means automated build system and by using this we can define a build configuration that apply to all modules in our application. In gradle **build.gradle (Project)**, **build.gradle (Module)** are used to build configurations that apply to all our app modules or specific to one app module.





Android Button

- Android Button represents a push-button. The **android.widget.Button** is subclass of **TextView** class and **CompoundButton** is the subclass of **Button** class.
- There are different types of buttons in android such as **RadioButton**, **ToggleButton**, **CompoundButton** etc.





Attributes of Button in Android

- **id:** id is an attribute used to **uniquely identify a text Button**.
`android:id="@+id/simpleButton"`
- **gravity:** The gravity attribute is an optional attribute **which is used to control the alignment of the text** like left, right, center, top, bottom, center_vertical, center_horizontal
`android:gravity="right|center_vertical"`
- **text:** text attribute is **used to set the text in a Button**. We can set the text in **xml** as well as in the **java** class.

Using XML"

`android:text="AbhiAndroid"`

Using Java :

```
Button button = (Button) findViewById(R.id.simpleButton);  
button.setText("AbhiAndroid");
```





Attributes of Button in Android

- **textColor:** textColor attribute is used to set the text color of a Button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.



android:textColor="#f00"

- **textSize:** textSize attribute is used to set the size of the text on Button. We can set the text size in sp(scale independent pixel) or dp(density pixel).



android:textSize="25sp"

- **textStyle:** textStyle attribute is used to set the text style of a Button. The possible text styles are bold, italic and normal.



android:textStyle="bold | italic"

Android Button 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="example.javatpoint.com.sumoftwonumber.MainActivity">
```

```
<EditText
```

```
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="61dp"
    android:ems="10"
    android:inputType="number"
    tools:layout_editor_absoluteX="84dp"
    tools:layout_editor_absoluteY="53dp" />
```

```
<EditText
```

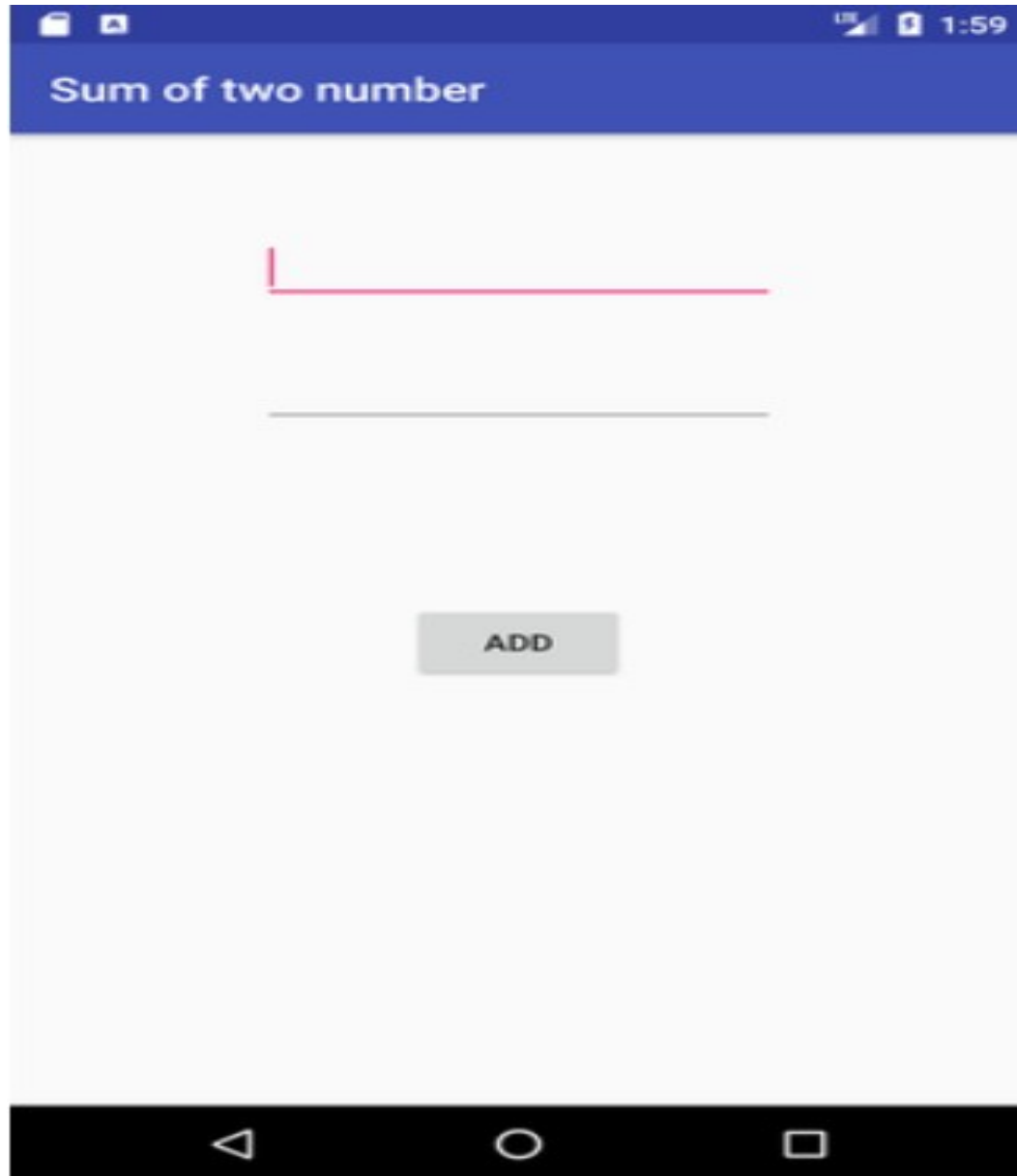
```
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="32dp"
    android:ems="10"
    android:inputType="number"
    tools:layout_editor_absoluteX="84dp"
    tools:layout_editor_absoluteY="127dp" />
```

```
<Button
```

```
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="109dp"
    android:text="ADD"
    tools:layout_editor_absoluteX="148dp"
    tools:layout_editor_absoluteY="266dp" />
```

```
</RelativeLayout>
```

Android Button Example activity_main.xml



Android Button Example MainActivity.java



```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText text1, text2;
    private Button buttonSum;

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

        addListenerOnButton();
    }
}
```

```
public void addListenerOnButton() {

    text1 = (EditText) findViewById(R.id.editText1);
    text2 = (EditText) findViewById(R.id.editText2);
    buttonSum = (Button) findViewById(R.id.button);

    buttonSum.setOnClickListener(new
    View.OnClickListener() {

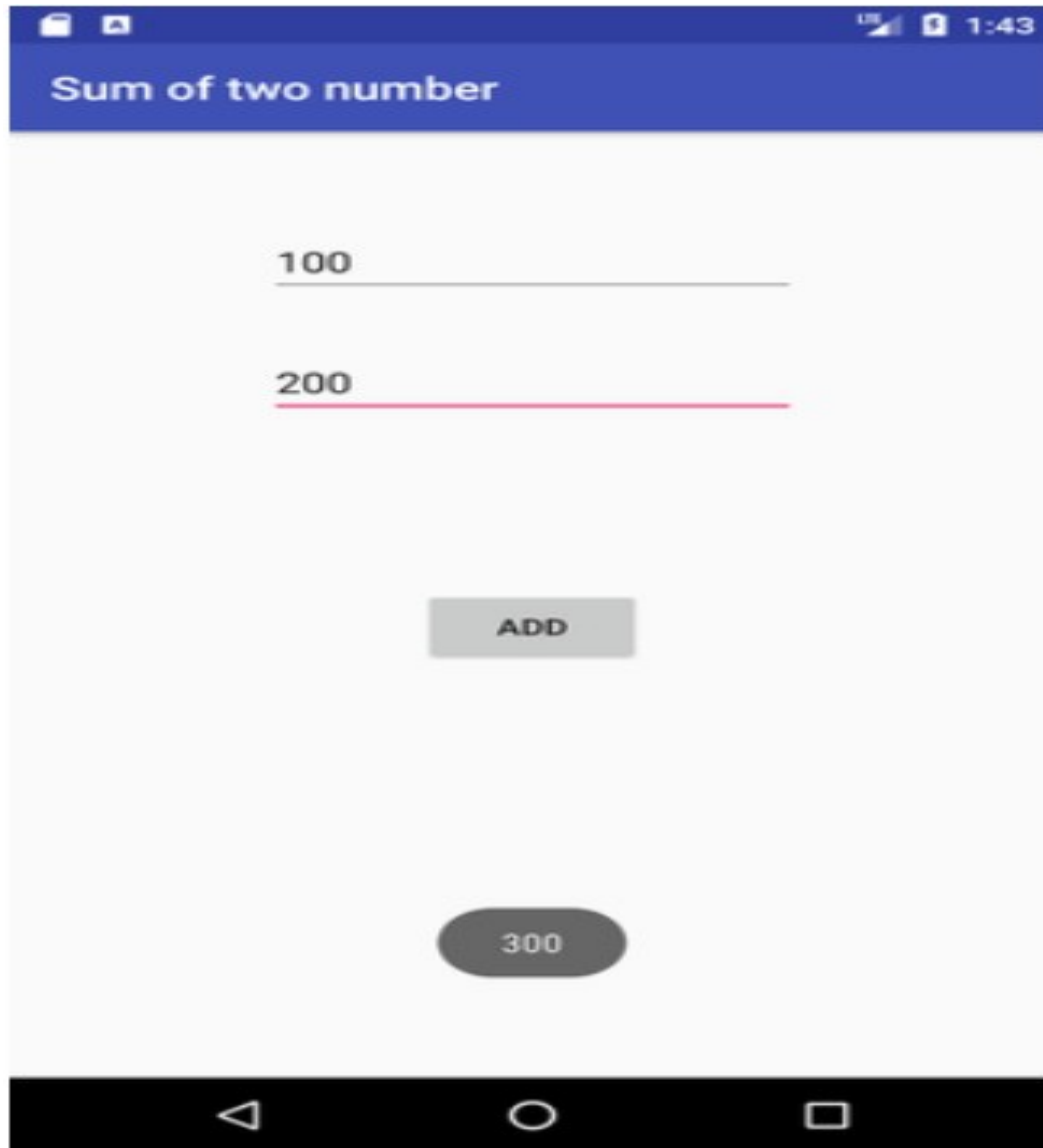
        @Override
        public void onClick(View view) {

            String value1=text1.getText().toString();
            String value2=text2.getText().toString();
            int a=Integer.parseInt(value1);
            int b=Integer.parseInt(value2);
            int sum=a+b;

            Toast.makeText(getApplicationContext(),String.valueOf(s
            um), Toast.LENGTH_LONG).show();

        }
    });
}
}
```


Android Button Example OutPut





Google Maps in Android

iBores

THE **GPS** ON THIS IS A MIRACLE. IT TRACKS YOU WHEREVER YOU ARE IN THE WORLD



.... SO YOU NEVER GET LOST



AMAZINGLY CLEAR SOUND AFFECT

by Blower



CARTOONSTOCK.com

Search ID: pbon7

CARTOONSTOCK.com

Search ID: pbon7

Google Maps Api Key

- To use the Google Maps functionality in Android, you **need to register for a Maps API** key with Google.
- **Without this key, you cannot display Google Maps on your Activity.**
- An **API key** is needed to access the Google Maps servers.



Steps For Getting The Google Maps Api Key


- **Step 1:** Open Google developer console and signin with your gmail account:

<https://console.developers.google.com/project>


A screenshot of the Google Developer Console login interface. At the top is the Google logo. Below it, the text "Hi Ranjeetsingh" is displayed. Underneath is a dropdown menu showing a profile icon with the letter 'R' and the email address "ranjeetsinghsuryawanshi@gmail.com" with a downward arrow. Below the dropdown is a password input field with the placeholder text "Enter your password". To the right of the password field is a small icon of an eye with a slash through it, indicating a toggle for password visibility. At the bottom left is a link that says "Forgot password?". At the bottom right is a blue button with the text "Next".

Google

Hi Ranjeetsingh

 ranjeetsinghsuryawanshi@gmail.com ▼

Enter your password



[Forgot password?](#) [Next](#)

Steps For Getting The Google Maps Api Key



- **Step 2:** Now create new project. You can create new project by clicking on the **Create Project** button and give name to your project.

A screenshot of the Google Cloud Platform console. The browser address bar shows 'https://console.developers.google.com/cloud-resource-manager'. Below the address bar, there is a notification about a free trial. The main header shows 'Google APIs' with a search bar. The main content area is titled 'Manage resources' and features a prominent '+ CREATE PROJECT' button, which is circled in black and pointed to by a blue arrow. To the right of this button is a 'DELETE' button. Below the buttons is a filter input field and a 'Columns' dropdown. A table lists existing projects, with one entry visible: 'My Project' with ID 'zinc-wares-178607'.

Steps For Getting The Google Maps Api Key



← ⓘ 🔒 | <https://console.developers.google.com/projectcreate?project=zinc-wares-178607>

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products

☰ Google APIs 🔍

New Project

Project Name *
MyFirstGPS ⓘ

Project ID: myfirstgps-217806. It cannot be changed later. [EDIT](#)

Location *
 No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

Steps For Getting The Google Maps Api Key



Browser address bar: <https://console.developers.google.com/home/dashboard?creatingProject=true&project=zinc-wares-1> Search

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

Google APIs My Project

DASHBOARD ACTIVITY

Project info

Project name
My Project

Project ID
zinc-wares-178607

Project number
739260133835

→ Go to project settings

Resources

This project has no resources

API APIS

Requests (requests/sec)

No data is available for the selected time frame.

11:30 11:45 12 PM 12:15

→ Go to APIs overview

Steps For Getting The Google Maps Api Key



Browser address bar: <https://console.developers.google.com/apis/dashboard?project=zinc-wares-178607&duration=PT1H>

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

Google APIs My Project

APIs & Services

Dashboard

Library

Credentials

Dashboard

+ ENABLE APIS AND SERVICES

Enabled APIs and services

Some APIs and services are enabled automatically

Activity for the last hour

1 ho

Traffic

Errors

A blue arrow points from the bottom right towards the 'ENABLE APIS AND SERVICES' button, highlighting the next step in the process.

Steps For Getting The Google Maps Api Key



Google Maps Tutorial ... Obtaining a Google Ma... Android Google Maps Creating a Google Map... How to generate and s...

https://console.developers.google.com/apis/library?project=zinc-wares-178607&q=google map andr Search

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

Google APIs My Project

Search google map android api


Filter by

CATEGORY

Maps (2)


Mobile (1)

2 results



Maps SDK for Android
Google

Maps for your native Android app.



Places SDK for Android
Google

Make your Android app stand out with detailed information about 100 million places

Steps For Getting The Google Maps Api Key

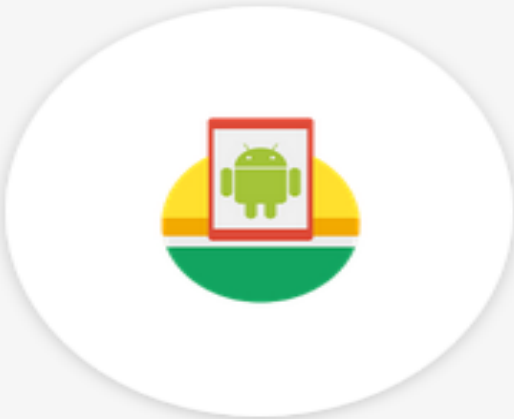


← ⓘ 🔒 | <https://console.developers.google.com/apis/library/maps-android-backend.googleapis.com?q=goo>

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud pro

☰ Google APIs ☼ My Project 🔍

← API Library



Maps SDK for Android

Google

Maps for your native Android app.

[MANAGE](#) API enabled

Steps For Getting The Google Maps Api Key



Browser address bar: <https://console.developers.google.com/google/maps-apis/apis/maps-android-backend.googleapis.c>

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

Google APIs My Project

Google Maps

Maps SDK for Android DISABLE

Metrics Quotas Credentials

Use one of these credentials to access this API, or create new credentials by visiting [Credentials in the API Manager](#).

API keys

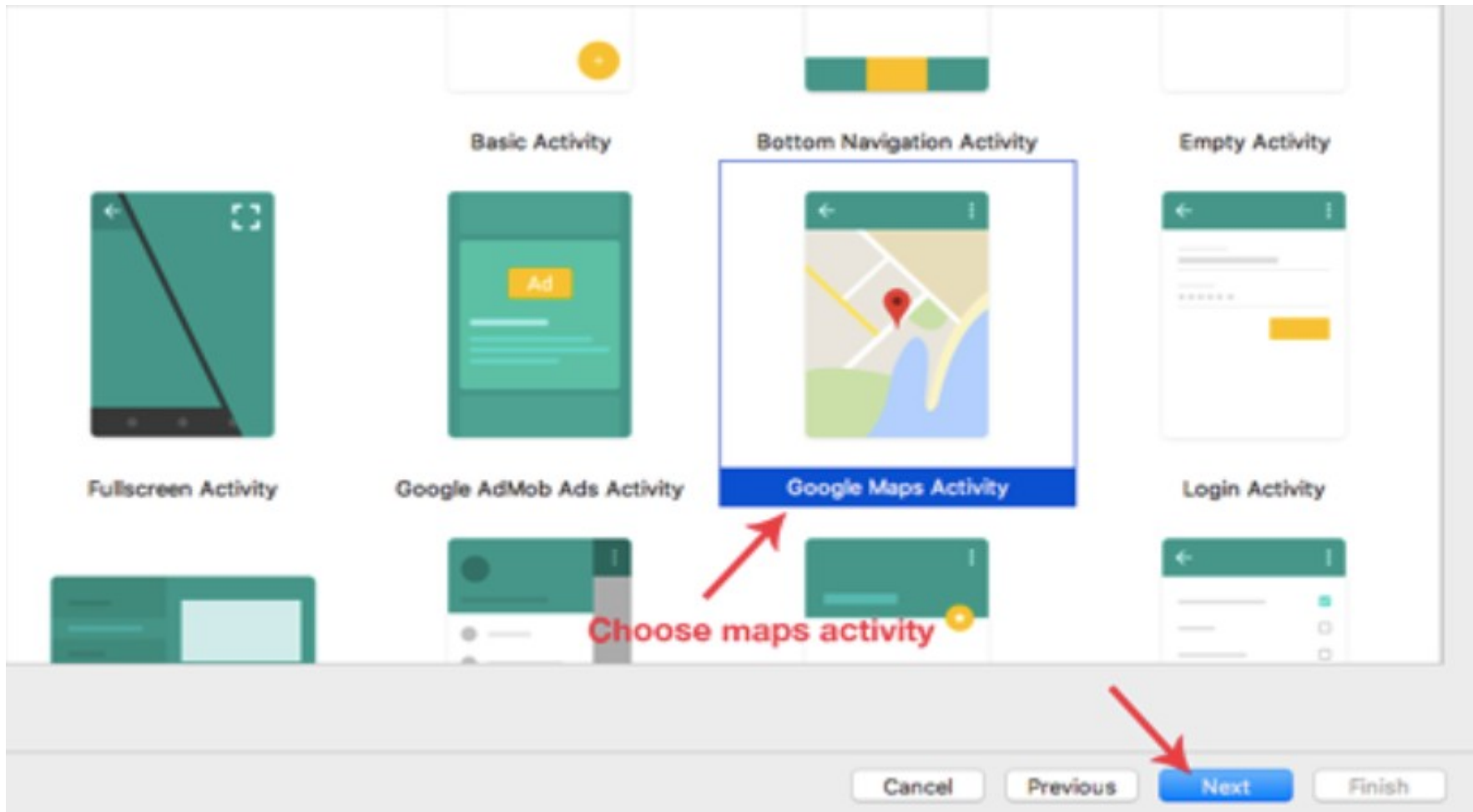
Name	Creation date	Restrictions	Key
API key 1	Sep 1, 2017	Android apps	AlzaSyDcZhlG2pkzPKelAet2TR-yavdnrVVStYs

Google Maps Example To Access User Current Location In Android Studio



- **Step 1:** Create a New Android Project and name it **AndroidGPSTrackingActivity**.

Google Maps Example To Access User Current Location In Android Studio

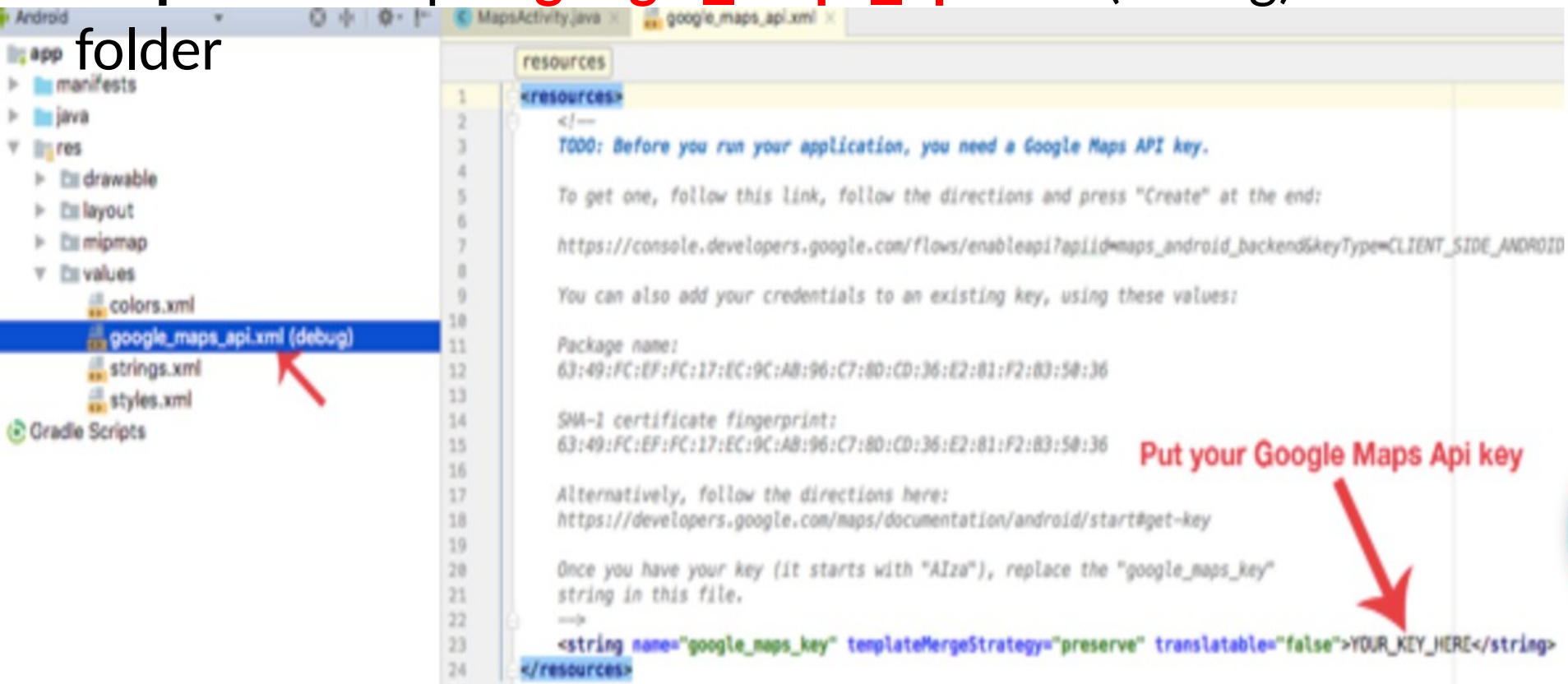


- **Step 2:** Now select **Google Maps Activity** and then click Next and finish.

Google Maps Example To Access User Current Location In Android Studio

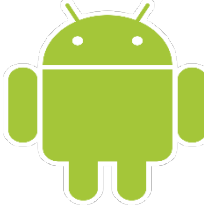


- Step 3: Now open **google_maps_api.xml** (debug) in values folder



<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AlzaSyDV2_xy58r15K6TskZy4KWMuhUDVq67jqM</string> </resources>

Google Maps Example To Access User Current Location In Android Studio



- **Step 4:** Now open **build.gradle** and add **compile 'com.google.android.gms:play-services:8.4.0'** in dependencies

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    androidTestCompile('com.android.support.test.espresso:  
        core:2.2.2', {  
            exclude group: 'com.android.support', module: 'support-  
                annotations'  
        })  
    compile 'com.android.support:appcompat-v7:26.+'  
    compile 'com.google.android.gms:play-services:8.4.0'  
    testCompile 'junit:junit:4.12'  
}
```


Google Maps Example To Access User Current Location In Android Studio



- **Step 5:** Now open activity_maps.xml and add a fragment code in it

```
<?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"
    android:orientation="vertical" >
    <fragment xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <Button android:id="@+id/btnShowLocation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Location"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```



Google Maps Example To Access User Current Location In Android Studio

- **Step 6:** Now define internet and location permissions in Android Manifest

```
<uses-permission  
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

```
<uses-permission  
    android:name="android.permission.INTERNET"/>
```

Google Maps Example To Access User Current Location In Android Studio



- **Step 7:** Now we will code `MapsActivity.java` file for inserting callbacks in Google Maps:



Search ID: mbcn3032

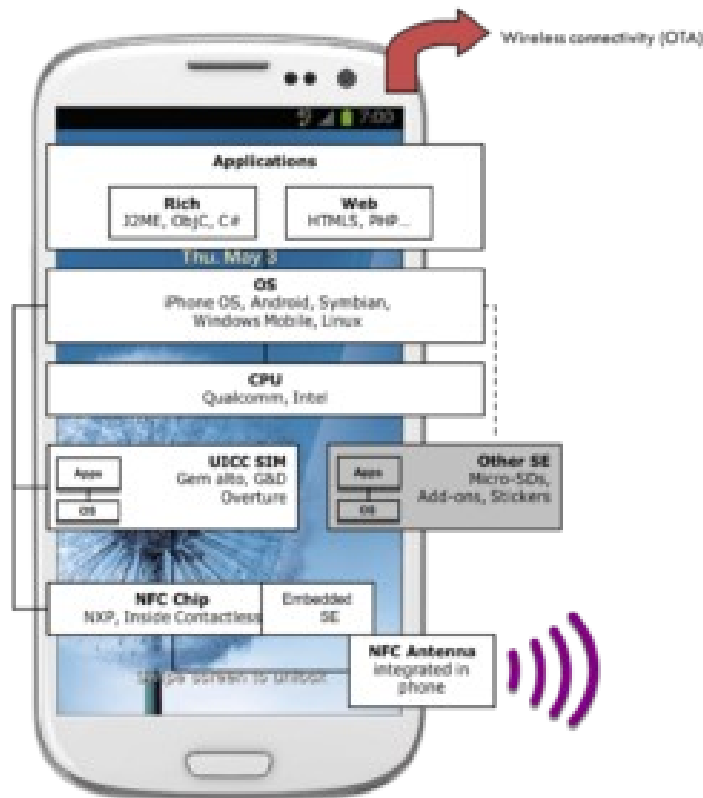
"Removing the phone is easy. Getting your head and arms to their original positions will take weeks of physical therapy."

NFC Device Architecture and Secure Element

What is special about the NFC device (a cellphone) architecture is that besides chip and antenna (similar to any other RFID device) it also includes the Secure Element. This is a component that allows for security of NFC communication

There are three types of Secure Elements:

- SIM Card as a Secure Element
- Embedded Secure Element
- Alternative (Add-on) Secure Elements



The SIM card as a Secure Element

The **SIM card** is the preferred Secure Element by mobile operators operating networks based on the GSM standard. SIM Cards are issued and owned by the mobile

operators and will put the mobile operator in a position, where they control, which NFC applications are being downloaded and used. Many mobile operators are working on a wallet, where they provide the NFC service to their subscribers and the NFC applications are installed on the SIM card.

There are a few appealing technical reasons for having the SIM as the Secured Element. Over The Air management of the SIM is standardized and is a proven technology being used by most mobile operators today. Portability is another interesting aspect; when the subscriber puts the SIM into a new NFC enabled phone the NFC applications follows. Many mobile operators have Automatic Device Detection solutions in place, where backend servers detect when the SIM is being put into a new device. The SIM also holds the mobile subscription and can always be reached by using the phone number of the subscription.

Embedded Secure Elements

The secure element can be integrated into the phone itself by the phone manufacturer as an **Embedded Secure Element** since it is a part of the phone. Many of the NFC chips come with an embedded Secure Element like the chips from NXP and Inside Secure. An interesting point with embedded Secure Elements is who will be the owner of the Secure Element.

Some handset manufacturers like Google and RIM are launching their own wallet, where they own and control the embedded Secure Element. Google has come up originally with Google Pay and dropped use of the NFC technology (perhaps due to the fact that it was owned and run by the manufacturers) but in 2015 Google acquired the IP of Softcard (carrier backed competitor) and integrated it into the Google Wallet as well as made an agreement with mobile operators (AT&T, T-Mobile US and Verizon) that they would bundle Google Wallet with their devices. The Google Wallet resulting from these agreements was renamed to Android Pay and is again utilizing NFC.

Alternative (add-on) form factors.

Over the last couple of years, we have seen multiple of different attempts to make existing mobile phones NFC enabled by **adding an external device with included NFC chip, antenna and Secure Element**. The most successful which has been used in many pilots and trials is the microSD card. Many mobile phones today have a microSD slot and by inserting a NFC enabled microSD, there would be millions of potential NFC users. Service providers like banks has performed many tests with microSDs and one of the benefits they see is that they can be the issuer and don't have to depend on a mobile operator.

One of the technical challenges with NFC enabled microSDs is the RF performance which is dependent on where microSD slot is placed on the phone and how the back cover is constructed. A few new NFC mobile phones support having a microSD as the Secure Element and communicate to the microSD using Secure Wire Protocol (SWP) as with the SIM card.

Site & Workflow Assessment Services

- Current Data Collection System Assessment – Analyze existing environment, identify current and potential problem areas and suggest improvements, which will address the problem areas.
- Create a detailed action plan to implement recommended improvements.

This includes:

- Time
- Cost
- Material
- Procedures
- Training
- Software Application

As a deliverable for this Phase, a detailed move-forward plan should be provided that describes how RFID can be deployed to address the current and future needs. The plan should include:

- RFID training for end-users and stake holders.
- Value Proposition – look at operation and identify what areas could benefit easily and dramatically from the introduction of RFID (find low-hanging fruit).
- Identify Benefactors – Based on identified processes, identify which areas of the operation would benefit from implementation and what intersecting processes should be considered.

- Suggest Move-forward Plan – recommend a plan test/pilot RFID on a limited footprint.

The solution design should support scalability in all dimensions:

- Functional Scalability
- Solution Scalability
- Vertical Scalability
- Horizontal Scalability

The site assessment provides critical insight that can spell out the difference between a successful and unsuccessful RFID deployment. It will provide crucial information that helps define a wide-variety of deployment variables such as antenna direction and location, power setting, field strength and characterize the electromagnetic environment in the facilities to identify any sources of interference.

A proper site survey constitutes a physical review, analysis, recommendations and report by qualified RFID Engineers of the site where RFID infrastructure and equipment will be installed so that your RFID processes work 100% of the time.

The survey provides a complete understanding of your requirements for an RFID installation for your facility and identifies the feasibility of successfully deploying RFID technology. The engineers will determine optimal locations for the physical installation of your RFID readers and antennae, the number of unique RFID Zones you will need and document integration requirements for systems requiring integration with your RFID network.

Please note, that it is difficult to predict the propagation of radio waves and detect the presence of interfering signals without the use of specialized test equipment. RFID4U uses the latest techniques and equipment like spectrum analyzers to help ensure accurate results and save our clients the time, expense and training needed to acquire and properly use the equipment. We will even tag and read your products under actual conditions to confirm our findings.

The site survey will provide equipment quantity, placement recommendations, product brand recommendations, power considerations, and wiring requirements as they relate to your facility.

Included with each site survey should be a site survey analysis document, detailing the required location for RFID hardware, the type of cabling to be installed and an outline of how the RFID system will integrate into your existing network.

You should receive documentation with detailed recommendations regarding:

- Site Requirements,
- Estimated Project Schedule,
- Statement of Work and Best-Practice Recommendations,
- and Site Survey.

To wrap it up, there is always need for a site survey before there can be any proposals and recommendation made. RFID system is not plug and play (and never will be) and there is a lot of things involved in a proper system design and deployment that cannot be done from the comfort of the office without never visiting the site.

NFC Technology



NEAR-FIELD COMMUNICATION
MAKING LIFE EASY

Presenter: Grace Chen

What is NFC?

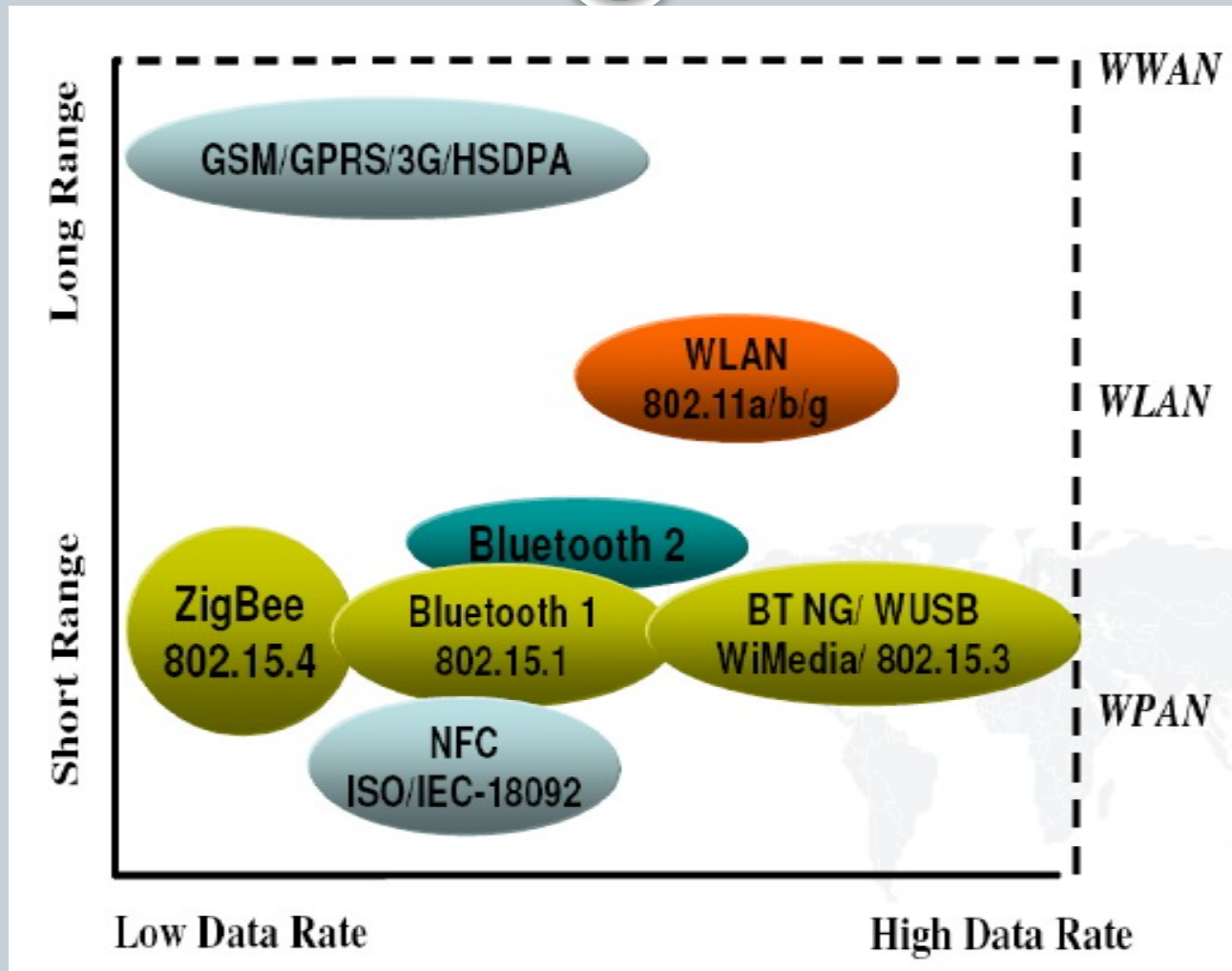


NFC = **N**ear **F**ield **C**ommunication

- A short-range wireless connectivity technology
 - Operates at 13.56 Mhz
 - Based on RFID technology
 - Data transfer rate: up to 424 kbits/second
 - Reader & Tag communicate
 - Promoted by the NFC-Forum



Comparison with other Wireless Networks



Comparison with other Wireless Networks



- Advantages:
 - Quick setup time ($<0.1\text{ms}$)
 - Short range distance (up to 10cm)
 - Does not require line of site
 - Backward compatibility
 - Consumer experience

NFC Use Cases



- **Connect Electronic Devices**
 - Exchange data
 - Simple, secure pairing e.g. Bluetooth
- **Access Information**
 - Advertisements
 - Identification Cards
- **Mobile transactions**
 - Contactless payment
 - Mobile ticketing

NFC Use Cases



**Get information
by touching
smart posters!**



**Your NFC device
is your ticket!**



**Your NFC
device is your
travel card!**

TOUCH



**Buy goods from
vending machines
with your phone!**



**Get information
about your current
job or task!**



**Your NFC device
is your credit card!**

Phone = Keys, ID card, Wallet

NFC Challenges



- **Security**
 - Eavesdropping, Data corruption
 - Solution: Establish secure channel
- **NFC ecosystem**
 - Technology already in-place
 - Monetization issues
 - ▮ Operators, Handset Manufacturers, Banks
 - Solution: Market push, agreement

Future Work



- Mobile Payment infrastructure
 - Goals: Security, Openness, Interoperability
- New NFC use cases
 - Location sensing
 - Supply Chains
 - Multiple Tags



Thank You

What is near-field communication (NFC)?

Near-field communication (NFC) is a short-range wireless connectivity technology that uses magnetic field induction to enable communication between devices when they're touched together or brought within a few centimeters of each other. This includes authenticating credit cards, enabling physical access, transferring small files and jumpstarting more capable wireless links. Broadly speaking, it builds on and extends the work of existing ecosystems and standards around radio frequency ID tags ([RFID](#)).

NFC extends RFID and contactless capabilities with more dynamic features enabled by modern smartphones. All modern phones now support NFC chips and applications, such as Apple Pay and Google Pay, to take advantage of the billions of RFID tags and terminals already deployed. NFC makes it easier to load multiple cards into a single phone for payments, municipal transit, building access, opening car doors and other use cases. NFC supports interactive applications built on basic RFID capabilities such as automatically pairing Bluetooth headphones and Wi-Fi connections. It can also automatically pull up data or an app from a poster or ad.

It was originally intended to be used to transfer files between phones using Android Beam. Modern services, such as Google Nearby Share, employ NFC to configure wireless services across faster networks like Bluetooth or Wi-Fi direct.

NFC is limited to short-range communication, which has important implications for physical access security. A user must be within 3.5 inches (10 cm) of an NFC terminal to process a payment or open a door. Another important aspect is that no power is required for the basic mechanics of listening to and responding to NFC requests. This makes it possible to implement in items that lack a battery, such as credit cards.

NFC also complements wireless technologies such as Bluetooth, Ultrawideband ([UWB](#)), Wi-Fi direct and QR codes. Its most significant advantage is that it is the easiest wireless technology for setting up a connection, which makes it [useful for IoT devices](#). However, it is not as good at maintaining a connection over distances or for long periods.

How does NFC work?

NFC works on top of three crucial innovations in wireless tag readers, cryptographic credit card processing and peer-to-peer ([P2P](#)) connectivity to enable various applications.

NFC builds on the work of the RFID set of standards and specifications, such as ISO/IEC 14443 and ISO/IEC 15963. These take advantage of a wireless communication technique using different physical principles than most wireless radios. Whereas most radios transmit data via radio wave propagation, NFC transmits data via magnetic field induction. NFC data transmits data at 13.56 MHz, which corresponds to a wavelength of 22 meters.

One critical aspect of transmitting data via induction coupling rather than radio waves is that the field fades out far more quickly than radio waves. This is useful for preventing people from listening in on sensitive conversations about credit card transactions, door access codes or other confidential information.

The second significant NFC innovation involves cryptographic credit card processing used for [contactless payments](#). [Public-key cryptography](#) allows the card to generate a new authentication code for each transaction without revealing the raw card details or three-digit code on the back. This ensures that even if someone were to listen in or a hacker queried a card on a busy subway, they would never glean the original card details.

The [NFC forum](#), a nonprofit industry association, took these two building blocks and added P2P connectivity on top of the ISO/IEC 18092 standard. Classic RFID and

credit card use cases involve an active card reader that queries a passive tag or card, which is a one-way interaction. The NFC forum introduced specifications that allowed more capable devices like smartphones, headphones, routers, home appliances and industrial equipment to initiate or react to NFC queries. This opened a wide range of interaction and connectivity patterns. It also took a lot of work to simplify the exchange of information while minimizing [security vulnerabilities](#). For example, you can tap two phones together to trade contact details using Android Beam but not accidentally swap executable code that may spread a virus.

Smartphone vendors are starting to build some basic application execution capabilities on top of this. In the Google ecosystem, a smart tag might launch a progressive web app running in the browser. Apple recently launched Apple App Clips, in which an NFC tag or QR code can launch apps with basic functionality for things like ordering in a restaurant or unlocking a rental scooter kiosk without downloading a full-blown app. These apps are limited from accessing sensitive data on the phone.

Examples of NFC

A few examples of NFC use cases include the following:

- mobile payments, such as Apple Pay and Google Pay;
- transit card payments;
- ticket redemption at a concert or theater;
- access authentication for doors or offices;
- unlocking car doors or rental scooters;
- venue or location check-in to alert friends on social media;
- device pairing smartphones and headsets by tapping them together;
- automatic setup for Wi-Fi connectivity by tapping a phone to a router or internet gateway;

- connection via smartphone to a radiator to configure its temperature settings and schedule; and
- connection via smartphone or tablet to industrial equipment to access a more complex control panel.

Benefits of NFC

NFC has several real-world benefits, including the following:

- increases operational efficiency for payment processors;
- ensures more security than traditional credit cards for payments;
- allows users to choose from multiple cards dynamically;
- difficult to intercept NFC communications from a distance;
- ease of use for consumers in paying for goods;
- simplifies access to back-end information; and
- allows simple setup of new connections compared to other wireless protocols.

Limitations of NFC

Challenges of NFC technology include the following:

- very short range of only a few inches precludes many use cases;
- slower than other protocols;
- can limit usability for apps that require sensitive data on a smartphone;
- app innovation stymied by Apple and Google restrictions and tech implementations;
- not suitable for location tracking; and
- not as universal and easy to integrate into venue ticketing apps as QR codes.

Differences between NFC and other wireless technologies

NFC complements a variety of other wireless technologies, including RFID, [EMV](#) (Europay, Mastercard and Visa), Bluetooth, UWB and QR codes. It also differs from these technologies in important ways.

NFC is best suited for authenticating transactions, unlocking doors and configuring other wireless connections. Other technologies work best in the following ways:

- RFID uses readers that can scan simple ID tags at long distances. Because it is unidirectional, this is best for reading toll tags, unlocking doors, authenticating passports or scanning inventory between more active readers and more passive tags.
- EMV allows for credit card transactions using a chip and an equipped payment terminal. While it also authenticates transactions, it is not as dynamic and interactive as NFC, which allows for contactless payments.
- Bluetooth offers a greater connection range than NFC but is less secure. It works best for connecting peripherals, such as headphones, to mobile devices and computers.
- UWB is a new technology and operates at a very low power using pulse patterns to keep from interfering with other wireless technologies. This allows it to send data quickly without losing accuracy. UWB excels in short-range location tracking, which works well for newer use cases, such as wireless car entry.
- QR codes need to be activated by the user by scanning an image with a smartphone's camera app, instead of a simple tap used by NFC. Businesses can easily generate QR codes for customers to retrieve promotions, product manuals or webpages from printed tags. QR codes are less complex and are easier to include in emails without having to rely on integration with Apple or Google specific functionality.

As far as detection range, NFC can only identify whether a device is next to another. Bluetooth can recognize when an object is within a room. UWB can locate a remote buried between couch cushions to within 10 cm. The Apple AirTag has all three kinds

of radios to use a variety of patterns of connectivity, configuration, tracking and notification.

Why mobile device management is important

In recent years, mobile devices have become ubiquitous in enterprise use. Businesses and their workforces rely on mobile devices such as smartphones, tablets and laptops for a wide assortment of tasks. And as working remotely has become essential, mobile devices have become an integral part of most organizations — vital tools for productivity and efficiency.

But because enterprise mobile devices access critical business data, they can threaten security if hacked, stolen or lost. So, the importance of managing mobile devices has evolved such that IT and security leaders are now tasked to provision, manage and secure mobile devices within their respective corporate environments.

With a [mature MDM platform](#), IT and Security departments can manage all of a company's devices, no matter their type or operating system. An effective MDM platform helps keep all devices secure while keeping the workforce flexible and productive.

How mobile device management works

A common question on the web is: "Is mobile device management a piece of software?" The short answer is "yes" and "no." MDM is a solution that uses software as a component to provision mobile devices while protecting an organization's assets, such as data. Organizations practice MDM by applying software, processes and security policies onto mobile devices and toward their use. Beyond managing device inventory and provisioning, MDM solutions protect the device's applications, data and content. In this sense, MDM and [mobile security](#) are similar. However, MDM is a device-centric approach, whereas mobile security and unified endpoint management have evolved to a user-centric stance.

In an MDM program, employees can receive a dedicated work device, such as laptops or smartphones, or have a personal device remotely enrolled. Personal devices receive role-based access to enterprise data and email, a secure VPN, GPS tracking, password-protected applications, and other MDM software for optimal data security.

MDM software can then monitor the behaviors and business-critical data on enrolled devices. And with more sophisticated MDM solutions, they can be analyzed by machine learning and AI. These tools ensure devices are kept safe from malware and other cyberthreats. For example, a firm might assign a laptop or smartphone to a staff member or consultant, pre-programmed with a data profile, VPN and the other necessary software and applications. In this scenario, MDM offers the most control to the employer. With MDM tools, enterprises can track, monitor, troubleshoot and even wipe device data in the event of theft, loss or a detected breach.

So, what are mobile device management policies? MDM policies answer questions about how organizations will manage mobile devices and govern their use. To configure and publish their policies and processes, enterprises will ask questions, such as:

- Do devices need passcode protection?
- Should cameras be disabled by default?
- Is wifi connectivity important?
- What customization options will the device provide?
- Do certain devices need to be geo-fenced?

[Click here to learn](#) about Android device management, why it's important and how it works. Also learn about Android security threats and specific vulnerabilities.

[11 Best Practices of Mobile Device Management \(MDM\) \(1.2MB\) □](#)

Components of mobile device management tools

Device tracking

Each device enrolled with or issued by an enterprise can be configured to include GPS tracking and other programs. The programs allow an enterprise's IT professionals to monitor, update and troubleshoot the device in real time. They can also detect and report high-risk or non-compliant devices and even remotely lock or wipe a device if lost or stolen.

Mobile management

IT departments procure, deploy, manage and support mobile devices for their workforce, such as troubleshooting device functionality. These departments ensure each device comes with the needed operating systems and applications for their users — including applications for productivity, security and data protection, backup and restoration.

Application security

Application security can involve app wrapping, in which an IT administrator applies security or management features to an application. Then that application is re-deployed as a containerized program. These security features can determine whether user authentication is required to open an app; whether data from the app can be copied, pasted or stored on the device; and whether the user can share a file.

Identity and access management (IAM)

Secure mobile management requires strong identity and access management (IAM). IAM allows an enterprise to manage user identities associated with a device. Each user's access within an organization can be fully regulated, using such features as single sign-on (SSO), multifactor authentication and role-based access.

[Discover the importance of IAM_](#)

Endpoint security

Endpoint security encompasses all devices that access a corporate network, including wearables, Internet of Things (IoT) sensors and non-traditional mobile devices. Endpoint security can include standard [network security](#) tools such as antivirus software and network access control and [incident response](#), URL filtering and cloud security.

BYOD and mobile device management

Bring your own device

Bring your own device ([BYOD](#)) means employees use their personal mobile devices for work instead of company-issued devices. Applying enterprise security to a personal mobile device is more challenging than simply providing such devices. But BYOD is popular, especially among younger workers. Organizations make this compromise to increase employee satisfaction and productivity. BYOD can also make the mobile workforce more affordable because it eliminates the need to purchase extra hardware.

Enterprise mobility management

[Enterprise mobility management](#) (EMM) describes a broader form of mobile device management. Going beyond the device itself, its user and its data, EMM

encompasses application and endpoint management and BYOD. EMM solutions are highly scalable, and with new security features powered by AI analytics, these solutions can offer real-time insights and alerts about thousands of behaviors and activities coming in from multiple sources at once.

Unified endpoint management

Unified endpoint management (UEM) represents the integration and evolution of MDM and EMM. It solves more challenges associated with IoT, desktop or other mobile device security. UEM solutions can help enterprises secure and control the entire IT environment and its endpoints, such as smartphones, tablets, laptops and desktops. UEM solutions can also help secure their users' personal data, apps, content and enterprise data. With an agile UEM system, enterprises can choose scalable solutions based on needs, whether those enterprise are covering a single operating system or various devices across different platforms, such as Apple iOS iPhone, Android, Microsoft Windows, macOS and Chrome OS. Mature UEM solutions are powered by machine learning and AI, which can help an enterprise's IT department make quick security decision based on real-time data and analytics.

[Why UEM Is the New MDM: The Latest Stage in Enterprise Evolution](#)

Mobile device management best practices

Whether a cloud-based or on-premises model, an MDM solutions should allow an organization to see endpoints, users and everything in between. A good mobile device management software solution will:

- Save time
- Improve efficiency
- Increase production
- Increase security
- Ease of overall mobile management system

Here are three best practices to consider in selecting an MDM solution:

Automated reports

Be sure the reporting and inventory tool consolidates all enrolled devices and associated information into easy-to-follow reports. Daily updates should be generated automatically without manual input.

Automatic updates

Beyond the advantages of instant accessibility afforded by cloud MDM, there should be no hardware to buy, install or maintain — and no associated fees. The

platform should be automatically updated with new features at a company's disposal.

Easy search

The ability to search for anything and everything is key to a cloud-based solution. An organization should be able to access its devices, integrations, reports, apps and secure documents easily.

Related Solutions

Mobile device management (MDM)

Get full visibility, manageability and security for running iOS, macOS, Android and Windows. And take advantage of seamless over-the-air (OTA) device enrollment for easy, rapid deployment.

Mobile security solutions

Whether you support a single operating system type or have a mixed variety of devices, IBM mobile security offers the most secure, productive and intuitive solution on the market. IBM harnesses the power of AI technology to help you make rapid, better-informed decisions.

Unified endpoint management (UEM)

Powered by AI and analytics and integrated with your existing IT infrastructure, IBM simplifies and accelerates the support of a diverse, complex endpoint and mobile environment. Simplify the management and security of smartphones, tablets, laptops, wearables and IoT.

Enterprise mobility management

EMM combines user, app and content management with robust data security to simplify how you manage your device environment. Get the right balance between user productivity and mobile security with IBM EMM solutions.

Bring your own device (BYOD) security

When an employee can use their personal device, you empower them to do their best work in and out of the office. BYOD programs can have the added benefit of saving the budget by shifting hardware costs to the user. But employees need to know that you're protecting their personal use and privacy. Secure your remote workforce with IBM.

Identity and access management (IAM)

An identity and access management solution is essential for securing the hybrid multicloud enterprise. Securely connect every user to the right level of access with IBM identity and access management solutions.

MOBILE SECURITY:

The estimated number of mobile devices is around 5.8 billion, which is thought to have grown exponentially within five years and is supposed to reach nearly 12 billion within four years. Hence, it will be an average of two mobile devices per person on the planet. This makes us fully dependent on mobile devices with our sensitive data being transported all over. As a result, mobile security is one of the most important concepts to take in consideration.

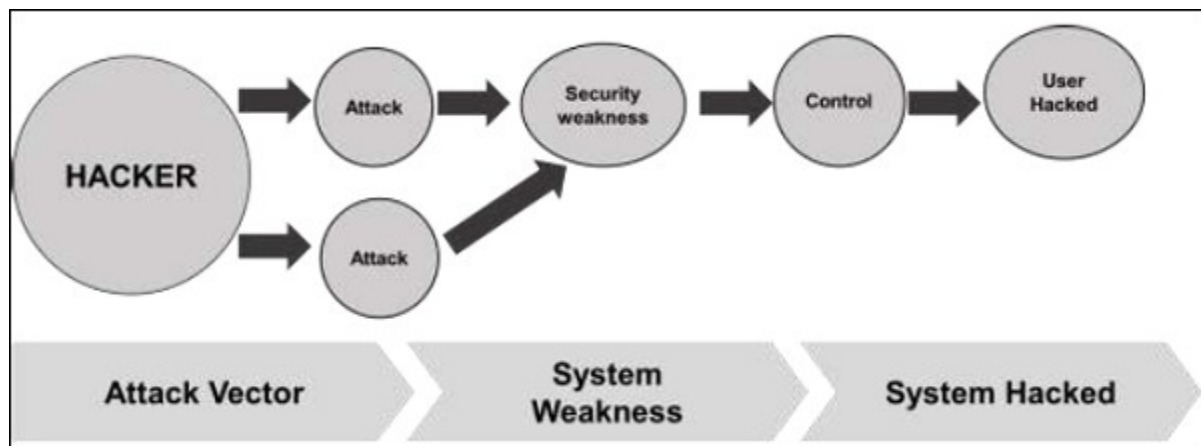
Mobile Security as a concept deals with the protection of our mobile devices from possible attacks by other mobile devices, or the wireless environment that the device is connected to.

Following are the major threats regarding mobile security –

- Loss of mobile device. This is a common issue that can put at risk not only you but even your contacts by possible phishing.
- Application hacking or breaching. This is the second most important issue. Many of us have downloaded and installed phone applications. Some of them request extra access or privileges such as access to your location, contact, browsing history for marketing purposes, but on the other hand, the site provides access to other contacts too. Other factors of concern are Trojans, viruses, etc.
- Smartphone theft is a common problem for owners of highly coveted smartphones such as iPhone or Android devices. The danger of corporate data, such as account credentials and access to email falling into the hands of a tech thief is a threat.

Mobile Security Vector Attack:

By definition, an **Attack Vector** is a method or technique that a hacker uses to gain access to another computing device or network in order to inject a “bad code” often called **payload**. This vector helps hackers to exploit system vulnerabilities. Many of these attack vectors take advantage of the human element as it is the weakest point of this system. Following is the schematic representation of the attack vectors process which can be many at the same time used by a hacker.



Some of the mobile attack vectors are –

- Malware
 - Virus and Rootkit
 - Application modification
 - OS modification
- Data Exfiltration
 - Data leaves the organization

- o Print screen
 - o Copy to USB and backup loss
- Data Tampering
 - o Modification by another application
 - o Undetected tamper attempts
 - o Jail-broken devices
- Data Loss
 - o Device loss
 - o Unauthorized device access
 - o Application vulnerabilities

Consequences of Attack Vectors

Attack vectors is the hacking process as explained and it is successful, following is the impact on your mobile devices.

- **Losing your data** – If your mobile device has been hacked, or a virus introduced, then all your stored data is lost and taken by the attacker.
- **Bad use of your mobile resources** – Which means that your network or mobile device can go in overload so you are unable to access your genuine services. In worse scenarios, to be used by the hacker to attach another machine or network.
- **Reputation loss** – In case your Facebook account or business email account is hacked, the hacker can send fake messages to your friends, business partners and other contacts. This might damage your reputation.
- **Identity theft** – There can be a case of identity theft such as photo, name, address, credit card, etc. and the same can be used for a crime.

Anatomy of a Mobile Attack

Following is a schematic representation of the anatomy of a mobile attack. It starts with the infection phase which includes attack vectors.



Infecting the device

Infecting the device with mobile spyware is performed differently for Android and iOS devices.

Android – Users are tricked to download an app from the market or from a third-party application generally by using social engineering attack. Remote infection can also be

performed through a Man-in-the-Middle (MitM) attack, where an active adversary intercepts the user's mobile communications to inject the malware.

iOS – iOS infection requires physical access to the mobile. Infecting the device can also be through exploiting a zero-day such as the JailbreakME exploit.

Installing a backdoor

To install a backdoor requires administrator privileges by rooting Android devices and jailbreaking Apple devices. Despite device manufacturers placing rooting/jailbreaking detection mechanisms, mobile spyware easily bypasses them –

Android – Rooting detection mechanisms do not apply to intentional rooting.

iOS – The jailbreaking “community” is vociferous and motivated.

Bypassing encryption mechanisms and exfiltrating information

Spyware sends mobile content such as encrypted emails and messages to the attacker servers in plain text. The spyware does not directly attack the secure container. It grabs the data at the point where the user pulls up data from the secure container in order to read it. At that stage, when the content is decrypted for the user's usage, the spyware takes controls of the content and sends it on.

How Can a Hacker Profit from a Successfully Compromised Mobile?

In most cases most of us think what can we possibly lose in case our mobile is hacked. The answer is simple - we will lose our privacy. Our device will become a surveillance system for the hacker to observe us. Other activities of profit for the hacker is to take our sensitive data, make payments, carry out illegal activities like **DDoS attacks**. Following is a schematic representation.



OWASP Mobile Top 10 Risks

When talking about mobile security, we base the vulnerability types on OWASP which is a not-for-profit charitable organization in the United States, established on April 21. OWASP is an international organization and the OWASP Foundation supports OWASP efforts around the world.

For mobile devices, OWASP has **10 vulnerability classifications**.

M1-Improper Platform Usage

This category covers the misuse of a platform feature or the failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.

M2-Insecure Data

This new category is a combination of M2 and M4 from Mobile Top Ten 2014. This covers insecure data storage and unintended data leakage.

M3-Insecure Communication

This covers poor handshaking, incorrect SSL versions, weak negotiation, clear text communication of sensitive assets, etc.

M4-Insecure Authentication

This category captures the notions of authenticating the end user or bad session management. This includes –

- Failing to identify the user at all when that should be required
- Failure to maintain the user's identity when it is required
- Weaknesses in session management

M5-Insufficient Cryptography

The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasn't done correctly.

M6-Insecure Authorization

This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.) It is distinct from authentication issues (e.g., device enrolment, user identification, etc.)

If the app does not authenticate the users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.

M7-Client Code Quality

This was the "Security Decisions Via Untrusted Inputs", one of our lesser-used categories. This would be the catch-all for code-level implementation problems in the mobile client. That's distinct from the server-side coding mistakes. This would capture

things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.

M8-Code Tampering

This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification.

Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.

M9-Reverse Engineering

This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back-end servers, cryptographic constants and ciphers, and intellectual property.

M10-Extraneous Functionality

Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

Prevention and Solutions

In order to protect ourselves from SMS phishing some rules have to be kept in mind.

- Financial companies never ask for personal or financial information, like username, password, PIN, or credit or debit card numbers via text message.
- Smishing scams attempt to create a false sense of urgency by requesting an immediate response. Keep calm and analyze the SMS.
- Don't open links in unsolicited text messages.
- Don't call a telephone number listed in an unsolicited text message. You should contact any bank, government, agency, or company identified in the text message using the information listed in your records or in official webpages.

- Don't respond to smishing messages, even to ask the sender to stop contacting you.
- Use caution when providing your mobile number or other information in response to pop-up advertisements and "free trial" offers.
- Verify the identity of the sender and take the time to ask yourself why the sender is asking for your information.
- Be cautious of text messages from unknown senders, as well as unusual text messages from senders you do know, and keep your security software and applications up to date.

Pairing Mobile Devices on Open Bluetooth and Wi-Fi Connections

Bluetooth is a similar radio-wave technology, but it is mainly designed to communicate over short distances, less than about 10m or 30ft. Typically, you might use it to download photos from a digital camera to a PC, to hook up a wireless mouse to a laptop, to link a hands-free headset to your cellphone so you can talk and drive safely at the same time, and so on.

To obtain this connection, devices exchange each other's PIN, but in general as a technology it is not secure. It is a good practice to repair the devices after a period of time.

What a hacker can do with a paired device?

- Play sounds of incoming call
- Activate alarms
- Make calls
- Press keys
- Read contacts
- Read SMS
- Turn off the phone or the network
- Change the date and time
- Change the network operator
- Delete applications

Security measures for Bluetooth devices

- Enable Bluetooth functionality only when necessary.
- Enable Bluetooth discovery only when necessary.
- Keep paired devices close together and monitor what's happening on the devices.
- Pair devices using a secure passkey.
- Never enter passkeys or PINs when unexpectedly prompted to do so.
- Regularly update and patch Bluetooth-enabled devices.

- Remove paired devices immediately after use.

Mobile Security - Android OS

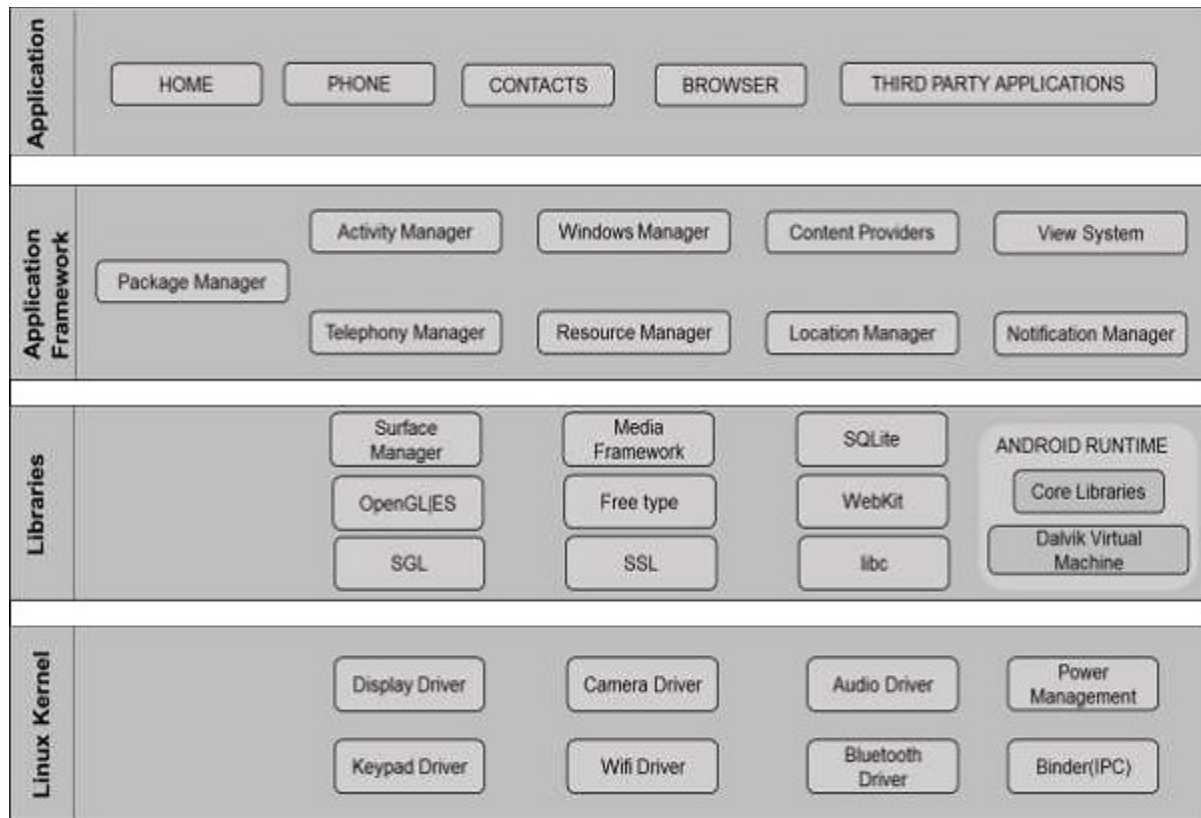
As many of us know, software is developed by Google for mobile devices with processing capabilities for smartphones and tablets. Its kernel is based on Linux. Its installed applications run in a sandbox. However, many producers have released its antiviruses for such OS, like Kasperky, McAfee, and AVG Technologies. Even though antivirus application runs under sandbox, it has a limit to scan the environment.

Some feature of android OS are as follows –

- Dalvik virtual machine optimized for mobile device
- SQLite database for structured data
- Integrated browser based on WebKit engine
- Support of different media formats like audio, images, video
- Rich development environment like emulators (Bluestack), debugging tools

Android OS Architecture

The following image shows the overall architecture of Android OS –



- **The first layer is Application**, includes applications such as SMS, calendars, and other third party applications.
- **The second layer is Application Framework**, which includes –
 - View system, which is for developers to create boxes, lines, grids, etc.
 - Content providers permit applications to access and use data from third party applications.
 - Activity manager controls the life cycle of an application.
 - Resource manager allocates resources to an application.
 - Notification manager helps to shows notifications of applications.
- **The third layer is libraries**, which is the most important part. It utilizes the function of the application, for example, to store date in a database. It is SQLite that utilizes this function.
- **The fourth layer is the Linux Kernel**. It holds all the drivers of the hardware components, such as camera, wireless, storage, etc.

Android Device Administration API

The Device Administration API introduced in Android 2.2 provides device administration features at the system level. These APIs allow developers to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices.

The device admin applications are written using the Device Administration API. These device admin applications enforce the desired policies when the user installs these applications on his or her device. The built-in applications can leverage the new APIs to improve the exchange support.

Here are some examples of the types of applications that might use the Device Administration API –

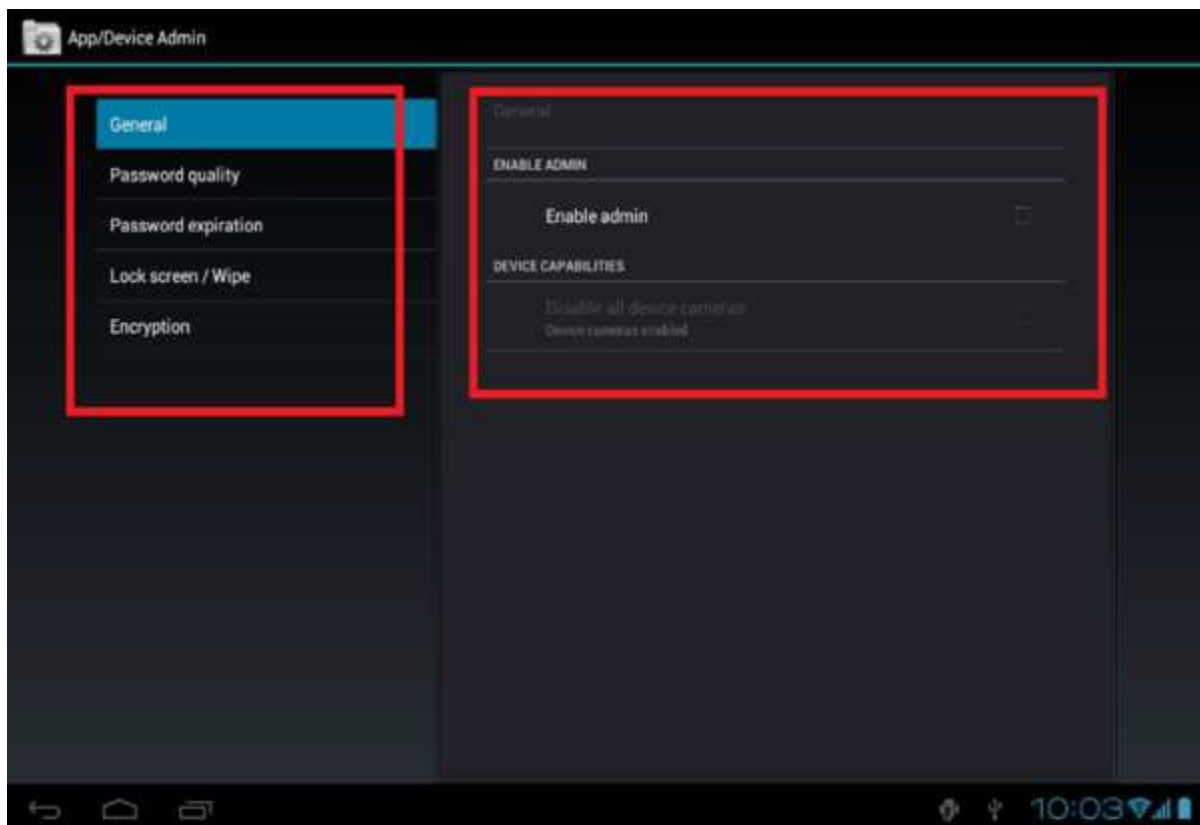
- Email clients
- Security applications that do remote wipe
- Device management services and application

The examples used in this tutorial are based on the Device Administration API sample, which is included in the SDK samples (available through the Android SDK Manager) and located on your system as

`<sdk_root>/ApiDemos/app/src/main/java/com/example/android/apis/app/DeviceAdminSample.java.`

Sample Application

This sample application offers a demo of device admin features. It presents the users with a user interface that lets them enable the device admin application.



Once the users have enabled the application, they can use the buttons in the user interface to do the following –

- Set password quality.
 - Specify requirements for the user's password, such as minimum length, the minimum number of numeric characters it must contain, and so on.
 - Set the password. If the password does not conform to the specified policies, the system returns an error.
 - Set how many failed password attempts can occur before the device is wiped (that is, restored to factory settings).
 - Set how long from now the password will expire.
 - Set the password history length (length refers to the number of old passwords stored in the history). This prevents the users from reusing one of the last passwords they previously used.
 - Specify that the storage area should be encrypted, if the device supports it.
 - Set the maximum amount of inactive time that can elapse before the device locks.
 - Make the device lock immediately.
 - Wipe device data (that is, restore factory settings).
 - Disable the camera.
-

Rooting is a word that comes from Linux syntax. It means the process which gives the users super privilege over the mobile phone. After passing and completing this process, the users can have control over SETTINGS, FEATURES, and PERFORMANCE of their phone and can even install software that is not supported by the device. In simple words, it means the users can easily alter or modify the software code on the device.

Rooting enables all the user-installed applications to run privileged commands such as –

- Modifying or deleting system files, module, firmware and kernels
- Removing carrier or manufacturer pre-installed applications
- Low-level access to the hardware that are typically unavailable to the devices in their default configuration

The advantages of rooting are –

- Improved performance
- Wi-Fi and Bluetooth tethering
- Install applications on SD card
- Better user interface and keyboard

Rooting also comes with many security and other risks to your device such as –

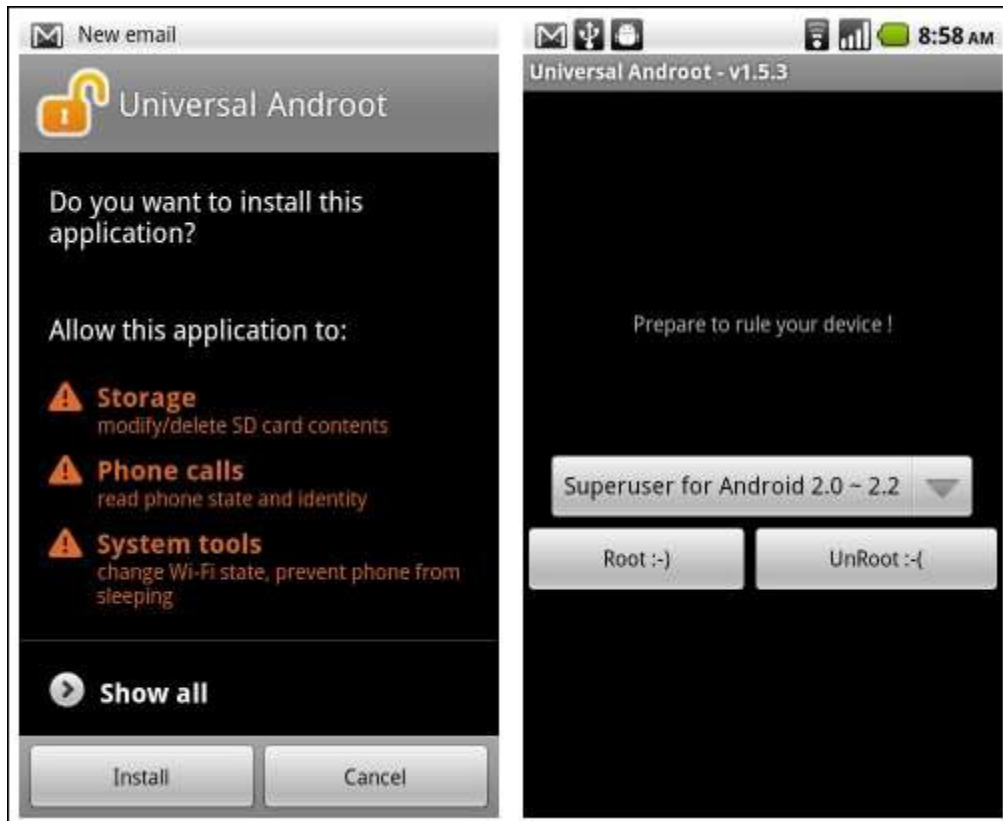
- Bricking the device
- Malware infection
- Voids your phone's warranty
- Poor performance

Android Rooting Tools

As Android OS is an open source, the rooting tools that can be found over the internet are many. However, we will be listing just some of them –

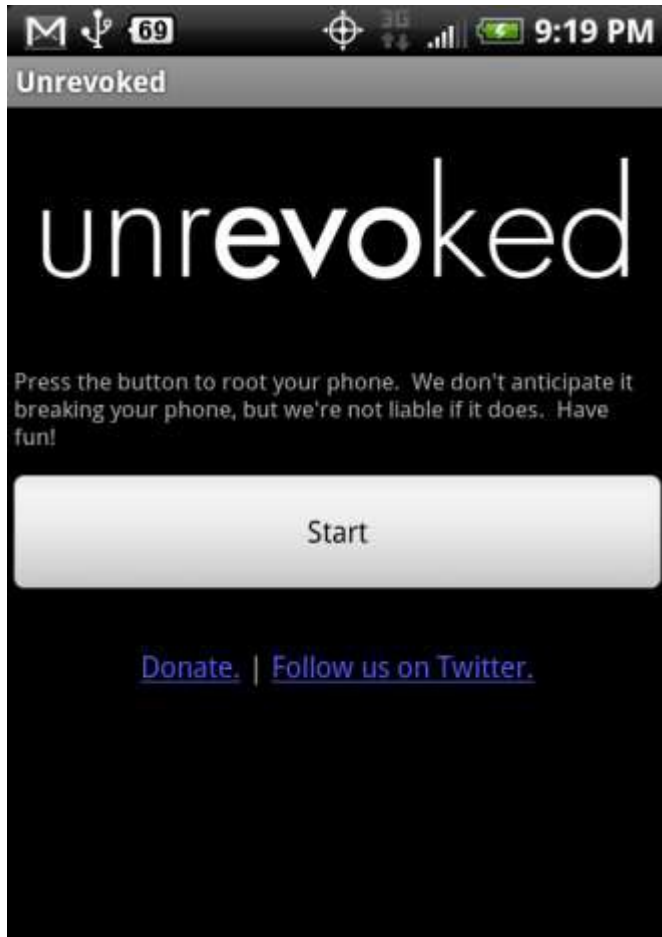
Universal Androot

You can download from <https://www.roidbay.com>



Unrevoked

Unrevoked available at <https://unrevoked.com>



Rooting Android Phones using SuperOneClick Rooting

SuperOneClick is one of the best tool designed especially for rooting an Android phone.

Let us see how to use it and root an android phone –

Step 1 – Plug in and connect your Android device to your computer with a USB cable.

Step 2 – Install the driver for the android device if prompted.

Step 3 – Unplug and re-connect, but this time select Charge only to ensure that your phone's SD card is not mounted to your PC.

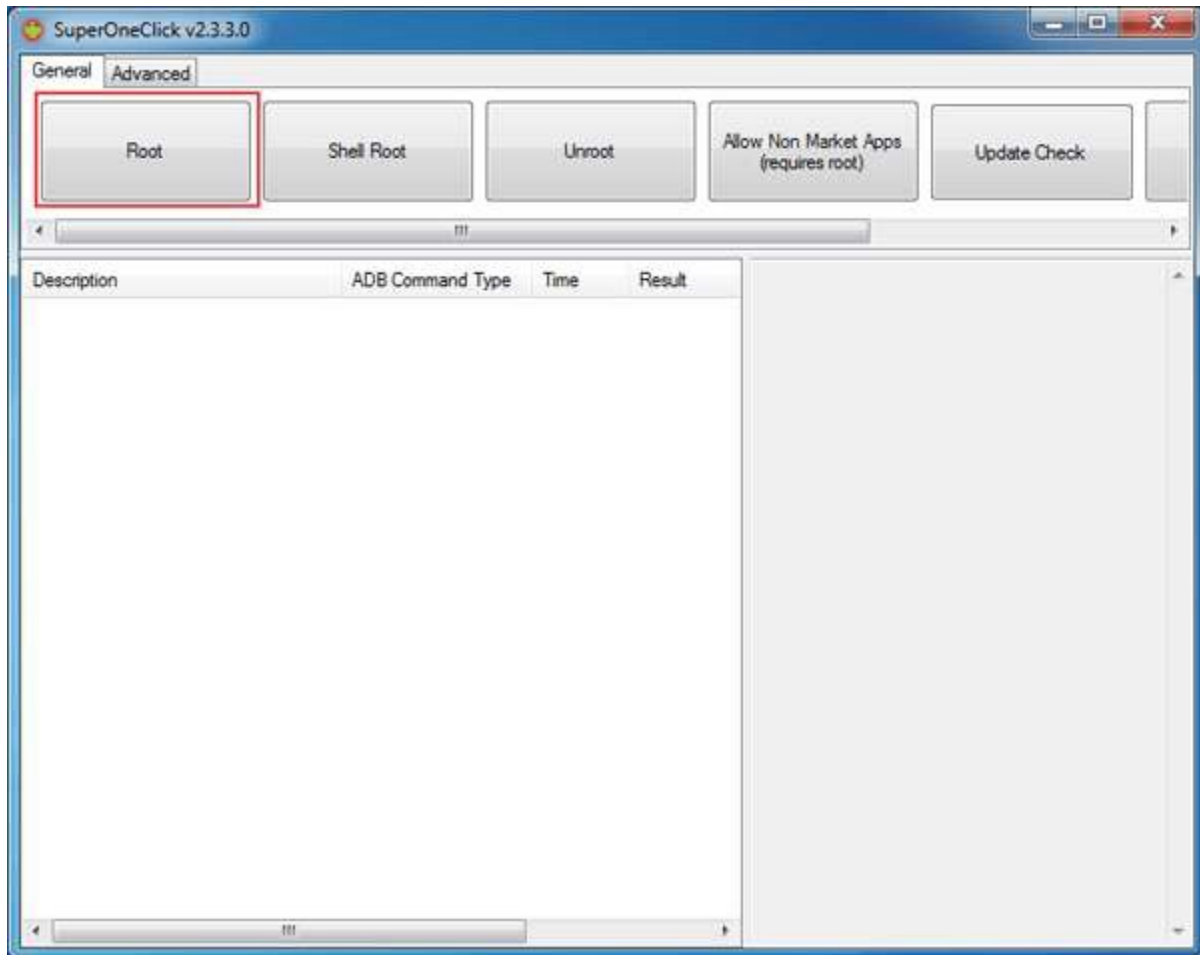


Step 4 – Go to Settings → Applications → Development and enable USB Debugging to put your android into USB Debugging mode.



Step 5 – Run SuperOneClick.exe that you have downloaded from <http://superoneclick.us/>.

Step 6 – Click the Root button.



Step 7 – Wait for some time until you see a "Running a Su test Success!"

Step 8 – Check out the installed apps in your phone.

Step 9 – Superuser icon means you now have root access.

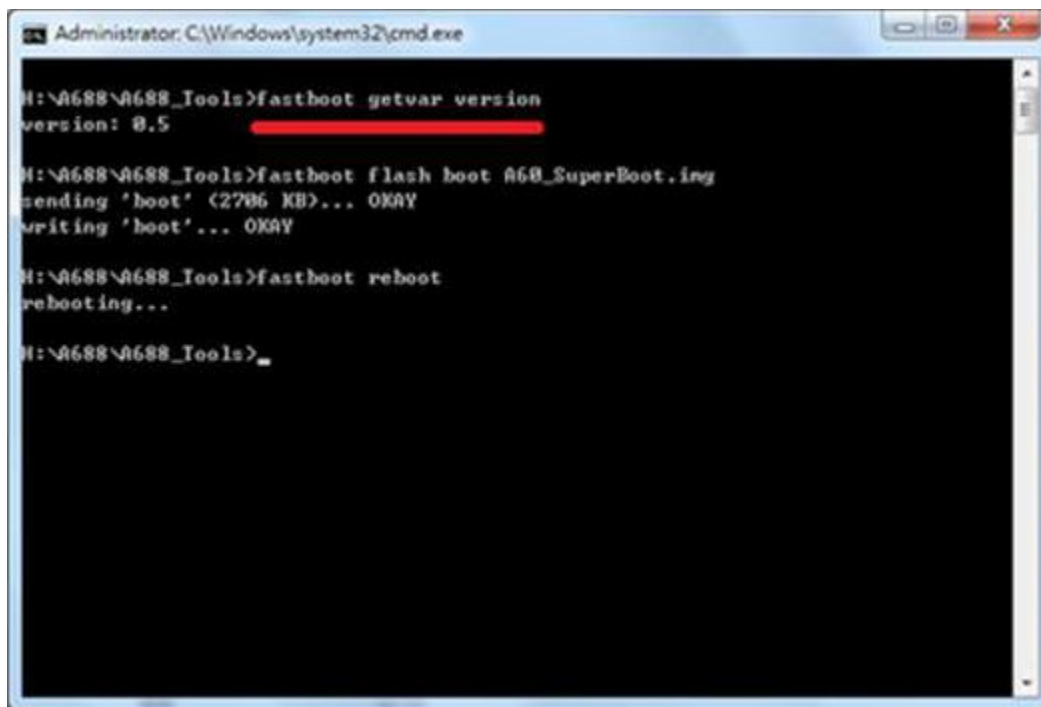
Rooting Android Phones Using Superboot

Superboot is a **boot.img**. It is designed specifically to root Android phones. It roots Android phones when they are booted for the very first time. Following are the steps

Step 1 – Download and extract the Superboot files from –

<http://loadbalancing.modaco.com>

Step 2 – Put your Android phone in the bootloader mode –



```
Administrator: C:\Windows\system32\cmd.exe

H:\A688\A688_Tools>fastboot getvar version
version: 0.5

H:\A688\A688_Tools>fastboot flash boot A68_SuperBoot.img
sending 'boot' (2706 KB)... OKAY
writing 'boot'... OKAY

H:\A688\A688_Tools>fastboot reboot
rebooting...

H:\A688\A688_Tools>
```

Step 3 – Turn off the phone, remove the battery, and plug in the USB cable.

Step 4 – When the battery icon appears on the screen, pop the battery back in.

Step 5 – Now tap the Power button while holding down the Camera key. For Android phones with a trackball: Turn off the phone, press and hold the trackball, then turn the phone back on.

Step 6 – Depending on your computer's OS, do one of the following.

- **Windows** – Double-click install-superboot-windows.bat.
- **Mac** – Open a terminal window to the directory containing the files, and type `chmod +x. Install-superboot-mac.sh` followed by `./install-superboot-mac.sh`.
- **Linux** – Open a terminal window to the directory containing the files, and type `chmod +x. Install-superboot-linux.sh` followed by `./install-superboot-linux.sh`.

Step 7 – Your Android device has been rooted.

Android Trojan

ZitMo (ZeuS-in-the-Mobile)

Zitmo refers to a version of the Zeus malware that specifically targets mobile devices. It is a malware Trojan horse designed mainly to steal online banking details from users. It circumvents mobile banking app security by simply forwarding the infected mobile's SMS messages to a command and control mobile owned by cybercriminals. The new versions of Android and BlackBerry have now added botnet-like features, such as enabling cybercriminals to control the Trojan via SMS commands.



FakeToken and TRAMP.A

FakeToken steals both authentication factors (Internet password and mTAN) directly from the mobile device.

Distribution Techniques – Through phishing emails pretending to be sent by the targeted bank. Injecting web pages from infected computers, simulating a fake security app that presumably avoids the interception of SMS messages by generating a unique digital certificate based on the phone number of the device. Injecting a phishing web page that redirects users to a website pretending to be a security vendor that offers the "eBanking SMS Guard" as protection against "SMS message interception and mobile Phone SIM card cloning".

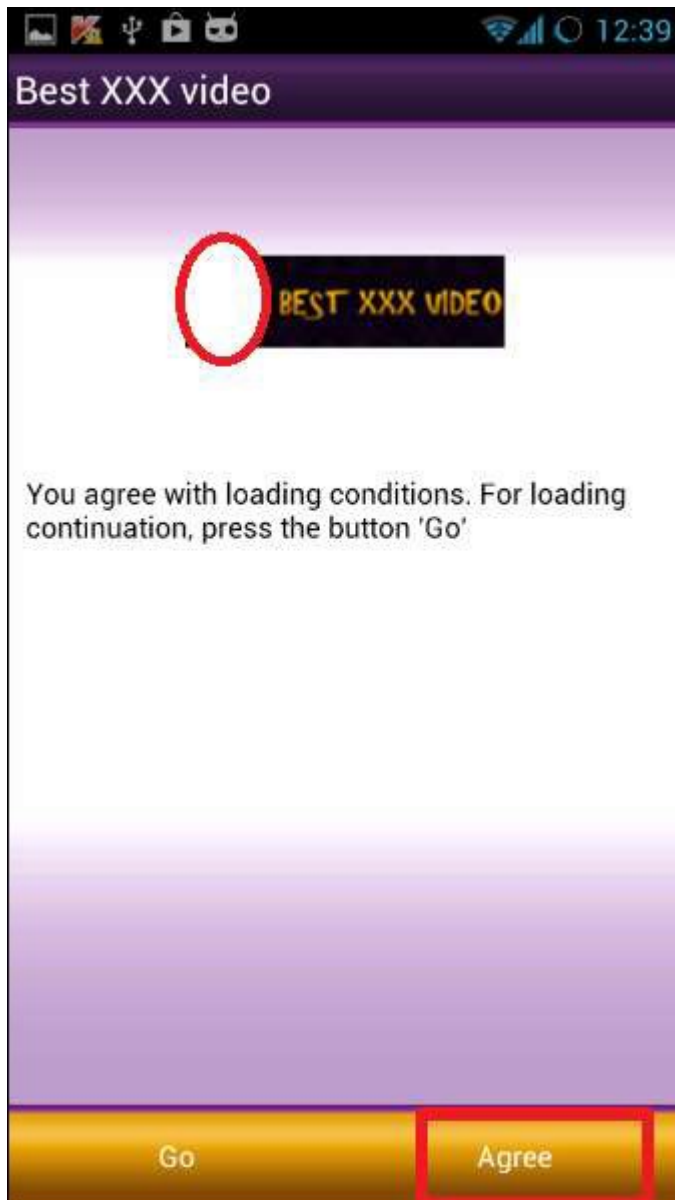
Permissions	Permissions
This application can access the following on your phone:	This application can access the following on your phone:
<ul style="list-style-type: none"> ✓ Your messages receive SMS ✓ Network communication full Internet access ✓ Your personal information read contact data ✓ Storage modify/delete SD card contents ✓ Phone calls read phone state and Identity ✓ Services that cost you money send SMS messages 	<ul style="list-style-type: none"> ✓ Your messages receive SMS ✓ Network communication full Internet access ✓ Storage modify/delete SD card contents ✓ Phone calls read phone state and Identity ✓ Services that cost you money send SMS messages

Fakedefender and Obad

Backdoor.AndroidOS.Obad.a is an Android Trojan known for its ability to perform several different functions such as, but not limited to, remotely performing commands in the console, sending SMS messages to premium-rate numbers, downloading other malware and even installing malware on an infected device just to send it to someone else through Bluetooth communication. The Backdoor.AndroidOS.Obad.a Android Trojan is a treacherous threat that disturbingly runs in the background lacking a common interface or front-end access.

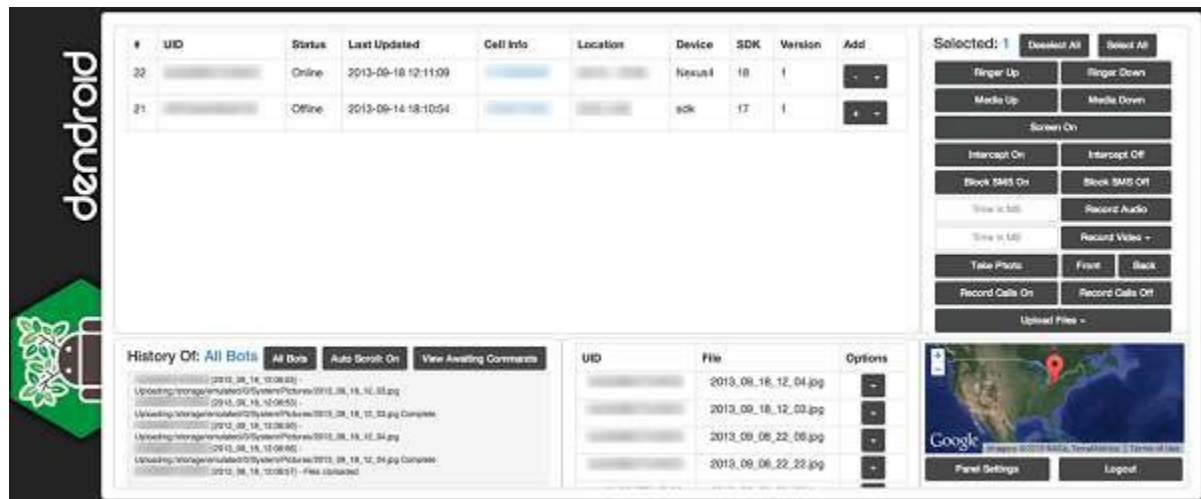
FakeInst and OpFake

Android/Fakeinst.HB is a repackaged clone of a popular, free racing game. Unlike the original, the repackaged clone requires the user to pay a charge, supposedly to "access higher game levels".



AndroRAT and Dendoroid

It is a free Android remote administration tool (RAT) known as AndroRAT (Android.Dandro) and what was believed to be the first ever malware APK binder. Since then, we have seen imitations and evolutions of such threats in the threat landscape. One such threat that is making waves in underground forums is called Dendoroid (Android.Dendoroid), which is also a word meaning - something is tree-like or has a branching structure.



Securing Android Devices:

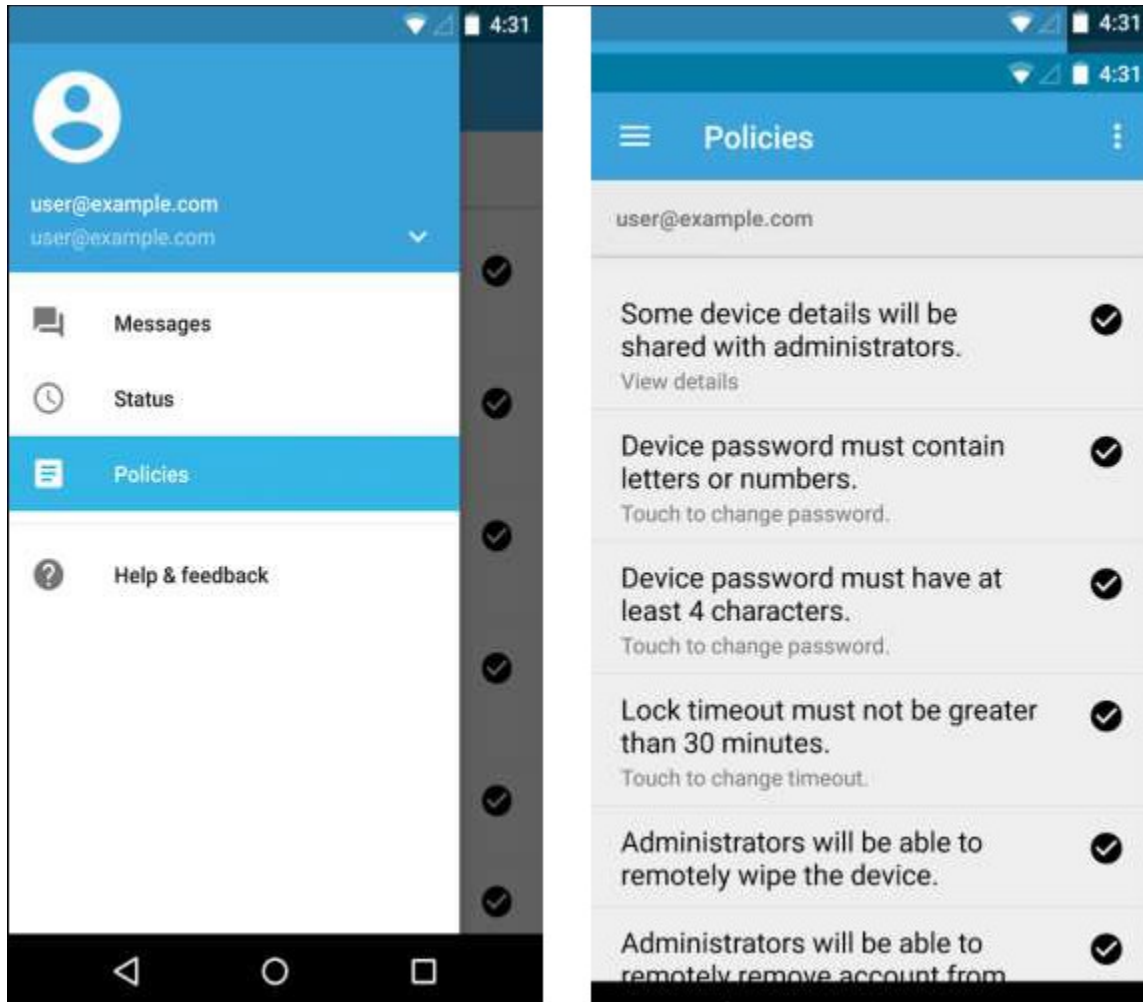
Nowadays, mobile phone devices are substituting computers in some special cases and from this comes the concern of the users and system administrators to restrict rights to the application or the user. Hence, we protect computers from being infected by installing antiviruses in order to prevent any possible unpleasant situation, where some data is lost or goes public.

Following are a few recommendations to protect our mobile devices –

- Enable lock screen so as not to be directly accessible by third parties.
- Keep the operating system updated and patch the apps all the time.
- Download apps that are officially marked by Google or from genuine sites that offers this app.
- Don't root android devices.
- Install and update antivirus app on android device.
- Don't download android package files directly.
- Use android protectors that allows you to set password to email, SMS, etc.

Google Apps Device Policy

Google Apps Device app allows a Google Apps domain admin to set security policies for android devices. This app is available only for business, government, and education accounts, which allows IT administrators remotely to push policies and enforce them. Additionally, it helps them locate mobile devices and lock them.



Remote Wipe Service

Remote Wipe Service is a service that allows administrators to reset or erase the information in the lost or stolen device. To avail this service, the device should install Google Sync or Device Policy. This can also delete all the information in the device such as mail, calendar, contacts, etc. but cannot delete the data stored on the device's SD card. When this service completes its task, it prompts the user with a message as an acknowledgement to the delete function.

Follow these steps to enable this setting for users –

Step 1 – Sign in to the Google Admin console

Step 2 – Click Device management → Mobile → Device management settings

Step 3 – Check the Allow user to remote wipe device box.

Step 4 – Click Save Changes.

You can apply this setting to your whole organization or by organizational unit to enable remote wipe for only a specific group of users.

Once enabled, a user can remotely wipe their device by following these steps –

Step 1 – Go to their **My Devices** page. The user will need to enter their password to access this page, even if they're already signed in to their account.

Step 2 – Click Wipe Device.

A window appears with this warning text: This will wipe all application and personal data from your device. Anything that hasn't been synced will be lost. Are you sure you want to proceed?

Step 3 – Click Confirm to wipe the device.

Following is the Administrator Console –

The screenshot displays the Administrator Console interface. The top navigation bar includes links for Dashboard, Organization & Users, Groups, Domain settings, Reports, Advanced tools, Setup, Support, and Settings. The left sidebar lists various services: Calendar, Chat, Contacts, Drive, Email, Chrome OS, Mobile, Postfix Services, Sites, and Video. The main content area is titled 'Mobile settings' and has tabs for Org Settings, Activation ID, and Device. The 'Device' tab is active, showing a table of managed devices. A modal window for a device named 'Xoom' is open, displaying its details and a set of action buttons. The table lists various devices with columns for Device ID, Name, Email, Model, OS, Type, and Last Sync.

Device ID	Name	Email	Model	OS	Type	Last Sync
2nd_4thrs	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Xoom	Android 3.2.1	Android	10/25/11
Appl_YVOCEP	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	iPod Touch 4	iOS 4.3	Google Sync	9/27/11
3116_6M73Z	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Xoom	Android 3.2.1	Android	10/23/11
Appl_1J5C3M2	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Google Sync	9/27/11
209A_113E3H	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Google Sync	11/9/11
309Z_105eC	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Android	10/19/11
3111_A16629	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Android	10/19/11
3116_6212db	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Android	9/23/11
3123_579643	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Android	10/27/11
313a_91702e	SyncOn ActivationOn	syncOnActivationOn@fashandash.net			Android	10/17/11
31cc_309628	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	10/5/11
31ec_72083	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	9/30/11
3299_315098	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus One	Android 2.2.1	Android	10/11/11
32a3_e71396	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	11/3/11
33a3_1a0408e	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus One	Android 2.2.1	Android	10/19/11
33a9_61239e	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	10/24/11
3422_e6272e	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	10/24/11
344a_e67097	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	9/23/11
346a_1c6d88	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus S 4G	Android 2.3.4	Android	10/5/11
3465_e6e2a8	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus One	Android 2.3.4	Android	10/25/11
34d_98b96	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus One	Android 2.3.4	Android	10/21/11
3521_80502e	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Full Android on Passion	2.3.4	Android	9/22/11
356a_1a0217	SyncOn ActivationOn	syncOnActivationOn@fashandash.net	Nexus One	Android 2.2.1	Android	10/25/11

Modal window for 'Xoom':

- Name: SyncOn ActivationOn
- Email: syncOnActivationOn@fashandash.net
- Device ID: 311669b40c3f727
- Hardware ID: 8800002017872
- Last Sync: 10/27/11 11:10 AM
- Post Sync: 10/27/11 11:17 AM

Action buttons: Block, Remote Wipe, Delete, View Details

What is Mobile Device Security?

Mobile Device Security refers to the measures designed to protect sensitive information stored on and transmitted by laptops, smartphones, tablets, wearables, and other portable devices. At the root of mobile device security is the goal of keeping unauthorized users from accessing the enterprise network. It is one aspect of a complete [enterprise security](#) plan.



A Forrester Consulting Thought Leadership Spotlight Commissioned By VMware



The VMware logo is displayed in a dark grey, sans-serif font.

Simplify Your Zero Trust Journey

For full feature access to this ebook,
please view in [Adobe Acrobat](#).

Simplify Your Zero Trust Journey



Why is Mobile Device Security important?

With more than half of business PCs now mobile, portable devices present distinct challenges to [network security](#), which must account for all of the locations and

uses that employees require of the company network. Potential threats to devices include malicious mobile apps, phishing scams, data leakage, spyware, and unsecure Wi-Fi networks. On top of that, enterprises have to account for the possibility of an employee losing a mobile device or the device being stolen. To avoid a security breach, companies should take clear, preventative steps to reduce the risk.

What are the benefits of Mobile Device Security?

Mobile device security, or mobile device management, provides the following:

- Regulatory compliance
- Security policy enforcement
- Support of “bring your own device” (BYOD)
- Remote control of device updates
- Application control
- Automated device registration
- Data backup

Above all, mobile device security protects an enterprise from unknown or malicious outsiders being able to access sensitive company data.

How does Mobile Device Security work?

Securing mobile devices requires a multi-layered approach and investment in enterprise solutions. While there are key elements to mobile device security, each organization needs to find what best fits its network.

To get started, here are some mobile security best practices:

- **Establish, share, and enforce clear policies and processes**

Mobile device rules are only as effective as a company's ability to properly communicate those policies to employees. Mobile device security should include clear rules about:

1. What devices can be used
 2. Allowed OS levels
 3. What the company can and cannot access on a personal phone
 4. Whether IT can remote wipe a device
 5. Password requirements and frequency for updating passwords
- **Password protection**

One of the most basic ways to prevent unauthorized access to a mobile device is to create a strong password, and yet weak passwords are still a persistent problem that contributes to the majority of data hacks. Another common security problem is workers using the same password for their mobile device, email, and every work-related account. It is critical that employees create strong, unique passwords (of at least eight characters) and create different passwords for different accounts.

- **Leverage biometrics**

Instead of relying on traditional methods of mobile access security, such as passwords, some companies are looking to biometrics as a safer alternative. Biometric authentication is when a computer uses measurable biological characteristics, such as face, fingerprint, voice, or iris recognition for identification and access. Multiple biometric authentication methods are now available on smartphones and are easy for workers to set up and use.

- **Avoid public Wi-Fi**

A mobile device is only as secure as the network through which it transmits data. Companies need to educate employees about the dangers of using public Wi-Fi networks, which are vulnerable to attacks from hackers who can easily breach a device, access the network, and steal data. The best defense is to encourage smart

user behavior and prohibit the use of open Wi-Fi networks, no matter the convenience.

- **Beware of apps**

Malicious apps are some of the fastest growing threats to mobile devices. When an employee unknowingly downloads one, either for work or personal reasons, it provides unauthorized access to the company's network and data. To combat this rising threat, companies have two options: instruct employees about the dangers of downloading unapproved apps, or ban employees from downloading certain apps on their phones altogether.

- **Mobile device encryption:**

Most mobile devices are bundled with a built-in encryption feature. Users need to locate this feature on their device and enter a password to encrypt their device. With this method, data is converted into a code that can only be accessed by authorized users. This is important in case of theft, and it prevents unauthorized access.

What are the different types of Mobile Device Security?

There are many aspects to a complete security plan. Common elements of a mobile security solution include the following:

- **Enterprise Mobile Management platform:** In addition to setting up internal device policies that protect against unauthorized access, it's equally important to have an Enterprise Mobile Management (EMM) platform that enables IT to gather real-time insights to catch potential threats.
- **Email security:** Email is the most popular way for hackers to spread ransomware and other malware. To combat such attacks, it's critical for businesses to be armed with advanced email security that can detect, block, and address threats faster; prevent any data loss; and protect important information in transit with end-to-end encryption.

- **Endpoint protection:** This approach protects enterprise networks that are remotely accessed by mobile devices. [Endpoint security](#) protects companies by ensuring that portable devices follow security standards and by quickly alerting security teams of detected threats before they can do damage. Endpoint protection also allows IT administrators to monitor operation functions and data backup strategies.
- **VPN:** A virtual private network, or VPN, extends a private network across a public network. This enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. VPNs' encryption technology allows remote users and branch offices to securely access corporate applications and resources.
- **Secure web gateway:** A [secure web gateway](#) protects against online security threats by enforcing company security policies and defending against phishing and malware in real-time. This is especially important for cloud security as this type of protection can identify an attack on one location and immediately stop it at other branches.
- **Cloud access security broker:** A cloud access security broker (CASB) is a tool that sits between cloud service consumers and cloud service providers to enforce security, compliance, and governance policies for cloud applications. CASBs help organizations extend the security controls of their on-premises infrastructure to the cloud.

How does Mobile Device Security complement existing application security and network security efforts?

In addition to monitoring and protecting against malicious threats to a company's data, mobile device security—when paired with an EMM platform and other network and application security solutions—enables an IT department to remotely

manage users and their devices. This capability provides security for all mobile devices connected to a network, while giving IT the option to remotely disable unauthorized users and applications. An EMM also allows IT to remotely wipe company data from a lost or stolen device and to control device updates. All of these measures enhance security significantly.

Making mobile devices secure is not a simple task, but it should be a high priority for any enterprise. To combat the growing threat of cyber-attacks, companies must continually audit their mobile security solutions and consider new security measures as they become available.

MOBILE DEVICE MANAGEMENT (MDM) EXPLAINED



Mobile device management (MDM) refers to a set of functions and features that control the use of mobile devices in compliance with organizational policies. These functions include the management of software apps, inventory, policy, security, and services for mobile and electronic devices.

How mobile device management works

The devices are managed against these functions by administrators running a backend MDM platform that enables remote control over device functions. An overlay app or software is installed on the device to enable the MDM functionality and integrate with the backend services of the corporate network such as:

- Information access
- Data transfer
- Device log sharing
- Other capabilities as needed

The MDM solution effectively reduces the risks associated with conducting sensitive business tasks on mobile devices, including [bring your own device](#) (BYOD) and corporate-owned smartphones.

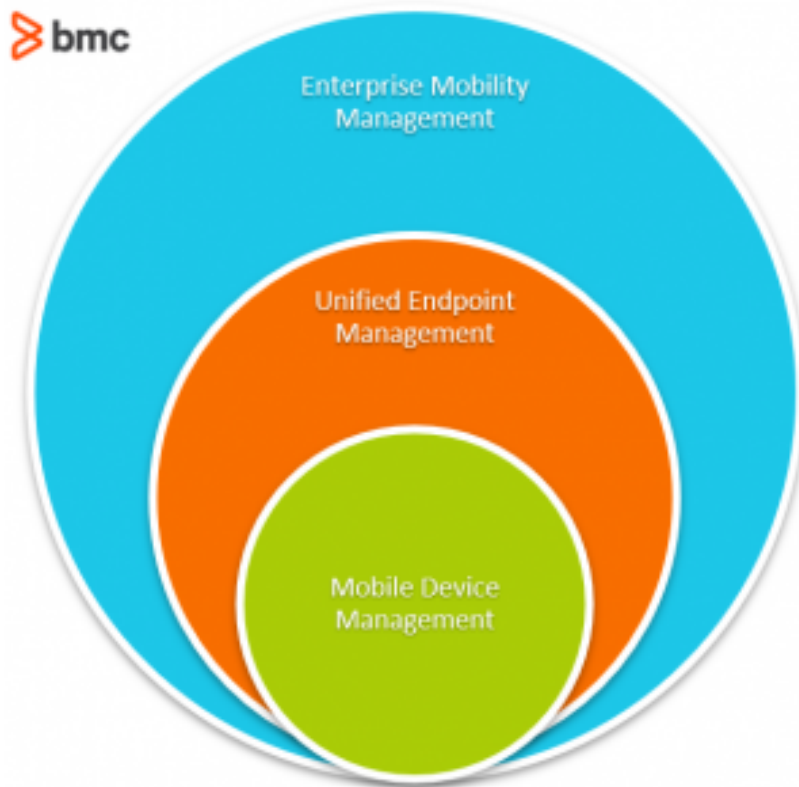
The importance of managing mobile devices

The proliferation of mobile devices and the growing BYOD trend fueled by the ongoing pandemic makes it crucial to adopt an MDM strategy. Take a look at some of the latest BYOD and enterprise mobility stats according to a recent [research](#) report:

- 67% of all employees use BYOD devices at the office.
- These devices enable users to spend an equivalent of an extra two hours daily for work related obligations.
- 87% of the employers are highly dependent on the workforce remotely accessing business information and apps on their devices.
- 69% of business decision makers support BYOD trends.
- 59% of businesses have already adopted BYOD strategies.
- The BYOD industry is expected to reach around \$367 billion by the year 2022.

MDM is often a component of [enterprise mobility management](#) (EMM) solutions, which includes a collective set of tools to secure and manage mobile apps, company-provided and BYOD devices, content, data, and access. Components within an EMM solution may have overlapping features.

For instance, an MDM solution may also offer features to manage apps and data to complement what may be extensively offered only with mobile applications management solutions.



Mastering mobility: MDM vs EMM vs UEM

The mobility management space has received a lot of attention in recent years, with enterprise IT vendors entering the market with their own flavor of device management solutions. Phrases such as Mobile Device Management, Enterprise Mobility Management, and Unified Endpoint Management (UEM) are used commonly to sell overlapping and common features.

So, before we look at the key capabilities of an MDM solution, let's [differentiate the terms](#) MDM vs

EMM vs UEM:

- **Mobile device management** (MDM) is focused on managing smartphones and mobile devices connecting to a corporate network.
- **Enterprise mobility management** (EMM) is a superset of MDM and includes many components such as application management, mobile content management, mobile security management, mobile expense management, and [identity and access management](#), among others.
- **Unified endpoint management** (UEM) consolidates the management of all endpoint devices including smartphones, [IoT devices](#), sensors, wearables, and other [endpoints](#). A single centralized platform unifies the management of all devices connecting to the network.

Capabilities of mobile device management

Switching back to MDM, we can identify the key elements of an MDM solution:

- **Asset management**, which includes multi-platform support for companies to apply custom organizational policies to enterprise mobility and BYO device use in the corporate network. [Asset management](#) might monitor and control how the devices can be used as well as enforce company policy across all enrolled devices, multiple platforms, and operating system versions.
- **Configurations management**, which can [identify, control, and manage](#) hardware and software settings based on geographic regions, user profiles, and identity.
- **Risk management**, audits, and reporting, which monitors device activity and [reports anomalous behavior](#) to limit issues such as unauthorized access of corporate networks or data transfers.
- **Software updates and distribution**, which can remotely control applications, software and OS updates, and licenses across multiple devices.
- **Profile management**, which allows management of policies and settings to specific groups of end users based on specific profiles.
- **Identity and access management**, which ensures that the device, data, network connection, and services are provided to appropriate authorized users.
- **Applications management**, which distributes, manages settings, and blocks or allows apps and software functionality.
- **Enterprise app stores**, which maintain a library of apps and services dedicated for corporate use that are available to authorized end-users.
- **Bandwidth optimization**, which manages bandwidth usage at the device and application level.
- **Data security**, which ensures that data is accessed, transferred, and utilized in accordance with organizational policies. For instance, in the event of device theft or loss, data stored on the device can be wiped out remotely.
- **Content management**, which synchronizes and secures business information across multiple devices.
- **Tech support**, which includes dedicated remote technology support can be provided remotely.

Best practices for mobile device management

The mobile device ecosystem is fragmented. Organizations constantly finding ways to enhance user productivity acknowledge the importance of BYOD devices for work, but struggle to translate

enterprise mobility into a productive workforce.

The following key best practices can help organizations adopt a risk-averse enterprise mobility strategy that also maximizes workforce productivity within the defined information security policies of your organization:

1. **Implement policies before deploying an MDM solution.** Establish the right set of policies to meet the unique technical and business needs of the organization before deploying an MDM solution.
2. **Make device enrollment to MDM solutions easy and convenient.** Ensure that no BYOD device goes under the radar, especially because of difficult or insufficient enrollment procedures or platform support.
3. **Establish self-service capabilities.** End user [self-service](#) is crucial in maintaining compliance with MDM solutions. Self-service capabilities can include remote data wipe-out, password reset, and lost device tracking.
4. **Ensure up-to-date MDM versions.** Push configuration changes, patch installations, and install software updates as soon as required and made available. A BYOD device running vulnerable outdated software is a [security incident](#) waiting to happen.
5. **Protect end-user privacy.** This will become key to ensuring end users continue compliance. Protect employee privacy by [restricting data collection](#) to a bare minimum and establishing procedures to eliminate misuse of personal employee information while still aligning with the company's technical and business needs.
6. **Deploy containment technologies.** These can separate corporate apps, data, and MDM controls from the personal use of a BYO device. With such containment in place, the MDM rules and features will only apply when the BYO device engages in corporate use.
7. **Monitor devices for specific activities or situations.** Monitor devices for anomalous activities or underoptimized data usage.

Top vendors for MDM technologies

The MDM solutions space is growing exponentially, and no individual vendor offers a one-size-fits-all solution for the enterprise market. The features span across the wide spectrum of Enterprise Mobility Management solutions, some of which may be more important to your enterprise than others.

- [IBM MaaS360](#): An end-to-end AI-driven MDM solution. Additional capabilities include Unified Endpoint Management (UEM) that uses AI capabilities to analyze threats and manage security of mobile devices across all fronts.
- [Cisco Meraki](#): A simplified platform that integrates well with the existing IT network. Granular BYOD management features that are easy to administer in a large enterprise. The attractive price point makes it a viable starting point for MDM at small and midsize business organizations.
- [Citrix Endpoint Management](#): A powerful UEM technology that includes a feature-rich MDM solution. Citrix is one of the leading mobile cybersecurity solutions providers and is known for its popular and unintrusive BYOD device management capabilities.

More solutions listed [here](#).

Related reading

- [BMC Business of IT Blog](#)
- [BMC Service Management Blog](#)
- [ITSM vs ITOM: Service Management & Operations Management Explained](#)
- [Top IT Trends Today](#)
- [What Is the Internet of Behaviors? IoB Explained](#)
- [5G for Companies: Hype, Reality & Potential](#)