

**MOBILE APPLICATION DEVELOPMENT****UNIT-1**

- 1.1 Introduction to Android, open handset alliance, Android Ecosystem
- 1.2 Need of Android
- 1.3 Features of Android
- 1.4 Tools and software required for developing an application
- 1.5 Android architecture

**UNIT-2**

- 2.1 operating system, java JDK, Android SDK
- 2.2 Android development tools
- 2.3 Android virtual devices
- 2.4 steps to install and configure Android studio and sdk

**UNIT-3**

- 3.1 control flow, directory structure
- 3.2 components of a screen
- 3.3 fundamental UI design
- 3.4 linear layout, absolute layout , table layout, relative layout
- 3.5 text view
- 3.6 edit text
- 3.7 button, image button, radio button, toggle button
- 3.8 radio group, check box, and progress bar
- 3.9 list view , grid view, image view , scroll view
- 3.10 time and date picker

**UNIT-4**

- 4.1 android platform services
- 4.2 Android system Architecture
- 4.3 Android Security model
- 4.4 Applications development: creating small application

**UNIT-5**

- 5.1 Introduction of MIT App Inventor
- 5.2 Application Coding
- 5.3 Programming Basics & Dialog
- 5.4 More Programming Basics
- 5.5 Alarm Clock Application
- 5.6 Audio & Video
- 5.7 Drawing Application
- 5.8 File
- 5.9 Game
- 5.10 Device Location
- 5.11 Web Browsing

**Text Books:**

1. Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, Wiley India Pvt Ltd, 2014.
2. App Inventor : create your own Android apps by Wolber, David (David Wayne)

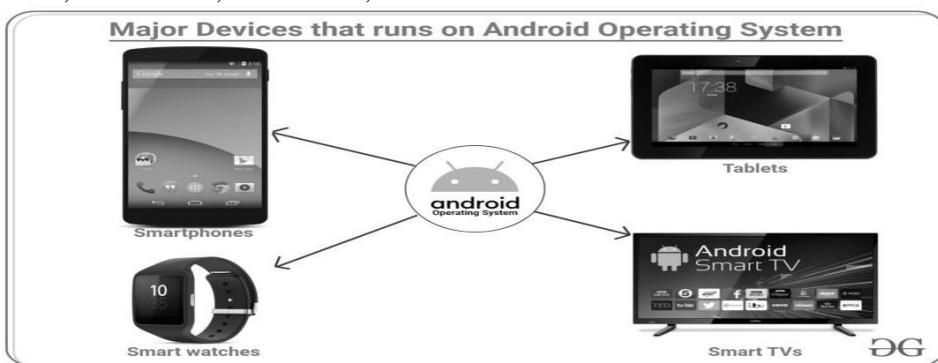
## UNIT-1

### Introduction to Android:

**Android operating system** is the largest installed base among various mobile platforms across the globe. Hundreds of millions of mobile devices are powered by **Android** in more than 190 countries of the world. It conquered around **71%** of the global market share by the end of 2021, and this trend is growing bigger every other day.

The company named **Open Handset Alliance** developed Android for the first time that is based on the modified version of the Linux kernel and other open-source software. **Google** sponsored the project at initial stages and in the year 2005, it acquired the whole company. In September 2008, the first Android- powered device was launched in the market. Android dominates the mobile OS industry because of the long list of features it provides. It's user-friendly, has huge community support, provides a greater extent of customization, and a large number of companies build Android- compatible smartphones.

As a result, the market observes a sharp increase in the demand for developing Android mobile applications, and with that companies need smart developers with the right skill set. At first, the purpose of Android was thought of as a mobile operating system. However, with the advancement of code libraries and its popularity among developers of the divergent domain, Android becomes an absolute set of software for all devices like tablets, wearables, set-top boxes, smart TVs, notebooks, etc.



Android is an open-source operating system and freely available for those who want to write code and build the operating system from it.

The android operating system is developed by Google for smartphone devices nowadays we can see android in various devices like TV, watches, and many more.

The billions of devices running an Android and become the most popular mobile operating system in the world.

Android is based on the Linux kernel and it is designed for touchscreen devices.

Android initial release date is 23 September 2008 and android written in Java, C, C++, XML, Assembly language, Python, Shell script, Go, Make, D

Official Site: [android.com](http://android.com)

### Open Handset Alliance (OHA):

The Open Handset Alliance (OHA) is a business alliance that was created for the purpose of developing open mobile device standards. The OHA is a group of more than 80 companies, including Google, HTC, Dell, Intel, Motorola, Qualcomm. OHA's main product is the Android platform - the world's most popular smartphone platform. Android is open to everyone: developers, designers, and device makers. That means more people can experiment, imagine, and create the world has never seen.

The Open Handset Alliance is an association of 84 companies developing free standards for mobile devices. Member companies include HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Google, Samsung Electronics, LG Electronics, Sprint Corporation, Nvidia.

OHA members are mainly device operators, device manufacturers, software development firms, semiconductor companies, and commercialization companies. Members share a commitment to increase the commercial viability of open platform development.

### Android ecosystem:

**Android ecosystem** is nothing but the relationship between Users, Developers/Programmers, and Hardware equipment makers, the Android ecosystem is nothing but the mutual dependence between Users, Developers, and equipment makers. they are independent of each other so one cannot exist without the other.

The main block of the android ecosystem is:

- Android User
- Developer
- Equipment Maker

### **Android User (Users buy handsets and software Application):**

Android users have more space for customizability for their android devices. Android users are smarter than other users and they are perceived to have greater levels of support. Android users are also more likely to prefer saving their cost and love the openness of the platform also they like to customize their device. Android users are fancier to prefer saving money and also android user like customizing their android handset/device

### **Developers (sell Application):**

Android Developers are the professional software developer in designing applications as well as developing applications for Android. Some of the following tasks where an android developer can play his role in the development of android apps:

Design and build advanced applications for the android platform

Collaborate and define with development teams for design and deliver new cool features.

Troubleshoot and fix bugs in new and existing applications for Users.

Evaluate and implement new development tools to work with outside data sources and APIs.

### **Equipment Maker:**

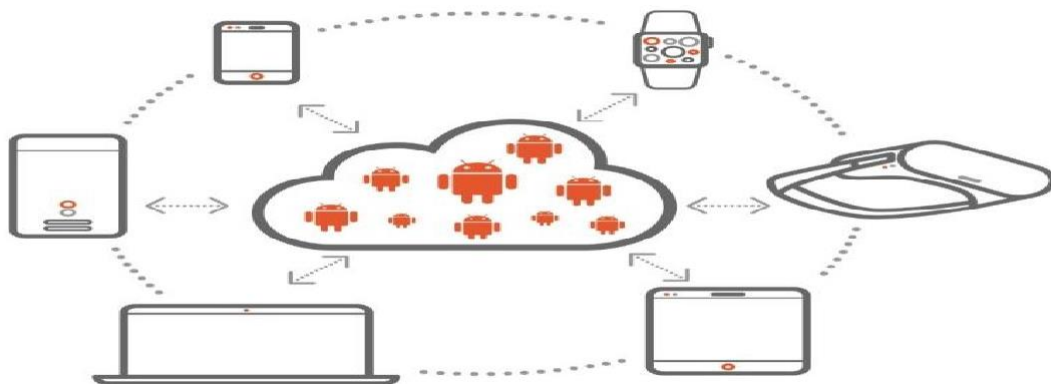
Android equipments are available in the market in a huge amount.

**Smart watches:** A smart watch is a handheld, wearable device that closely relates a wristwatch or other time device. In addition to telling time, many smart watches are wireless connectivity oriented such as Bluetooth capable. The traditional watch becomes, in effect, a wireless Bluetooth technology extending the capabilities of the wearer's smart phone to the watch.

**Smart TV:** An Android TV box is a small computer that plugs into any TV and gives the user the ability to stream content, locally and online. Apps can be downloaded from the Google Play Store, installed, and do most anything a standard computer can do from streaming videos to writing an email.

**Smart Speakers:** Smart speakers are booming in the market now, Smart speakers like Google Home, Alexa, We can control our android device via voice using these smart speakers.

**E-Reader:** E-Reader is a device used for reading e-books, digital newspapers, and other reading stuff



### **Need of Android:**



This is the era of Smartphone that includes Android and iPhone app development for every business. This leads businesses to various questions about the platform, operating system, and technology to develop a mobile application for their business. Primarily there are two popular operating systems in the market, Android and iOS. Microsoft and other companies are also providing competition to Android and iOS.

#### **The convenience of your customers:**

It is good to have a mobile application for your business. People nowadays use smartphones because smartphones are handy and less time-consuming. Smartphones offer multiple benefits like users can buy products online, compare various products, find places using Google Maps, find instant solutions, get knowledge and more. There is a great number of Android users around the world. Android applications help people communicate easily and share data and media files such as documents, images, and videos with each other.

#### **Increase Revenue and Sales:**

In today's competitive era, your business must have an effective mobile application. If you want to expand your business and are willing to increase your revenue and sales, then a mobile application could help you a great deal. Now maximum people are using smartphones, owing to the multiple utilities that smartphones offer to the users.

#### **Android app development point of view Android Studio**

Android Studio is an excellent IDE for Android app development. It is fast and efficient. Using Android Studio, Android developers can easily set up a new Android project for different types of Android apps within seconds.

#### **Following are the key features:**

- Gradle based build system
- There is an option to preview a layout on multiple screens
- Build variants and multiple APK file generation
- Enhance support for Android Wear, TV, and auto apps
- Integration with Google Cloud Platform

#### **Java**

This powerful programming language is used on a wide range of devices and operating systems. To develop an iPhone application, the iPhone developer should know objective C or Swift programming language that is used for iOS development only.

#### **Ease of accessibility**

Nowadays, owing to the popularity that the Android operating system has gained, many manufacturers opt for Android devices. Besides, customers can purchase and download various Android apps for their Android devices from the Play Store. While it takes a few hours for an Android app to be available for download, it takes a few weeks for an iOS app to be available for download in the App Store. An application can be updated multiple times in a day on the Google Play Store.

#### **Android market prediction**

Android is an open-source platform and provides a growing market, which can lead to bringing down the price of Android phones and expanding the market share of Android.

#### **Android app submission**

Submitting an Android mobile app is cheaper than submitting an iOS app. There is a one-time investment of \$25 for Android app submission, whereas a cost of \$99 per year for an iOS app. You need Apple devices such as Mac, iPhone, iPad to test an iOS app while an Android app can be tested on Android SDK.

#### **Easily port to other operating systems**

There are some portability issues with iOS as it is close sourced while Android is fully open source and can easily port to other operating systems like Ubuntu, Blackberry, Symbian, and Chrome OS.

The Android platform has a greater number of media users in each category. This is important for those are considering audience size or engagement. Google, LG, Samsung, HTC, Sony, Asus, Motorola, and many other device manufacturing companies are using Android.

#### **Features of Android**

After learning what is android, let's see the features of android. The important features of android are given below:

- It is open-source.
- Anyone can customize the Android Platform.
- There are a lot of mobile applications that can be chosen by the consumer.
- It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

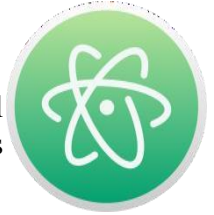
### **Tools and software required for developing an Application:**

A Software Development Tool, also known as a **Software Programming Tool**, is a computer program used by software developers to create, edit, manage, support, and debug other applications, frameworks, and programs.

Linkers, compilers, code editors, GUI designers, assemblers, debuggers, and performance analysis tools are examples of development tools. Depending on the project type, different things must be considered when picking the appropriate development tool.

#### **Atom:**

**Atom** is an open-source integrated development environment (IDE) that runs on all popular operating systems. Atom is well-known for its vast list of third-party integrations and rich level of customization.



Autocomplete is one of Atom's great attributes, as it makes writing code quicker and easier. Furthermore, its browser function simplifies project file management by enabling you to divide the Atom interface into numerous panes to view, edit, and compare files at the same time.

#### **Chrome DevTools**

Chrome DevTools is a collection of web authoring and debugging tools integrated directly into the Google Chrome browser for web developers. DevTools enables you to debug JavaScript in the browser, experiment with CSS on your website pages, and analyze the front-end effectiveness of your application.

Anyone functioning in the web domain should have this tool. **Chrome DevTools** is a set of free tools that can be accessed through the Chrome browser.



#### **Buddy:**

**Buddy** is a web developer's software development tool. To deploy, test, and develop applications, the tool makes use of delivery pipelines. The pipelines are simple to use because to a one-of-a-kind action system that allows you to combine them in whatever way you choose. When it comes to deployments, it strikes the mark. The configuration should take no more than 15 minutes.



#### **HTML5 Builder:**

HTML5 Builder, which is used to create mobile and online apps, has a lot to offer. It's adaptable and one of the quickest software development tools available.

It makes it simple to create cross-platform programs and is quite effective in collaborating. Because of the built-in features, many developers utilize it to build apps that necessitate geolocation.



#### **Azure:**

For many programmers who want to develop, administer, and build web apps, **Azure** is the way to go. It is very quick and enables a wide number of programming languages, devices, frameworks, and operating systems.

The system's capacity to identify and eliminate risks is our favorite Azure feature. It's believed to be ideal for apps that demand personal information, such as banking apps as Azure makes use of a cloud system.



#### **Android Architecture:**

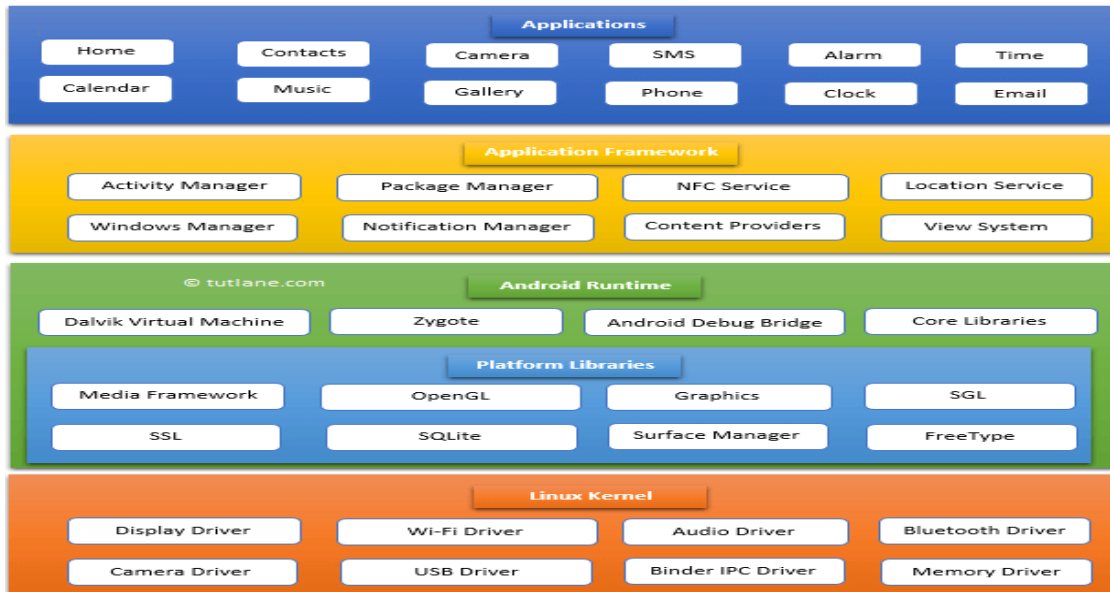
**Android architecture** is a software stack of components to support mobile device needs. Android software stack contains a Linux Kernel, collection of c/c++ libraries which are exposed through an application framework services, runtime, and application.

Following are main components of android architecture those are

- Applications
- Android Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

In these components, the **Linux Kernel** is the main component in android to provide its operating system functions to mobile and **Dalvik Virtual Machine (DVM)** which is responsible for running a mobile application.

Following is the pictorial representation of android architecture with different components.



### Applications

The top layer of the android architecture is **Applications**. The native and third-party applications like contacts, email, music, gallery, clock, games, etc. whatever we will build those will be installed on this layer only.

The application layer runs within the Android run time using the classes and services made available from the application framework.

### Application Framework

The **Application Framework** provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources. It basically provides the services through which we can create a particular class and make that class helpful for the Application creation.

The application framework includes services like telephony service, location services, notification manager, NFC service, view system, etc. which we can use for application development as per our requirements.

### Android Runtime

**Android Runtime** environment is an important part of Android rather than an internal part and it contains components like **core libraries** and the **Dalvik virtual machine**. The Android run time is the engine that powers our applications along with the libraries and it forms the basis for the application framework.

### Linux Kernel:

As with any operating system, the Linux kernel, or whatever we call it in our context, is one of the most important components of Android's architecture that resides at the root (bottom layer) of the entire system. It manages all the drivers needed during the runtime of an Android device, such as camera drivers, display drivers, audio drivers, Bluetooth drivers, and memory drivers, among others. Among the main features of the Linux kernel are as follows:

**Security:** The Linux kernel maintains the security between an application and the host system.

**Memory Management:** It efficiently manages memory, which allows us to develop our own applications without having to worry about memory allocation.

**Process Management:** It effectively manages the workflow process and allocates resources when required by processes.

**Network Stack:** It has the ability to handle network communications effectively and efficiently.

**Multitasking:** One of the main features of Linux is its support for preemptive multitasking. As a multitasking operating system with asynchronous execution, it enables multiple processes to share the same processors (CPUs) and other resources one at a time. A CPU is dedicated to performing just one task at a time.

**Dalvik Virtual Machine (DVM)** is a register-based virtual machine like Java Virtual Machine (JVM). It is specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

The **core libraries** in android runtime will enable us to implement android applications using standard JAVA programming language.

### Platform Libraries

The **Platform Libraries** includes various C/C++ core libraries and Java-based libraries such as SSL, libc, Graphics, SQLite, Webkit, Media, Surface Manger, OpenGL, etc. to provide support

for Android development.

The following are the summary details of some core android libraries available for android development.

- Media library for playing and recording audio and video formats
- The Surface manager library to provide a display management
- SGL and OpenGL Graphics libraries for 2D and 3D graphics
- SQLite is for database support and FreeType for font support
- Web-Kit for web browser support and SSL for Internet security.



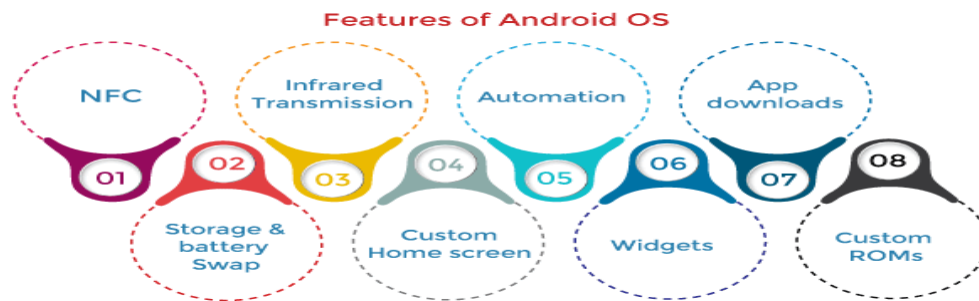
## Unit-II

### Android Operating System:

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a partnership of developers known as the Open Handset Alliance and commercially sponsored by Google. It was disclosed in November 2007, with the first commercial Android device, the HTC Dream, launched in September 2008.

It is free and open-source software. Its source code is Android Open Source Project (AOSP), primarily licensed under the Apache License. However, most Android devices dispatch with additional proprietary software pre-installed, mainly Google Mobile Services (GMS), including core apps such as Google Chrome, the digital distribution platform Google Play and the associated Google Play Services development platform.

### Characteristics of the android operating system:



### Near Field Communication (NFC)

Most Android devices support NFC, which allows electronic devices to interact across short distances easily. The main goal here is to create a payment option that is simpler than carrying cash or credit cards, and while the market hasn't exploded as many experts had predicted, there may be an alternative in the works, in the form of Bluetooth Low Energy (BLE).

### Infrared Transmission

The Android operating system supports a built-in infrared transmitter that allows you to use your phone or tablet as a remote control.

### Automation:

The Tasker app allows control of app permissions and also automates them.

### Wireless App Downloads:

You can download apps on your PC by using the Android Market or third-party options like AppBrain. Then it automatically syncs them to your Droid, and no plugging is required.

### Storage and Battery Swap:

Android phones also have unique hardware capabilities. Google's OS makes it possible to upgrade, replace, and remove your battery that no longer holds a charge. In addition, Android phones come with SD card slots for expandable storage.

### Custom Home Screens:

While it's possible to hack certain phones to customize the home screen, Android comes with this capability from the get-go. Download a third-party launcher like *Apex*, *Nova*, and you can add gestures, new shortcuts, or even performance enhancements for older-model devices.

### Widgets:

Apps are versatile, but sometimes you want information at a glance instead of having to open an app and wait for it to load. Android widgets let you display just about any feature you choose on the home screen, including weather apps, music widgets, or productivity tools that helpfully remind you of upcoming meetings or approaching deadlines.

### Custom ROMs:

Because the Android operating system is open-source, developers can twist the current OS and build their versions, which users can download and install in place of the stock OS. Some are filled with features, while others change the look and feel of a device. Chances are, if there's a feature you want, someone has already built a custom ROM for it.

### Java JDK:

The Java Development Kit (JDK) is one of three core technology packages used in Java programming, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment). It's important to differentiate between these three technologies, as well as understanding how they're connected:

The JVM is the Java platform component that executes programs.

The JRE is the on-disk part of Java that creates the JVM.



The JDK allows developers to create Java programs that can be executed and run by the JVM and JRE.

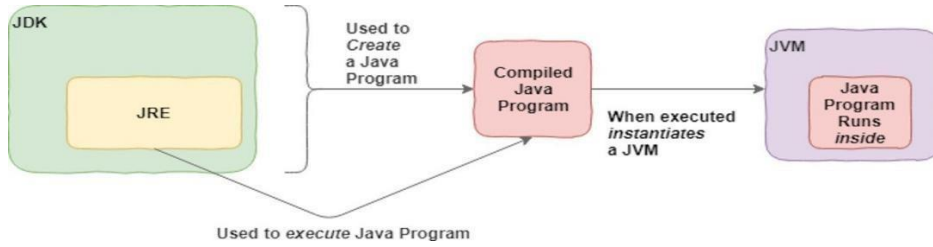


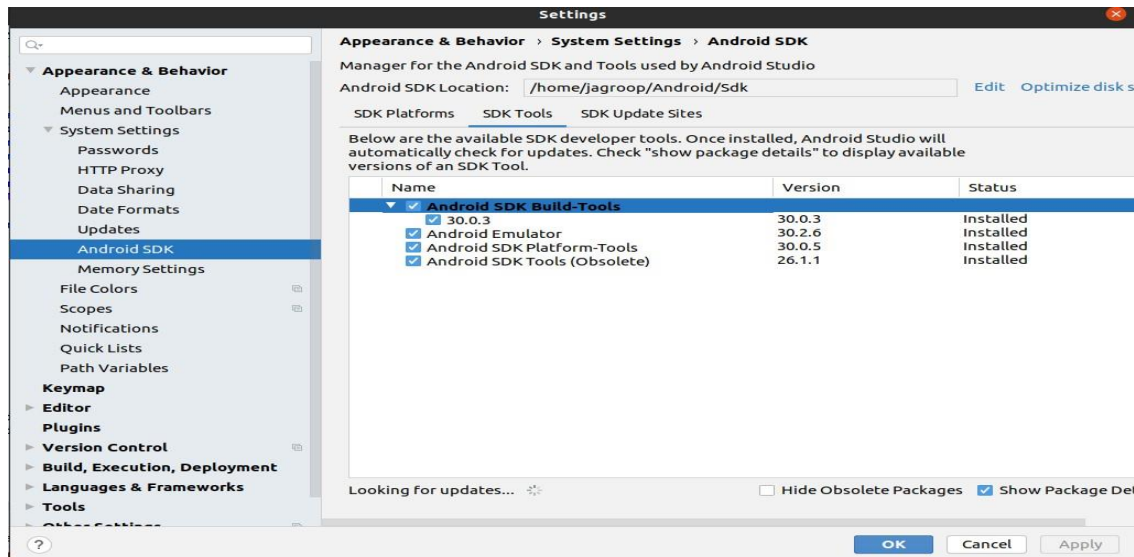
Figure 1. High-level view of the JDK

**The JDK & the Java compiler:**

In addition to the JRE, which is the environment used to run Java applications, every JDK contains a Java compiler. The *compiler* is the software program capable of taking raw .java files--which are plain text--and rendering them into executable .class files. We'll see the compiler in action soon. First, I'll show you how to download and setup a JDK in your development environment.

**Android SDK:**

Android SDK stands for Android Software Development Kit which is developed by Google for Android Platform. With the help of Android SDK, we can create android Apps easily. Android SDK is a collection of libraries and Software Development tools that are essential for Developing Android Applications. Whenever Google releases a new version or update of Android Software, a corresponding SDK also releases with it. In the updated or new version of SDK, some more features are included which are not present in the previous version. Android SDK consists of some tools which are very essential for the development of Android Application. These tools provide a smooth flow of the development process from developing and debugging. Android SDK is compatible with all operating systems such as Windows, Linux, macOS, etc.

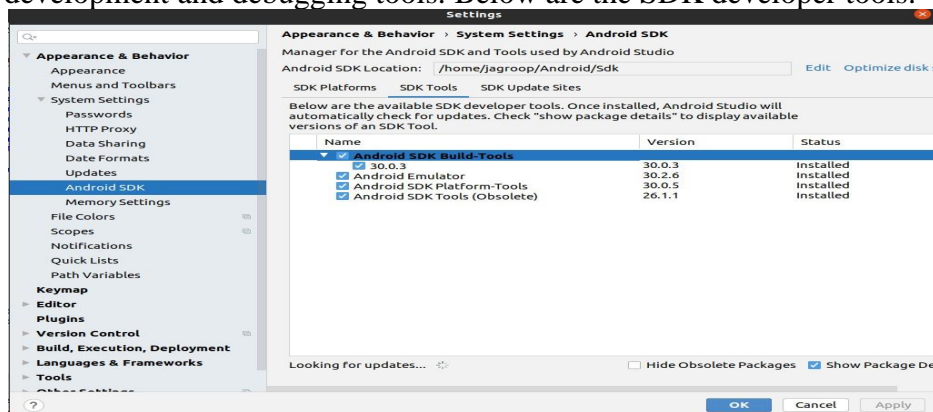


**Components of Android SDK:**

Android SDK Components play a major role in the Development of Android applications. Below are the important components:

**Android SDK Tools**

Android SDK tool is an important component of Android SDK. It consists of a complete set of development and debugging tools. Below are the SDK developer tools:



### **Android SDK Build-Tools**

Android SDK build tools are used for building actual binaries of Android App. The main functions of Android SDK Build tools are built, debug, run and test Android applications. The latest version of the Android SDK Build tool is 30.0.3. While downloading or updating Android in our System, one must ensure that its latest version is download in SDK Components.

### **Android Emulator**

An Android Emulator is a device that simulates an Android device on your system. Suppose we want to run our android application that we code. One option is that we will run this on our Android Mobile by Enabling USB Debugging on our mobile. Another option is using Android Emulator. In Android Emulator the virtual android device is shown on our system on which we run the Android application that we code.



In Android Virtual Emulator all functions that are feasible on real Android mobile is works on virtual Device like:

- phone calls, text messages.
- simulate different network speeds.
- specify the location of a device
- access on google play store and lot's more.

### **Android SDK Platform-tools:**

Android SDK Platform-tools is helpful when we are working on Project and they will show the error messages at the same time. It is specifically used for testing. It includes:

- Android Debug Bridge (ADB), is a command-line tool that helps to communicate with the device. It allows us to perform an action such as Installing App and Debugging App etc.
- Fastboot allows you to flash a device with a new system image.
- Systrace tools help to collect and inspect timing information. It is very crucial for App Debugging.

### **Android SDK Tools:**

Android SDK tool is a component of SDK tool. It consists of a set of tools which and other Utilities which are crucial for the development of Android Application. It contains the complete set of Debugging and Development tools for android.

### **SDK Platforms:**

For Each Android Software, one SDK platform is available as shown below:

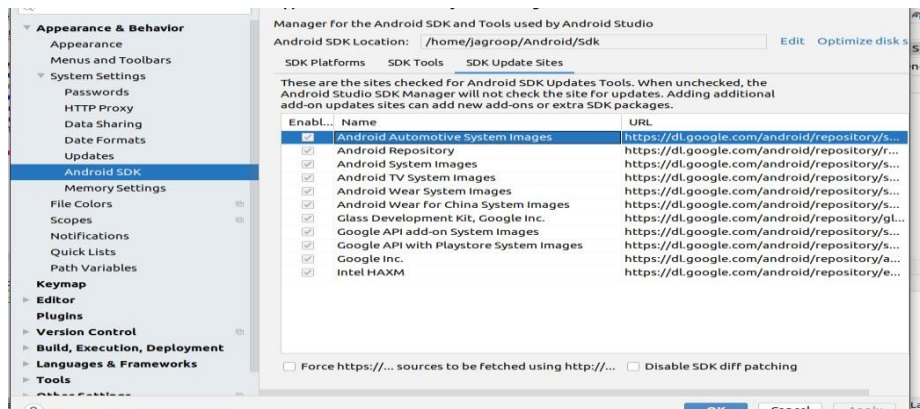
Name	API Level	Revision	Status
<input checked="" type="checkbox"/> Android 11.0 (R)	30	3	Installed
<input type="checkbox"/> Android 10.0 (Q)	29	5	Not installed
<input type="checkbox"/> Android 9.0 (Pie)	28	6	Not installed
<input type="checkbox"/> Android 8.1 (Oreo)	27	3	Not installed
<input type="checkbox"/> Android 8.0 (Oreo)	26	2	Not installed
<input type="checkbox"/> Android 7.1.1 (Nougat)	25	3	Not installed
<input type="checkbox"/> Android 7.0 (Nougat)	24	2	Not installed
<input type="checkbox"/> Android 6.0 (Marshmallow)	23	3	Not installed
<input type="checkbox"/> Android 5.1 (Lollipop)	22	2	Not installed
<input type="checkbox"/> Android 5.0 (Lollipop)	21	2	Not installed
<input type="checkbox"/> Android 4.4w (KitKat Wear)	20	2	Not installed
<input type="checkbox"/> Android 4.4 (KitKat)	19	4	Not installed
<input type="checkbox"/> Android 4.3 (Jelly Bean)	18	3	Not installed
<input type="checkbox"/> Android 4.2 (Jelly Bean)	17	3	Not installed
<input type="checkbox"/> Android 4.1 (Jelly Bean)	16	5	Not installed

Like in this Android 11.0(R) is installed.

These are numbered according to the android version. The new version of the SDK platform has more features and more compatible but the old version is less compatible with fewer features. Like in Android 11.0(R) have more compatible and have more feature but the below versions like Android 10.0(Q), Android4.4(KitKat) have less feature and is less compatible.

### **SDK Update Sites:**

In SDK Update Sites, some sites are embedded in it which will check for Android SDK Updates Tools. In this, one must ensure we don't unclick the button below because these are checked by default which will check for updates if we will unclick it then it doesn't check updates for those.



### **Android development tools:**

Android continues to be the most used operating system worldwide. And that means Android apps are extremely popular. Most companies who build mobile apps, create apps for multiple devices – both Android devices and iOS devices. To build an excellent Android app you need to use the best tools. Here's out top picks when it comes to Android development tools:

#### **Android Studio**

There's no talking about android app development without the Android Studio. It's the most basic tool for Android developers. Created by Google in 2013, it has pretty much become the standard software for Android Developers. It's a great tool because it has the support of Google as well as a large community of developers.

#### **Android Debug Bridge (ADB)**

Android Debug Bridge is included in Android Studio and it's basically a line of communication between Android devices and other computers that developers use for QA and testing purposes. Android Developers can connect their Android device to their computer and make necessary changes to both devices at the same time.

#### **Android Virtual Device (AVD) Manager**

Another great feature of Android Studio is the AVD. This is an emulator that will run your Android app on your computer so that you have a better inside into what your code looks like. It's great for actually seeing the work you've done and making any adjustments as needed.

#### **Eclipse**

Next up on the list we have Eclipse. Before Android Studio came around, Eclipse was the main tool for Android development. As of right now, Google doesn't support this software but some developers still use it to build Android apps as well as other apps. Eclipse is still a pretty useful tool, especially for developing cross-platform applications, and it supports a variety of programming languages.

#### **Fabric**

Up next we have Fabric. Many big companies have used it while developing their mobile apps, for example Twitter, Uber and Spotify. That in of itself is a big test for a tool like this. Google actually purchased Fabric from Twitter in January 2017. The platform offers several kits for developers to use during testing as well as kits for marketing and advertising. Those components make it really easy to ensure your application is user-friendly and fits into the target market.

#### **FlowUp**

When working on any web or mobile project, performance is always key. And Flow Up is a perfect tool for developers to check the performance of their Android app or any other app as well. It's a SaaS (Software as a Service) solution that you can use based on a monthly subscription. The system shows you a nice, organized dashboard of all the key metrics for your application like CPU, bandwidth, disk usage etc.

#### **GameMaker: Studio**

GameMaker: Studio is a great option for people coding their first Android game. If you're just starting out with Java and Android and want to learn how to create a game, this is the perfect option. The platform provides you with everything you need to create a 2D game with very little programming and code. It also has a drag-and-drop interface that makes it very easy for beginner developers to start their Android development journey.

#### **Genymotion**

Genymotion is an emulator that lets you view more than 3000 different device scenarios so that you can test your app in many different environments. Besides Android and Java, this tool also supports other programming languages and operating systems.

#### **Instabug**

The next tool on the list is a great testing and bug reporting system. Companies like Yahoo, BuzzFeed, Lyft and PayPal use it in their development process. It enables each developer to

document bugs, add screen shots and share this with other developers on the team to keep a log of all the bugs.

### **Visual Studio With Xamarin**

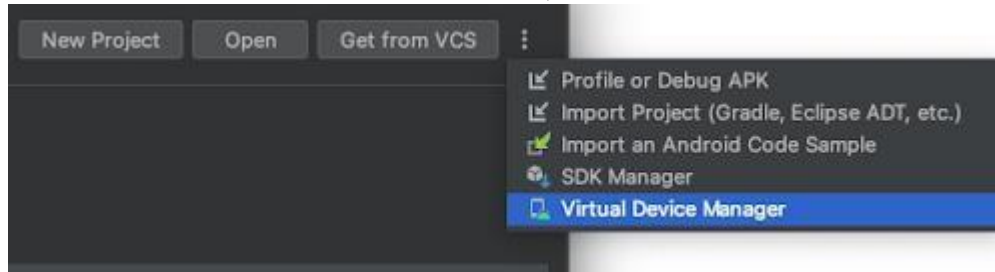
To end this list we have another classic tool, already known to most people. Visual Studio is Microsoft's original development environment. You can use almost any programming language with it and use it to make native Android, iOS and Windows apps when you combine it with Xamarin.

### **Virtual devices:**

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the [Android Emulator](#). The Device Manager is an interface you can launch from Android Studio that helps you create and manage AVDs.

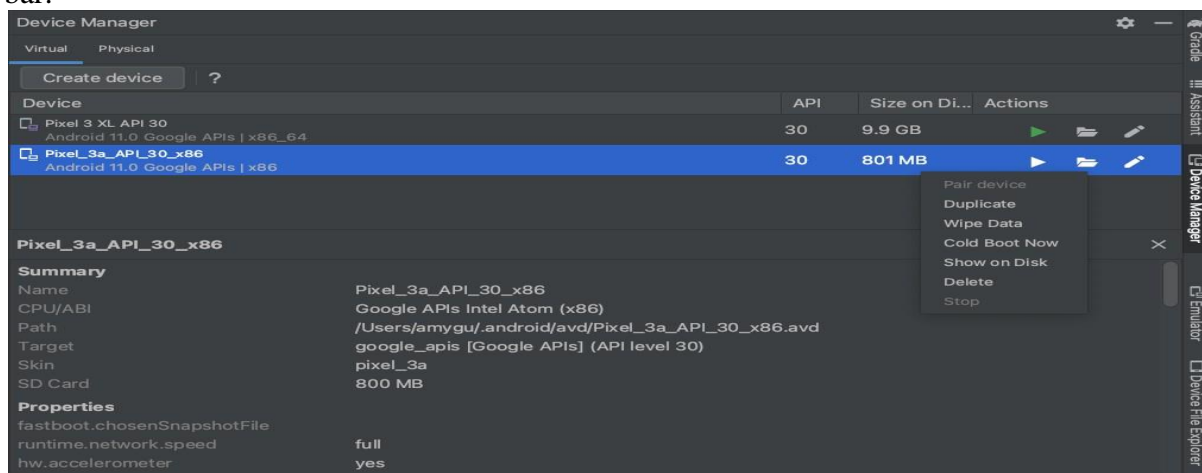
To open the new **Device Manager**, do one of the following:

From the Android Studio Welcome screen, select **More Actions > Virtual Device Manager**.



**Note:** You can currently create and manage only virtual devices from the welcome screen.

After opening a project, select **View > Tool Windows > Device Manager** from the main menu bar.



### **About AVDs**

An AVD contains a hardware profile, system image, storage area, skin, and other properties.

We recommend that you create an AVD for each system image that your app could potentially support based on the `<uses-sdk>` setting in your manifest.

### **Hardware profile**

The hardware profile defines the characteristics of a device as shipped from the factory. The Device Manager comes preloaded with certain hardware profiles, such as Pixel devices, and you can define or customize the hardware profiles as needed.

Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully **CTS** compliant and may use system images that include the Play Store app.

### **System images**

A system image labeled with **Google APIs** includes access to [Google Play services](#). A system image labeled with the Google Play logo in the **Play Store** column includes the Google Play Store app *and* access to Google Play services, including a **Google Play** tab in the **Extended controls** dialog that provides a convenient button for updating Google Play services on the device.

### **Storage area**

The AVD has a dedicated storage area on your development machine. It stores the device user data, such as installed apps and settings, as well as an emulated SD card. If needed, you can use the Device Manager to wipe user data, so the device has the same data as if it were new.

### **Skin**

An emulator skin specifies the appearance of a device. The Device Manager provides some predefined skins. You can also define your own, or use skins provided by third parties.



**AVD and app features**

Be sure your AVD definition includes the device features your app depends on. See [Hardware Profile Properties](#) and [AVD Properties](#) for lists of features you can define in your AVDs.

**Create an AVD**

Tip: If you want to launch your app into an emulator, instead [run your app from Android Studio](#) and then in the Select Deployment Target dialog that appears, click **Create New Virtual Device**.

**Android Studio** is the official **IDE (Integrated Development Environment)** for Android app development and it is based on **JetBrains' IntelliJ IDEA** software. Android Studio provides many excellent features that enhance productivity when building Android apps, such as:

A blended environment where one can develop for all Android devices

Apply Changes to push code and resource changes to the running app without restarting the app

A flexible Gradle-based build system

A fast and feature-rich emulator

GitHub and Code template integration to assist you to develop common app features and import sample code

Extensive testing tools and frameworks

C++ and NDK support

Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine, and many more.

**System Requirements**

Microsoft Windows 7/8/10 (32-bit or 64-bit)

4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

**Installation Guide**

**Step 1:** Head over to [this link](#) to get the Android Studio executable or zip file.

**Step 2:** Click on the **Download Android Studio** Button.



Android Studio provides the fastest tools for building apps on every type of Android device.

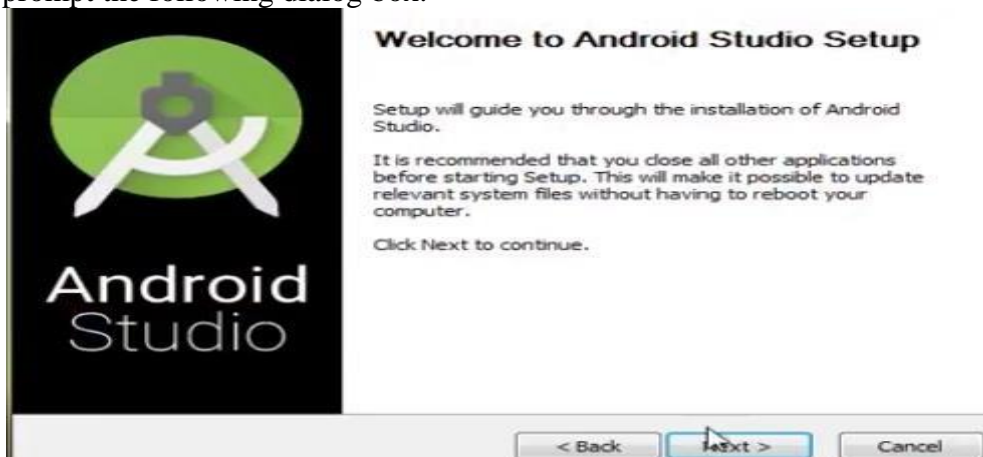
DOWNLOAD ANDROID STUDIO

4.1.3 for Windows 64-bit (896 MiB)

Click on the “I have read and agree with the above terms and conditions” checkbox followed by the download button.

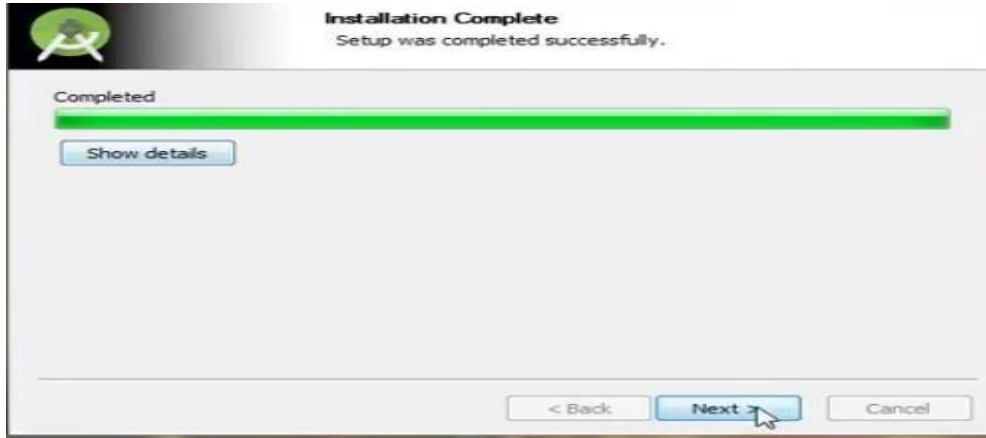
Click on the Save file button in the appeared prompt box and the file will start downloading.

**Step 3:** After the downloading has finished, open the file from downloads and run it. It will prompt the following dialog box.

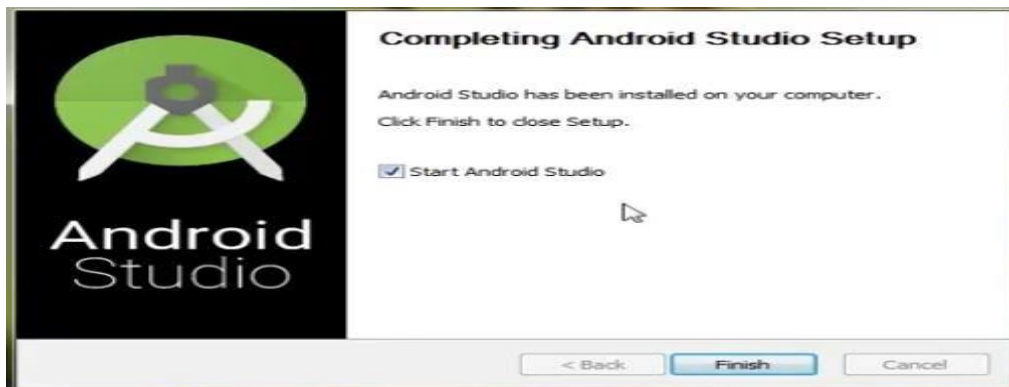


Click on next. In the next prompt, it'll ask for a path for installation. Choose a path and hit next.

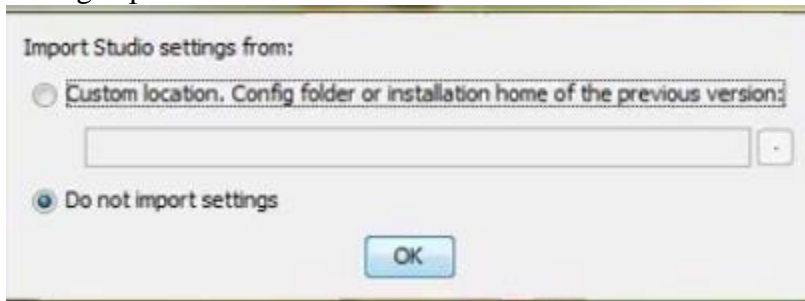
**Step 4:** It will start the installation, and once it is completed, it will be like the image shown below.



Click on next.



**Step 5:** Once “**Finish**” is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the ‘Don’t import Settings option’.

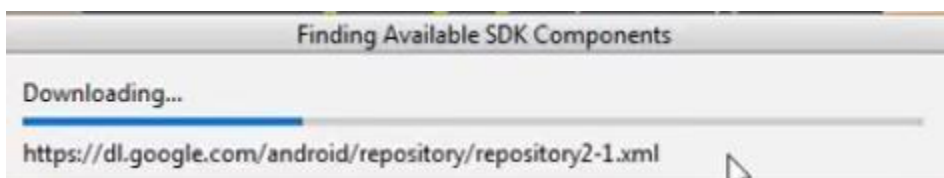


Click the **OK** button.

**Step 6:** This will start the Android Studio.



Meanwhile, it will be finding the available SDK components.

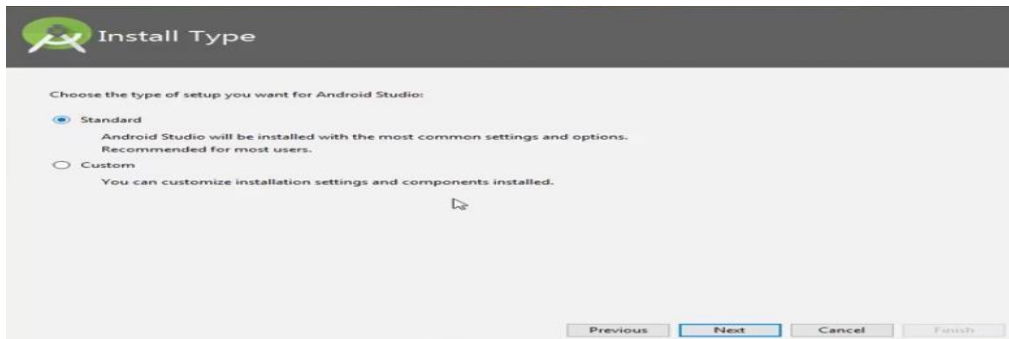


**Step 7:** After it has found the SDK components, it will redirect to the Welcome dialog box.

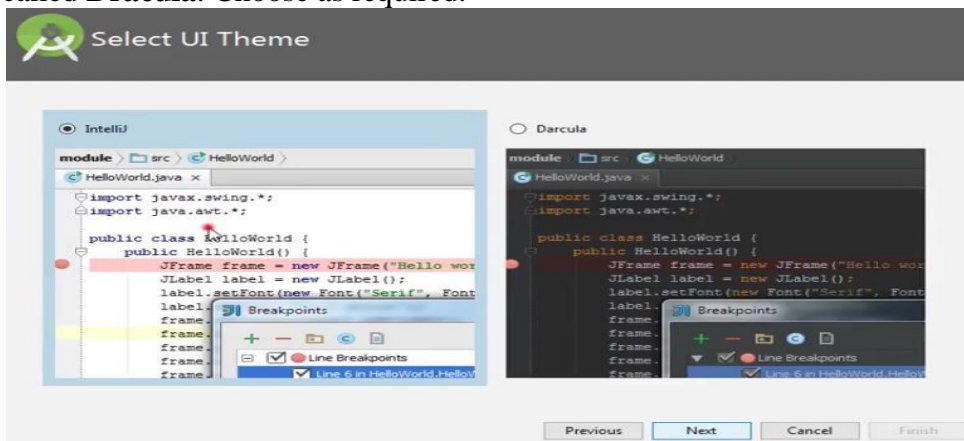




Click on Next.

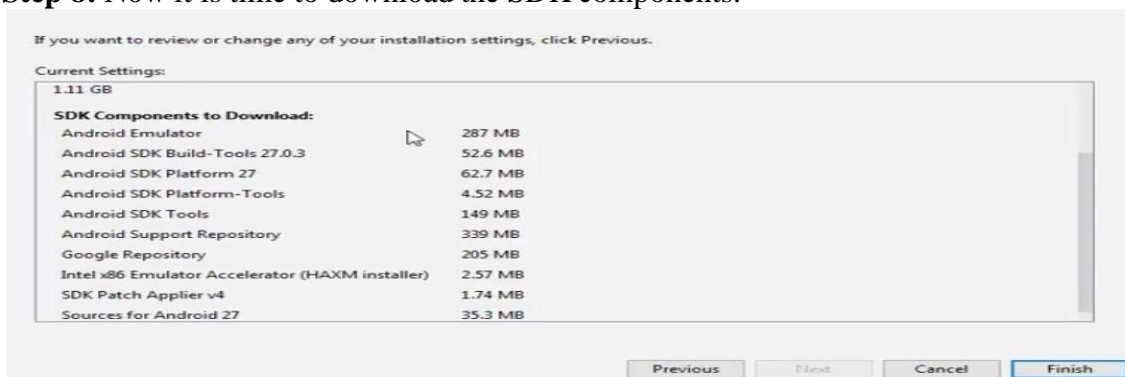


Choose Standard and click on Next. Now choose the theme, whether the **Light** theme or the **Dark** one. The light one is called the **IntelliJ** theme whereas the dark theme is called **Dracula**. Choose as required.



Click on the Next button.

**Step 8:** Now it is time to download the SDK components.



Click on Finish. Components begin to download let it complete.



The Android Studio has been successfully configured. Now it's time to launch and build apps. Click on the Finish button to launch it.

**Step 9:** Click on **Start a new Android Studio project** to build a new app.



## UNIT -III

### Control flow:

As of 2016, about 86% of all vulnerabilities on Android are memory safety related. Most vulnerabilities are exploited by attackers changing the normal control flow of an application to perform arbitrary malicious activities with all the privileges of the exploited application. Control flow integrity (CFI) is a security mechanism that disallows changes to the original control flow graph of a compiled binary, making it significantly harder to perform such attacks.

In Android 8.1, we enabled LLVM's implementation of CFI in the media stack. In Android 9, we enabled CFI in more components and also the kernel. System CFI is on by default but you need to enable kernel CFI.

LLVM's CFI requires compiling with Link-Time Optimization (LTO). LTO preserves the LLVM bitcode representation of object files until link-time, which allows the compiler to better reason about what optimizations can be performed. Enabling LTO reduces the size of the final binary and improves performance, but increases compile time. In testing on Android, the combination of LTO and CFI results in negligible overhead to code size and performance; in a few cases both improved.

For more technical details about CFI and how other forward-control checks are handled, see the LLVM design documentation.

### Implementing system CFI

CFI is enabled by default if you use Clang and the Android build system. Because CFI helps keep Android users safe, you should not disable it.

In fact, we strongly encourage you to enable CFI for additional components. Ideal candidates are privileged native code, or native code that processes untrusted user input. If you're using clang and the Android build system, you can enable CFI in new components by adding a few lines to your makefiles or blueprint files.

### Supporting CFI in makefiles

To enable CFI in a make file, such as /platform/frameworks/av/cmds/stagefright/Android.mk, add:

```
LOCAL_SANITIZE := cfi
# Optional features
LOCAL_SANITIZE_DIAG := cfi
LOCAL_SANITIZE_BLACKLIST := cfi_blacklist.txt
```

LOCAL\_SANITIZE specifies CFI as the sanitizer during the build.

LOCAL\_SANITIZE\_DIAG turns on diagnostic mode for CFI. Diagnostic mode prints out additional debug information in logcat during crashes, which is useful while developing and testing your builds. Make sure to remove diagnostic mode on productions builds, though.

LOCAL\_SANITIZE\_BLACKLIST allows components to selectively disable CFI instrumentation for individual functions or source files. You can use a blacklist as a last resort to fix any user-facing issues that might otherwise exist. For more details, see Disabling CFI.

### Troubleshooting

If you're enabling CFI in new components, you may run into a few issues with function *type* mismatch errors and assembly code type mismatch errors.

Function type mismatch errors occur because CFI restricts indirect calls to only jump to functions that have the same dynamic type as the static type used in the call. CFI restricts virtual and non-virtual member function calls to only jump to objects that are a derived class of the static type of the object used to make the call. This means, when you have code that violates either of these assumptions, the instrumentation that CFI adds will abort. For example, the stack trace shows a SIGABRT and logcat contains a line about control flow integrity finding a mismatch.

Here are two example CLs:

**Bluetooth:** /c/platform/system/bt/+532377

**NFC:** /c/platform/system/nfc/+527858

Another possible issue is trying to enable CFI in code that contains indirect calls to assembly. Because assembly code is not typed, this results in a type mismatch.

To fix this, create native code wrappers for each assembly call, and give the wrappers the same function signature as the calling pointer. The wrapper can then directly call the assembly code. Because direct branches are not instrumented by CFI (they cannot be repointed at runtime and so do not pose a security risk), this will fix the issue.

If there are too many assembly functions and they cannot all be fixed, you can also blacklist all

functions that contain indirect calls to assembly. This is not recommended as it disables CFI checks on these functions, thereby opening attack surface.

### **Android Directory Structure**

Generally we have three kinds of android projects such as:

#### **Android Projects:**

An Android project is the container for our application's source code, resource files, and files such as the Ant build and Android Manifest file. An application project is the main type of project and the contents are eventually built into an **.apk** file that you install on a device.

#### **Test Projects:**

These projects contain code to test our application projects and are built into applications that run on a device.

#### **Library Projects:**

These projects contain shareable Android source code and resources that we can reference in Android projects. This is useful when we have common code that we want to reuse. Library projects cannot be installed onto a device, however, they are pulled into the **.apk** file at build time.

#### **Source: <https://developers.android.com/>**

The main objective of an android project is to build an **.apk** file from it. And for this when we start with an android project then some directories creates by default and some needs few customization to enhance android applications. So let's understand the directory structure in android project to develop an android application.

**/src** – This is the most common folder in any android project. It contains our java source code for android application. Under this folder we can see our java source files in which we write logic for android applications.

**/gen** – In most IDE(such as Eclipse) this folder is created by IDE for configuration purpose. Under this folder we can find two java files such as *BuildConfig.java* and *R.java*. Both files are auto generated by IDE which helps android project to work smoothly.

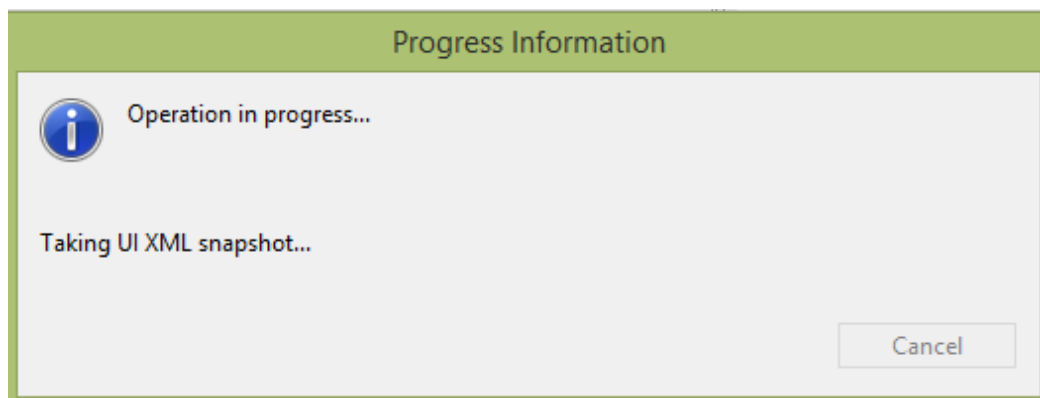
#### **UI:**

UI screen components

A typical user interface of an android application consists of action bar and the application content area.

- Main Action Bar
- View Control
- Content Area
- Split Action Bar

These components have also been shown in the image below –



### **Understanding Screen Components**

The basic unit of android application is the activity. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

#### **View and ViewGroups**

An activity is consist of views. A view is just a widget that appears on the screen. It could be button e.t.c. One or more views can be grouped together into one ViewGroup. Example of ViewGroup includes layouts.

#### **Types of layout:**

There are many types of layout. Some of which are listed below –

- Linear Layout
- Absolute Layout
- Table Layout
- Frame Layout
- Relative Layout

### **Linear Layout:**

Linear layout is further divided into horizontal and vertical layout. It means it can arrange views in a single column or in a single row. Here is the code of linear layout(vertical) that includes a text view.

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

### **Absolute Layout:**

The AbsoluteLayout enables you to specify the exact location of its children. It can be declared like this.

```
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >

    <Button
        android:layout_width="188dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="126px"
        android:layout_y="361px" />
</AbsoluteLayout>
```

### **Table Layout:**

The TableLayout groups views into rows and columns. It can be declared like this.

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent" >
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width="120dp"
            />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200dp" />
    </TableRow>
</TableLayout>
```

### **Relative Layout:**

The RelativeLayout enables you to specify how child views are positioned relative to each other. It can be declared like this.

```
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >
</RelativeLayout>
```

**Frame Layout:**

The FrameLayout is a placeholder on screen that you can use to display a single view. It can be declared like this.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/lblComments"
    android:layout_below="@+id/lblComments"
    android:layout_centerHorizontal="true" >
    <ImageView
        android:src="@drawable/droid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</FrameLayout>
```

Apart from these attributes, there are other attributes that are common in all views and ViewGroups. They are listed below –

Sr.No	View & description
1	<b>layout_width</b> Specifies the width of the View or ViewGroup
2	<b>layout_height</b> Specifies the height of the View or ViewGroup
3	<b>layout_marginTop</b> Specifies extra space on the top side of the View or ViewGroup
4	<b>layout_marginBottom</b> Specifies extra space on the bottom side of the View or ViewGroup
5	<b>layout_marginLeft</b> Specifies extra space on the left side of the View or ViewGroup
6	<b>layout_marginRight</b> Specifies extra space on the right side of the View or ViewGroup
7	<b>layout_gravity</b> Specifies how child Views are positioned
8	<b>layout_weight</b> Specifies how much of the extra space in the layout should be allocated to the View

**Units of Measurement**

When you are specifying the size of an element on an Android UI, you should remember the following units of measurement.

Sr.No	Unit & description
1	<b>dp</b> Density-independent pixel. 1 dp is equivalent to one pixel on a 160 dpi screen.



2	<b>sp</b> Scale-independent pixel. This is similar to dp and is recommended for specifying font sizes
3	<b>pt</b> Point. A point is defined to be 1/72 of an inch, based on the physical screen size.
4	<b>px</b> Pixel. Corresponds to actual pixels on the screen

#### Screen Densities

Sr.No	Density & DPI
1	<b>Low density (ldpi)</b> 120 dpi
2	<b>Medium density (mdpi)</b> 160 dpi
3	<b>High density (hdpi)</b> 240 dpi
4	<b>Extra High density (xhdpi)</b> 320 dpi

#### Optimizing layouts:

Here are some of the guidelines for creating efficient layouts.

- Avoid unnecessary nesting
- Avoid using too many Views
- Avoid deep nesting

#### User Interface (UI) Design In Android

**User Interface (UI) design** in Android is a graphical representation of views displayed on a smartphone or tablet. It allows users to interact with the features, functions, and contents of the application.

To design UI, you need no prior programming knowledge, although it is nice to have web developing skills or programming skills.

Every application has a user interface with which users can interact. Android provides various pre-built UI components that allow you to build a GUI for your application. Android

also provides other UI modules for special interfaces, such as dialogues, notifications, menu, etc.

### **Components used for Android application:**

There are many components for the Android application. Let's consider a few of them here:

- Main Action Bar (MAR)
- Split Action Bar (SAB)
- Content Area

These play a significant role while developing a complex Android application.

### **Views components**

Views are used to customize User Interface (UI) design. A view is considered a building block for a proper User Interface created from the view class.

A view can also be defined as a small rectangular box in Android development that responds to the user inputs, for example, buttons, checkboxes, etc. Basically, a View is what a user sees and interacts with.

**View group** is an invisible container of other views, such as child view and other view groups. This view class is a super class of all the **GUI** components in Android. Views play a key role in the development of an application because you need settings present in the application, e.g.,



color, style, themes, etc.

Figure 1.0. A simple UI Design

In the Android studio, we commonly use a view called “edit text and images”. Images are classified as widgets used to create an interactive UI component, such as buttons, text fields, etc.

### **Layout as a subclass of View group**

The Layout defines the visual structures of the UI of user application. All elements in the layout are built using a hierarchy of views and view group object.

The layout can be declared through a programming language, or through a simple XML layout file located in the resource layout folder of any project you work on. Android provides a straightforward XML vocabulary that corresponds to the view class and subclasses, such as those present for widget and layout. Due to this, the layout can be treated.

Layouts are kept in the folder called resources. Android studio creates a default XML layout file in the resources layout folder that is extremely useful if you know the basic at the time of compiling.

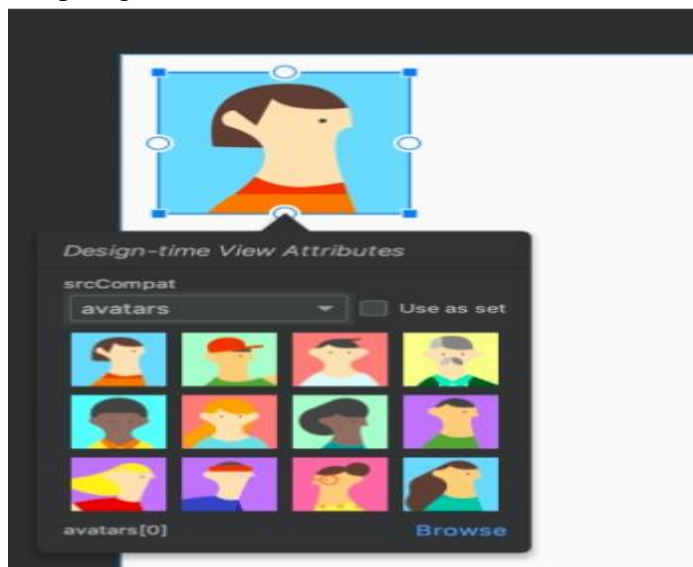


Figure 1.1. A simple layout with imageView

### **TextView in Android:**

TextView is the user interface which displays the text message on the screen to the user. It is based on the layout size, style, and color, etc. TextView is used to set and display the text

according to our specifications.

In Android, when we create a new project with a specific name Android provides us a TextView in our activity\_main.xml file, and the second file is the MainActivity.java file for the coding of Java programming language.

### **Properties of TextView in Android**

**android:id-ID** is an attribute used to define a TextView uniquely.

```
android:id="@+id/text_view"
```

**android:layout\_height-** Height is the attribute which is used to set the height of the TextView. It can be like wrap\_content that means TextView will be the same size as the text that is given by the user, match\_parent that means TextView will consume the size of the whole screen. It is not related to the text of the user, and we can also define the size of the text according to the user in DPs.

**android:layout\_width-** Width is the attribute which is used to set the width of the TextView. Like height attribute, it is also can be like wrap\_content, which means TextView will be the same size as the text that is given by the user, match\_parent which means TextView will consume the size of the whole screen. It is not related to the text of the user, and we can also define the size of the text according to the user in DPs.

**android:layout\_centerInParent-** centerInParent is the attribute which is used to set the text in the center of the screen. It is an optional attribute of the TextView. It is set as true and false by the user.

```
android:layout_centerInParent="true"
```

**android:text-** This attribute handles the text that is given by the user in the form of the string.

```
android:text="Hello World!"
```

**android:hint-** This attribute is used when there is no text is given by the user, in this hint attribute will show how the text will display on the screen.

```
android:hint="Hello Android"
```

**android:inputType-** This attribute is used to set the type of the TextView. The type of text can be phone number, eMail, and Password, etc.

```
android:inputType="text"
```

**android:textSize-** This attribute is used to set the size of the TextView. The size of the TextView given in the size of sp.

```
android:textSize="30sp"
```

**android:textStyle-** This attribute is used to set the style of the TextView. The style of the TextView given as bold, italic, etc.

```
android:textStyle="bold"
```

**android:textColorHighlight-** This attribute is used to set the color of the text selection highlight.

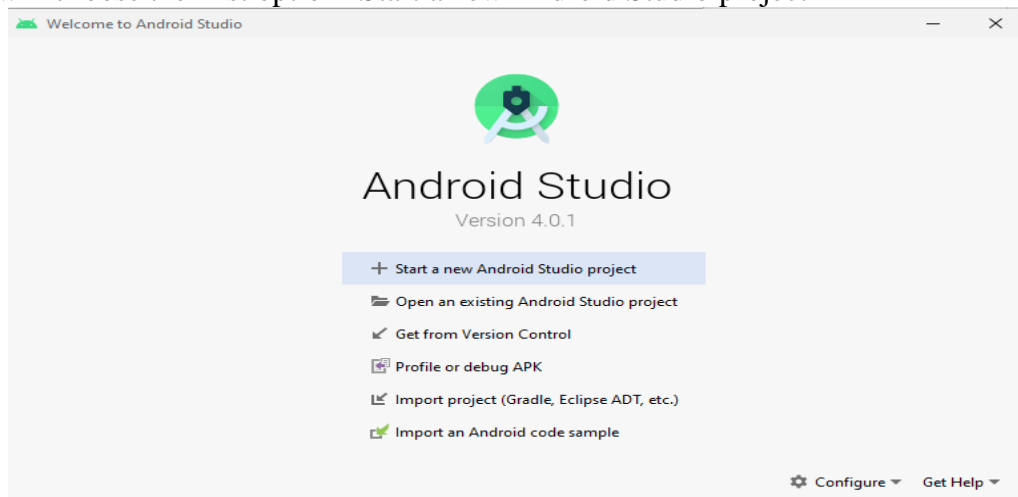
```
android:textColorHighlight="@color/colorPrimaryDark"
```

### **Note**

Here we will create a new project for the TextView example in Android. Let's start working with Android Studio.

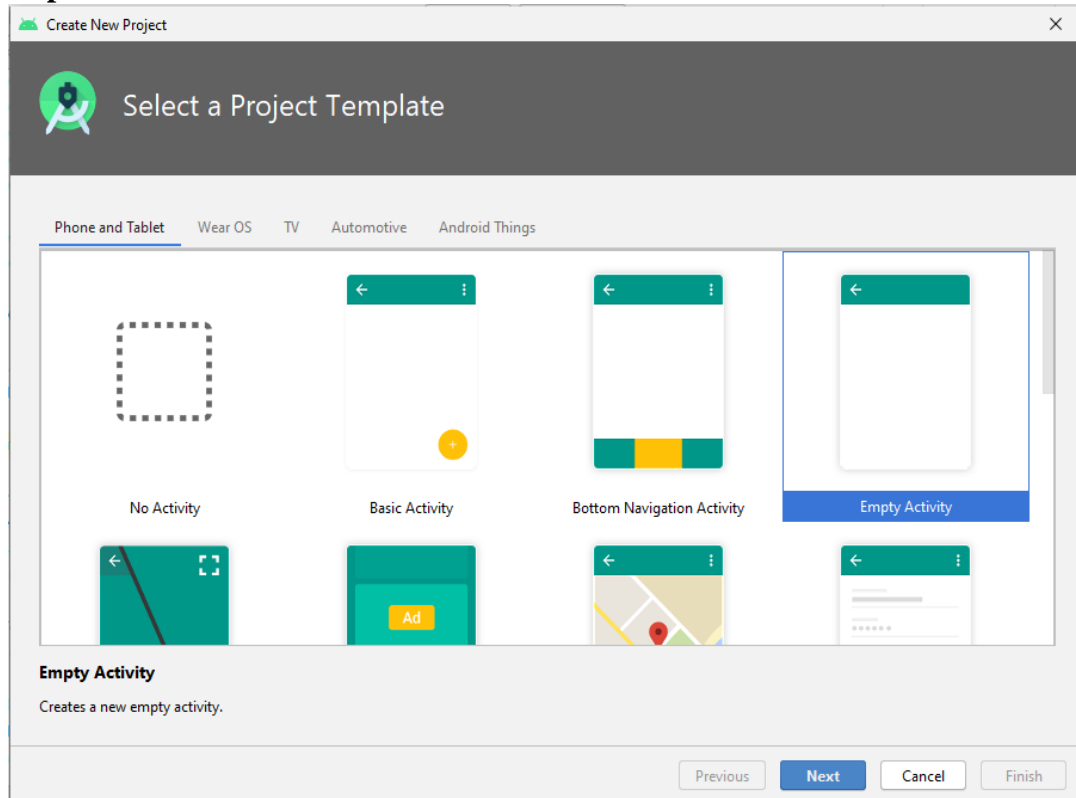
#### **Step 1**

First of all, when we will open the Android Studio we will see the screen with many options. we will choose the first option "Start a new Android Studio project"



**Explanation**

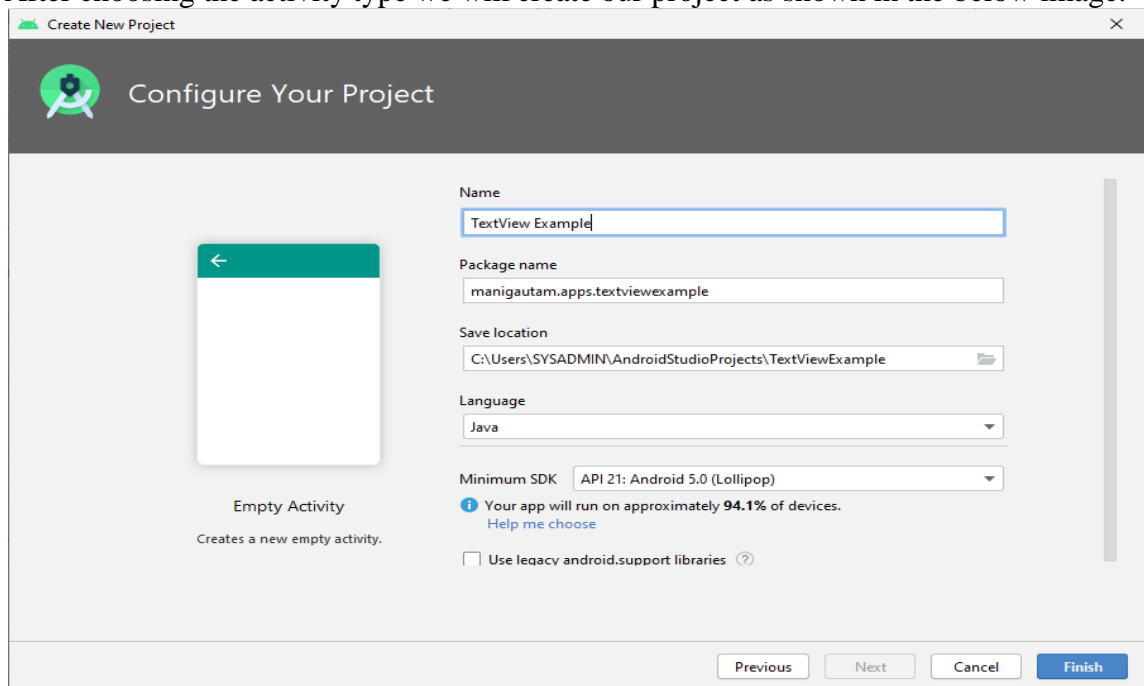
In the above image, we choose "Start a new Android Studio project", and create a new project for our TextView example.

**Step 2****Explanation:**

In the above image, as we can see, there are many examples of the activities we want to create for our project. We will choose an "Empty Activity" for our example project for the TextView in Android.

**Step 3**

After choosing the activity type we will create our project as shown in the below image.

**Explanation:**

We will give the name "TextView Example" for our project and will click on the "Finish" button.

**Note**

Android Studio will create a project named "TextView Example". Android Studio will create two files in the project MainActivity.java and activity\_main.xml.

**Step 4**

As we know, Android Studio creates two files. Now we will see the activity\_main.xml file of the project. The code of this file is listed below.

```

<?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=".MainActivity">
    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
    </RelativeLayout>

```

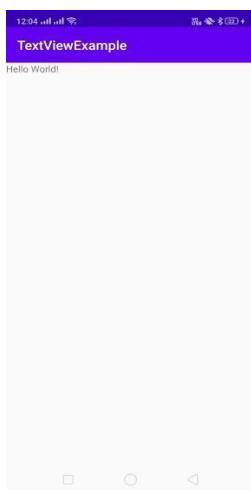
### Explanation

In the above code example, as we see Android Studio provides a TextView as an example. In this example, we have RelativeLayout as the root Layout. In the TextView, we have four attribute ids for the identification of the TextView. Height, and width is set as the wrap\_content so the TextView will take the space according to the text, the last attribute is the text that takes the text as the string.

### Explanation

This is the code of the MainActivity.java file of our project. Now we will run our TextView Example project.

The following code of our project generates the following output.



### TextView examples with different attributes

Now we will update our XML file of the project with different attributes of the TextView. Let's see the different examples of TextView.

#### Example 1

```

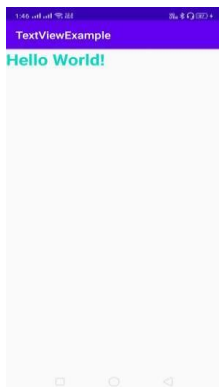
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
android:id="@+id/text_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
android:textColor="@color/colorAccent"
android:textSize="30sp"
android:textStyle="bold" />
</RelativeLayout>
```

### Explanation

In this code example, we use attributes like `textColor`, `textSize`, `textStyle` for making the `textView` more attractive, and here is the output of the code.



### Example 2

```
<?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=".MainActivity">
<TextView
android:id="@+id/text_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="30dp"
```



```

android:text="Hello World!"
android:textColor="@color/colorAccent"
android:textSize="30sp"
android:textStyle="bold" />
</RelativeLayout>

```

### Explanation

In this code example, we use attributes **android: padding="30dp"** for making the textView more attractive, and here is the output of the code.



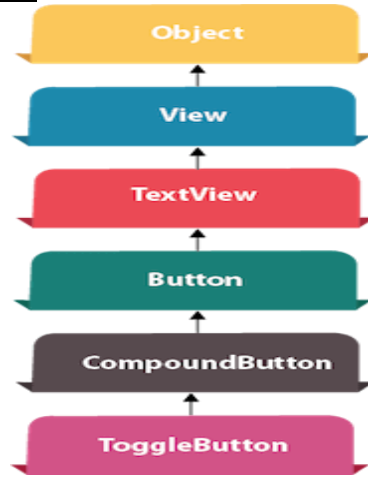
### Example 3

```

<?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=".MainActivity">
    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_centerInParent="true"
        android:textColor="@color/colorAccent"
        android:textSize="30sp"
        android:textStyle="bold" />
    </RelativeLayout>

```

**Android ToggleButton:**



**Android Toggle Button** can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Android Toggle Button and Switch both are the subclasses of Compound Button class.

Android Toggle Button class

ToggleButton class provides the facility of creating the toggle button.

XML Attributes of Toggle Button class

The 3 XML attributes of Toggle Button class.

XML Attribute	Description
android:disabledAlpha	The alpha to apply to the indicator when disabled.
android:textOff	The text for the button when it is not checked.
android:textOn	The text for the button when it is checked.

**Methods of ToggleButton class**

The widely used methods of ToggleButton class are given below.

Method	Description
CharSequence getTextOff()	Returns the text when button is not in the checked state.
CharSequence getTextOn()	Returns the text for when button is in the checked state.
void setChecked(boolean checked)	Changes the checked state of this button.

**Android ToggleButton Example**

**activity\_main.xml**

Drag two toggle button and one button for the layout. Now the activity\_main.xml file will look like this:

**File: activity\_main.xml**

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

```

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
pk/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.togglebutton.MainActivity">
    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toStartOf="@+id/toggleButton2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="60dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="144dp"
        android:layout_marginLeft="148dp"
        android:text="Submit"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>

```

### Android RadioButton

**RadioButton** is a two states button which is either checked or unchecked. If a single radio button is unchecked, we can click it to make checked radio button. Once a radio button is checked, it cannot be marked as unchecked by user.

RadioButton is generally used with *RadioGroup*. RadioGroup contains several radio buttons, marking one radio button as checked makes all other radio buttons as unchecked.

Example of Radio Button

In this example, we are going to implement single radio button separately as well as radio button in **RadioGroup**.

**File: activity\_main.xml**

```

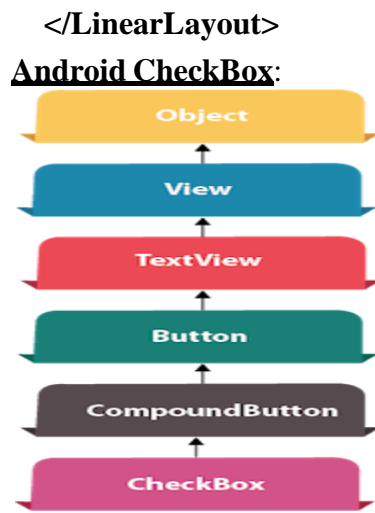
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="example.javatpoint.com.radiobutton.MainActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"

```

```

        android:layout_marginTop="30dp"
        android:gravity="center_horizontal"
        android:textSize="22dp"
        android:text="Single Radio Buttons" />
<!-- Default RadioButtons -->
<RadioButton
    android:id="@+id/radioButton1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="Radio Button 1"
    android:layout_marginTop="20dp"
    android:textSize="20dp" />
<RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Radio Button 2"
    android:layout_marginTop="10dp"
    android:textSize="20dp" />
<View
    android:layout_width="fill_parent"
    android:layout_height="1dp"
    android:layout_marginTop="20dp"
    android:background="#B8B894" />
<TextView
    android:id="@+id/textView2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:gravity="center_horizontal"
    android:textSize="22dp"
    android:text="Radio button inside RadioGroup" />
<!-- Customized RadioButtons -->
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/radioGroup">
    <RadioButton
        android:id="@+id/radioMale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=" Male"
        android:layout_marginTop="10dp"
        android:checked="false"
        android:textSize="20dp" />
    <RadioButton
        android:id="@+id/radioFemale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=" Female"
        android:layout_marginTop="20dp"
        android:checked="false"
        android:textSize="20dp" />
</RadioGroup>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Selected"
    android:id="@+id/button"
    android:onClick="onclickbuttonMethod"
    android:layout_gravity="center_horizontal" />

```



**Android CheckBox** is a type of two state button either checked or unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.

#### Android CheckBox class

The android.widget.CheckBox class provides the facility of creating the CheckBoxes.

Methods of CheckBox class

There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	Description
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

#### Android CheckBox Example

##### activity\_main.xml

Drag the three checkboxes and one button for the layout. Now the activity\_main.xml file will look like this:

##### File: activity\_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
pk/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.checkbox.MainActivity">
    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
  
```

```

    android:layout_marginTop="28dp"
    android:text="Coffee"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkBox" />
<CheckBox
    android:id="@+id/checkBox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="144dp"
    android:layout_marginTop="28dp"
    android:text="Burger"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkBox2" />
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="144dp"
    android:layout_marginTop="184dp"
    android:text="Order"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

```

### **Android ProgressBar:**



We can display the **android progress bar** dialog box to display the status of work being done e.g. downloading file, analyzing status of work etc.

In this example, we are displaying the progress dialog for dummy file download operation.

Here we are using **android.app.ProgressDialog** class to show the progress bar. Android ProgressDialog is the subclass of AlertDialog class.

The **ProgressDialog** class provides methods to work on progress bar like setProgress(), setMessage(), setProgressStyle(), setMax(), show() etc. The progress range of Progress Dialog is 0 to 10000.

Let's see a simple example to display progress bar in android.

- ProgressDialog progressBar = **new** ProgressDialog(**this**);
- progressBar.setCancelable(**true**); //you can cancel it by pressing back button
- progressBar.setMessage("File downloading ...");
- progressBar.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);
- progressBar.setProgress(0); //initially progress is 0
- progressBar.setMax(100); //sets the maximum value 100
- progressBar.show(); //displays the progress bar

### **Android Progress Bar Example by ProgressDialog**

Let's see a simple example to create progress bar using ProgressDialog class.

#### **activity\_main.xml**

Drag one button from the palette, now the activity\_main.xml file will look like this:

#### **File: activity\_main.xml**

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```



```

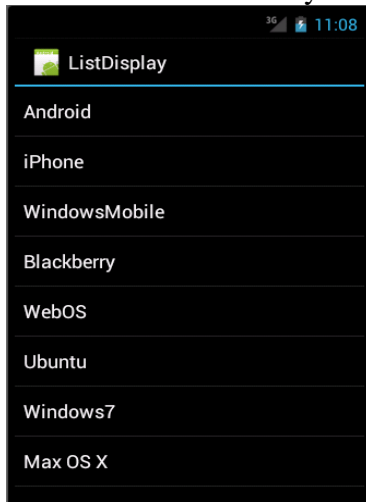
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="116dp"
    android:text="download file" />

```

</RelativeLayout>

### **List view:**

Android **List View** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.



### **List View:**

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter holds the data and send the data to adapter view, the view can takes the data from adapter view and shows the data on different views like as spinner, list view, grid view etc.

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.

Android provides several subclasses of Adapter that are useful for retrieving different kinds of data and building views for an AdapterView ( i.e. ListView or GridView). The common adapters are **ArrayAdapter**, **Base**

**Adapter**, **CursorAdapter**, **SimpleCursorAdapter**, **SpinnerAdapter** and **WrapperListAdapter** r. We will see separate examples for both the adapters.

### **ListView Attributes**

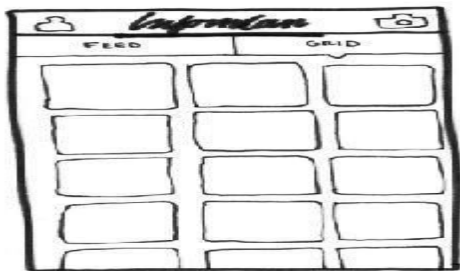
Following are the important attributes specific to GridView –

Sr.No	Attribute & Description
1	<b>android:id</b> This is the ID which uniquely identifies the layout.
2	<b>android:divider</b> This is drawable or color to draw between list items.
3	<b>android:dividerHeight</b> This specifies height of the divider. This could be in px, dp, sp, in, or mm.
4	<b>android:entries</b> Specifies the reference to an array resource that will populate the ListView.

5	<p><b>android:footerDividersEnabled</b></p> <p>When set to false, the ListView will not draw the divider before each footer view. The default value is true.</p>
6	<p><b>android:headerDividersEnabled</b></p> <p>When set to false, the ListView will not draw the divider after each header view. The default value is true.</p>

### Grid View:

Android **GridView** shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a **ListAdapter**



### Grid view

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc.

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.

### GridView Attributes

Following are the important attributes specific to GridView –

Sr.No	Attribute & Description
1	<p><b>android:id</b></p> <p>This is the ID which uniquely identifies the layout.</p>
2	<p><b>android:columnWidth</b></p> <p>This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.</p>
3	<p><b>android:gravity</b></p> <p>Specifies the gravity within each cell. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.</p>
4	<p><b>android:horizontalSpacing</b></p> <p>Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.</p>
5	<p><b>android:numColumns</b></p> <p>Defines how many columns to show. May be an integer value, such as "100" or auto_fit</p>

	which means display as many columns as possible to fill the available space.
6	<p><b>android:stretchMode</b></p> <p>Defines how columns should stretch to fill the available empty space, if any. This must be either of the values –</p> <ul style="list-style-type: none"> <li>➤ none – Stretching is disabled.</li> <li>➤ spacingWidth – The spacing between each column is stretched.</li> <li>➤ columnWidth – Each column is stretched equally.</li> <li>➤ spacingWidthUniform – The spacing between each column is uniformly stretched..</li> </ul>
7	<p><b>android:verticalSpacing</b></p> <p>Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.</p>

#### Example

This example will take you through simple steps to show how to create your own Android application using GridView. Follow the following steps to modify the Android application we created in *Hello World Example* chapter –

Step	Description
1	You will use Android studio IDE to create an Android application and name it as <i>HelloWorld</i> under a package <i>com.example.helloworld</i> as explained in the <i>Hello World Example</i> chapter.
2	Modify the default content of <i>res/layout/activity_main.xml</i> file to include GridView content with the self explanatory attributes.
3	No need to change string.xml, Android studio takes care of defaults strings which are placed at string.xml
4	Let's put few pictures in <i>res/drawable-hdpi</i> folder. I have put sample0.jpg, sample1.jpg, sample2.jpg, sample3.jpg, sample4.jpg, sample5.jpg, sample6.jpg and sample7.jpg.
5	Create a new class called <b>ImageAdapter</b> under a package <i>com.example.helloworld</i> that extends <i>BaseAdapter</i> . This class will implement functionality of an adapter to be used to fill the view.
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

#### **Image View:**

In Android, ImageView class is used to display an image file in application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. An android is enriched with some of the best UI design widgets that allows us to build good looking and attractive UI based application.

**Important Note:** ImageView comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the imageview. Some of them scaleTypes configuration properties are center, center\_crop, fit\_xy, fitStart etc. You can read our ScaleType tutorial to learn all details on it.

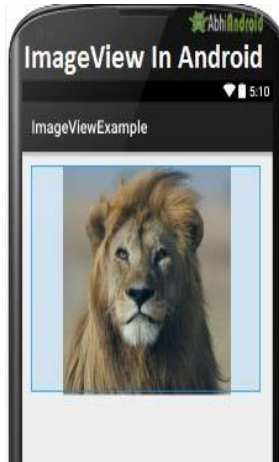
#### **Below is an ImageView code in XML:**

Make sure to save lion image in drawable folder

```

<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/lion" />

```



### Attributes of ImageView:

Now let's we discuss some important attributes that helps us to configure a ImageView in your xml file.

**id:** id is an attribute used to uniquely identify a image view in android. Below is the example code in which we set the id of a image view.

```

<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>

```

**src:** src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.

### Android ScrollView (Vertical)

The **android.widget.ScrollView** class provides the functionality of scroll view. ScrollView is used to scroll the child elements of palette inside ScrollView. Android supports vertical scroll view as default scroll view. Vertical ScrollView scrolls elements vertically.

Android uses *HorizontalScrollView* for horizontal ScrollView.

Let's implement simple example of vertical ScrollView.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.example.test.scrollviews.MainActivity">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Vertical ScrollView example"
    android:id="@+id/textView"
    android:layout_gravity="center_horizontal"
    android:layout_centerHorizontal="true"
    android:layout_alignParentTop="true" />
  <ScrollView android:layout_marginTop="30dp"

```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/scrollView">
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 3" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 4" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 5" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 6" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 7" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 8" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 9" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 10" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 11" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 12" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 13" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

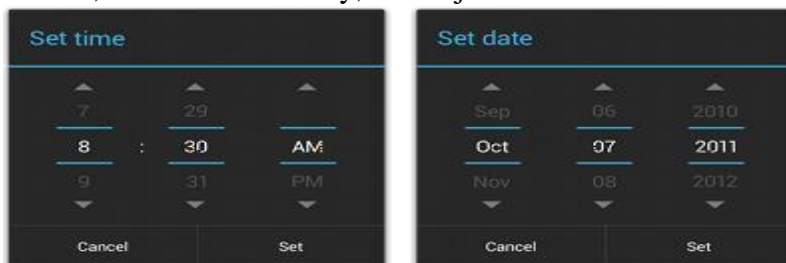
```

    android:text="Button 14" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 15" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 16" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 17" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 18" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 19" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 20" />
</LinearLayout>
</ScrollView>
</RelativeLayout>

```

### **Pickers:**

**Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs.** Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year). Using these pickers helps ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale.



We recommend that you use [DialogFragment](#) to host each time or date picker. The [DialogFragment](#) manages the dialog lifecycle for you and allows you to display the pickers in different layout configurations, such as in a basic dialog on handsets or as an embedded part of the layout on large screens.

Key classes are the following:

[DatePickerDialog](#)

[TimePickerDialog](#)

### **Creating a Time Picker**

To display a [TimePickerDialog](#) using [DialogFragment](#), you need to define a fragment class that extends [DialogFragment](#) and return a [TimePickerDialog](#) from the fragment's [onCreateDialog\(\)](#) method.

### **Showing the time picker**

Once you've defined a [DialogFragment](#) like the one shown above, you can display the time picker by creating an instance of the [DialogFragment](#) and calling [show\(\)](#).

For example, here's a button that, when clicked, calls a method to show the dialog:

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/pick_time"
    android:onClick="showTimePickerDialog" />

```

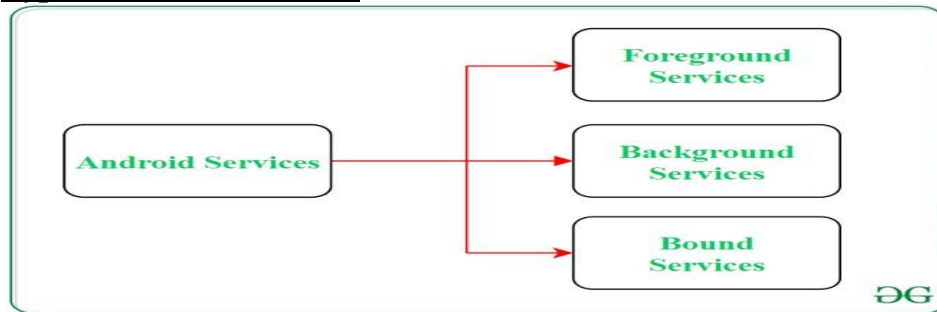


## Unit-IV

### Services in Android with Example:

Services in Android are a special component that facilitates an application to run in the background in order to perform long-running operation tasks. The prime aim of a service is to ensure that the application remains active in the background so that the user can operate multiple applications at the same time. A user-interface is not desirable for android services as it is designed to operate long-running processes without any user intervention. A service can run continuously in the background even if the application is closed or the user switches to another application. Further, application components can bind itself to service to carry out **inter-process communication(IPC)**. There is a major difference between android services and threads, one must not be confused between the two. Thread is a feature provided by the Operating system to allow the user to perform operations in the background. While service is an android component that performs a long-running operation about which the user might not be aware of as it does not have UI.

### Types of Android Services



### Foreground Services:

Services that notify the user about its ongoing operations are termed as Foreground Services. Users can interact with the service by the notifications provided about the ongoing task. Such as in downloading a file, the user can keep track of the progress in downloading and can also pause and resume the process.

### Background Services:

Background services do not require any user intervention. These services do not notify the user about ongoing background tasks and users also cannot access them. The process like schedule syncing of data or storing of data fall under this service.

### Bound Services:

This type of android service allows the components of the application like activity to bound themselves with it. Bound services perform their task as long as any application component is bound to it. More than one component is allowed to bind themselves with a service at a time. In order to bind an application component with a service **bindService()** method is used.

### The Life Cycle of Android Services

In android, services have 2 possible paths to complete its life cycle namely **Started and Bounded**.

### Started Service (Unbounded Service):

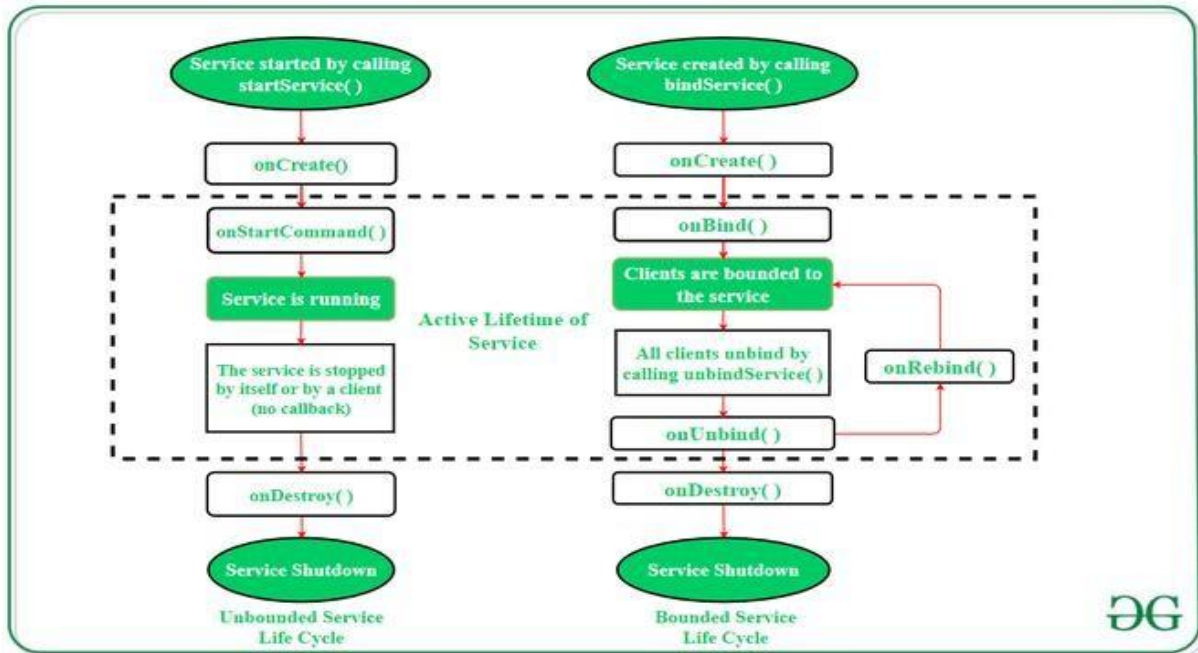
By following this path, a service will initiate when an application component calls the **startService()** method. Once initiated, the service can run continuously in the background even if the component is destroyed which was responsible for the start of the service. Two option are available to stop the execution of service:

By calling **stopService()** method,

The service can stop itself by using **stopSelf()** method.

### Bounded Service:

It can be treated as a server in a client-server interface. By following this path, android application components can send requests to the service and can fetch results. A service is termed as bounded when an application component binds itself with a service by calling **bindService()** method. To stop the execution of this service, all the components must unbind themselves from the service by using **unbindService()** method.



To carry out a downloading task in the background, the **startService()** method will be called. Whereas to get information regarding the download progress and to pause or resume the process while the application is still in the background, **the service must be bounded with a component** which can perform these tasks.

**Example of Android Services**

**Step 1: Create a new project**

- Click on File, then New => New Project.
- Choose Empty activity
- Select language as Java/Kotlin
- Select the minimum SDK as per your need.

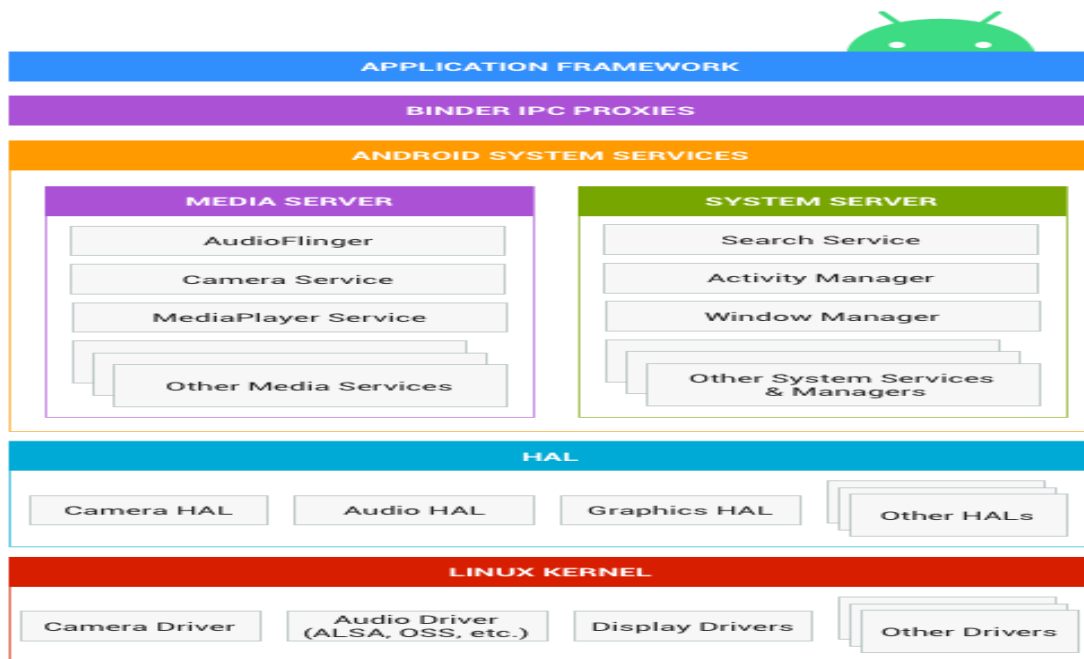
**Step 2: Modify strings.xml file**

All the strings which are used in the activity are listed in this file.

```
<resources>
  <string name="app_name">Services_In_Android</string>
  <string name="heading">Services In Android</string>
  <string name="startButtonText">Start the Service</string>
  <string name="stopButtonText">Stop the Service</string>
</resources>
```

**Android Architecture:**

Android system architecture contains the following components:



**Figure 1.** Android system architecture

**Application framework:** The application framework is used most often by application developers. As a hardware developer, you should be aware of developer APIs as many map

directly to the underlying HAL interfaces and can provide helpful information about implementing drivers.

**Binder IPC:** The Binder Inter-Process Communication (IPC) mechanism allows the application framework to cross process boundaries and call into the Android system services code. This enables high level framework APIs to interact with Android system services. At the application framework level, this communication is hidden from the developer and things appear to "just work".

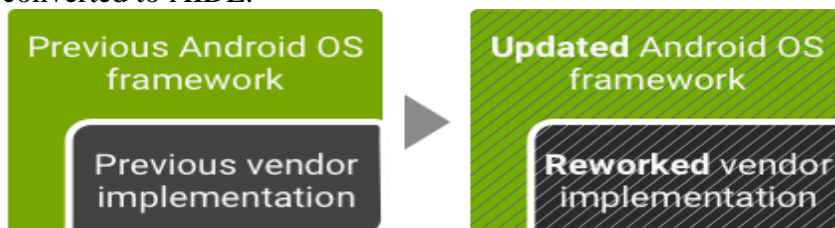
**System services:** System services are modular, focused components such as Window Manager, Search Service, or Notification Manager. Functionality exposed by application framework APIs communicates with system services to access the underlying hardware. Android includes two groups of services: *system* (such as Window Manager and Notification Manager) and *media* (services involved in playing and recording media).

**Hardware abstraction layer (HAL):** A HAL defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations. Using a HAL allows you to implement functionality without affecting or modifying the higher level system. HAL implementations are packaged into modules and loaded by the Android system at the appropriate time. For details, see [Hardware Abstraction Layer \(HAL\)](#).

**Linux kernel:** Developing your device drivers is similar to developing a typical Linux device driver. Android uses a version of the Linux kernel with a few special additions such as Low Memory Killer (a memory management system that is more aggressive in preserving memory), wake locks (a [PowerManager](#) system service), the Binder IPC driver, and other features important for a mobile embedded platform. These additions are primarily for system functionality and do not affect driver development. You can use any version of the kernel as long as it supports the required features (such as the binder driver). However, we recommend using the latest version of the Android kernel. For details, see [Building Kernels](#).

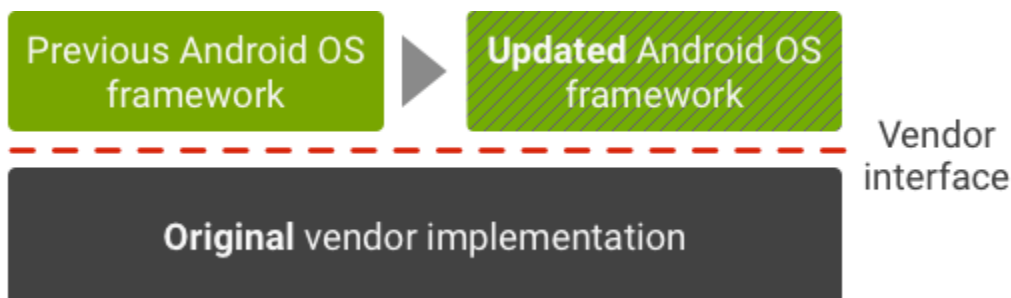
**HAL interface definition language (AIDL/HIDL):**

Android 8.0 re-architected the Android OS framework (in a project known as *Treble*) to make it easier, faster, and less costly for manufacturers to update devices to a new version of Android. In this new architecture, the HAL interface definition language (HIDL, pronounced "hide-l") specifies the interface between a HAL and its users, enabling the Android framework to be replaced without rebuilding the HALs. In Android 10, HIDL features were incorporated into AIDL. Since then, HIDL is deprecated and is only used by subsystems which haven't yet converted to AIDL.



**Figure 2.** Legacy Android update environment

In Android 8.0 and higher, a new stable vendor interface provides access to the hardware-specific parts of Android, so device makers can deliver new Android releases simply by updating the Android OS framework—without additional work required from the silicon manufacturers:



**Figure 3.** Current Android update environment

**Security is a compatibility requirement:**

The security model is part of the Android specification, which is defined in the **Compatibility Definition Document (CDD)** and enforced by the Compatibility (CTS), Vendor, and other test suites. Devices that do not conform to CDD and do not pass CTS are not Android. Within the scope of this article, we define *rooting* as modifying the system to allow starting processes that are not subject to sandboxing and isolation. Such rooting, both intentional and malicious, is a specific example of a non-compliant change that violates CDD. As such, only CDD-compliant devices are considered.

**Factory reset restores the device to a safe state:**

In the event of security model bypass leading to a persistent compromise, a factory reset, which wipes/reformats the writable data partitions, returns a device to a state that depends only on integrity protected partitions. In other words, system software does not need to be re-installed, but wiping the data partition(s) will return a device to its default state.

**Applications are security principals:**

The main difference to traditional operating systems that run apps in the context of the logged-in user account is that Android apps are not considered to be fully authorized agents for user actions. In the traditional model typically implemented by server and desktop OS, there is often no need to even exploit the security boundary, because running malicious code with the full

**Plan Functionality and Features:**

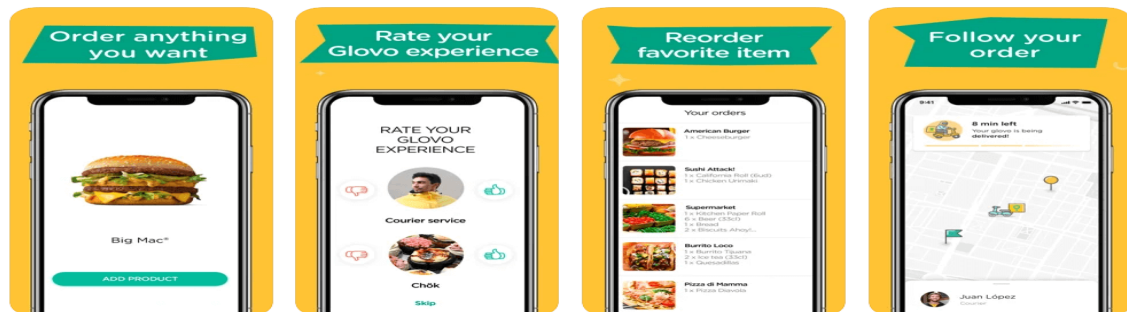
Let's move on to the second step to create an app. After defining the objectives, the second important thing that comes into play is planning your app's functionality and features.



**Identify Top Functionalities and Features for Your App:**

This is one of the creative steps to make an app where you'll have to write down all the functionalities you want to add and features needed to achieve the solutions and get the expected results, mostly known as an MVP version of the app.

The best way is to make sure you carry out market research to find what your competitors offer on the app stores and see what they are lacking or innovating.



**Image Source: Glovo**

For Glovo, we have integrated features like real-time order tracking, push notifications, search orders, places, and items, and POS. So, ensure you are integrating all the essential features in your app to make it successful in the long run.

**Remove Irrelevant Features From Your List**

Just remember, adding irrelevant features won't do any good. Ultimately, it affects the app's performance. At the initial stage, we recommend you make sure you only list down the features that will add value to your app. Once you release the first version, then work on the remaining features.

**Research Your Competitors**

Never underestimate the value of research and finding insights before making an app. This way, you can find the scope of your app idea and also implement the required features into your app. So, ensure you do research on your competitors before building apps.



**Design Wireframes:**

You've defined your objectives, planned functionalities and features, and done competitors' research to get insights on creating an app.





### **Wireframing:**

Wireframing is the visual representation of your app's layout and the flow between the different screens. This is one of the best distraction-free methods where you don't have to worry about other graphic elements.

### **Test Wireframes:**

Once you are done with designing the use cases of the app, it's time to test them. This is also a very important step of your app-building where you can test the flow of your app to make improvements in the user experience.

### **Use InVision**

Here is one of the short video tutorials on creating an interactive prototype with InVision.

InVision is simple and very easy to use. First, go to the official website and register for a free account. Then, click on the '+' icon to create your first project and select 'create a new prototype'.

### **Choose a Custom App Development Path**

Once you finish the wireframing for creating an app, one of the essential steps comes to build an app. In this part, you must choose the app development platform and coding language. We know it's not easy for all people to discuss application-building platforms and programming languages. But if you are clear with your requirements, budget, and target users, this isn't that difficult too.

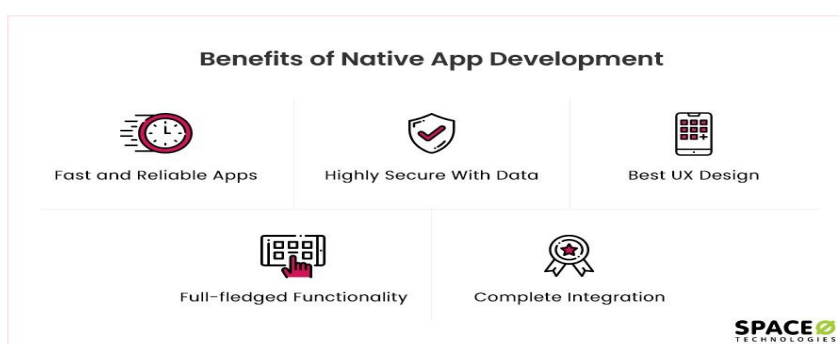


### **Select Platform As Per Your Requirements And Budget**

One of the crucial steps to building an app is to choose the powerful app development platform which suits your requirements and budget. Also, if you have a question such as what are the requirements for developing an app, don't worry, we have covered that in the following section as well.

### **Native App Development**

This app development simply means developing an app for a particular operating system. Android devices or iPhone devices. Suppose you want to build an app to gain maximum downloads and generate revenue through ads. In this case, you can easily leverage the user base of Android.



### **Mobile App Frameworks**

The second path allows you to create hybrid apps that can run on both Android and iOS platforms. Several frameworks are available, such as React Native, Framework 7, PhoneGap, which lets you create a single app and deploy it on both Google Play Store and Apple App Store. As a result, there's no need to spend a hefty amount on two development teams and maintain two code bases.

### **Drag and Drop Mobile App Builder**

If your budget is extremely limited or you are just not comfortable with the above two ways, let's check out one more way to develop mobile apps. This method is called low code app development.

Many different app-building platforms like Appy Pie, AppSheet, and BuildFire allow you to develop mobile apps with zero coding knowledge. You just have to choose a template or drag and drop elements, and your Android or iPhone app will be ready in no time.

### **Select the Right Programming Language:**

When you have decided on the development path, it won't take time to select the right programming language.

Starting with the Android app, the first thing you'll need is Android Studio. After that, you can design the interface of your app using XML and write all the logic using any language like Kotlin, Java, and C++. According to Google, more than 50% of developers are now using Kotlin to develop their apps, so if you're just starting out, go with Kotlin.

### **Develop Your Mobile App**

Building a quality mobile application considering all the requirements and following the guidelines takes time.

Being the best mobile app development company, we have highlighted some important points

## Unit-v

### Introduction to MIT inventor:

The App Inventor development environment is supported for Mac OS X, GNU/Linux, and Windows operating systems, and several popular Android phone models. Applications created with App Inventor can be installed on any Android phone. (See [system requirements](#).)

Before you can use App Inventor, you need to set up your computer and install the *App Inventor Setup* package on your computer.

### **Steps to use MIT app inventor:**

**Step 1:** Open a Gmail account in case you don't have one.

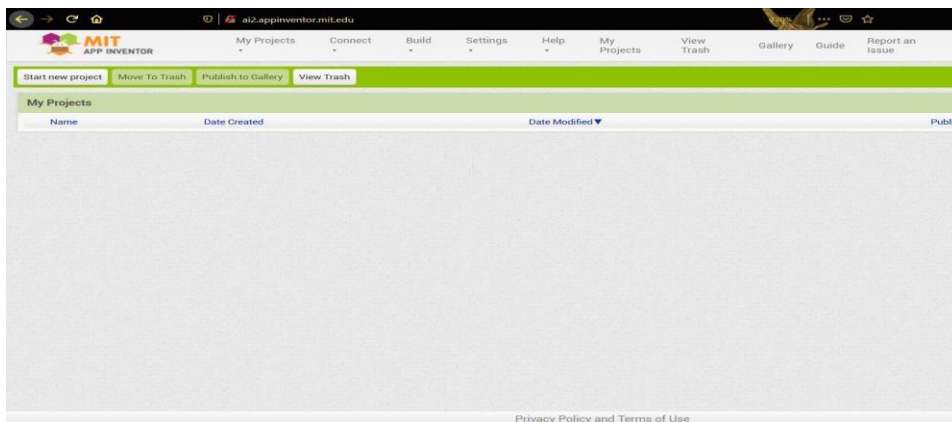
**Step 2:** Open the link <https://appinventor.mit.edu/> and log in to your Gmail account.

**Step 3:** You need to install the App Inventor Companion App (MIT AI2 Companion) on our mobile device that helps in live testing of our application.

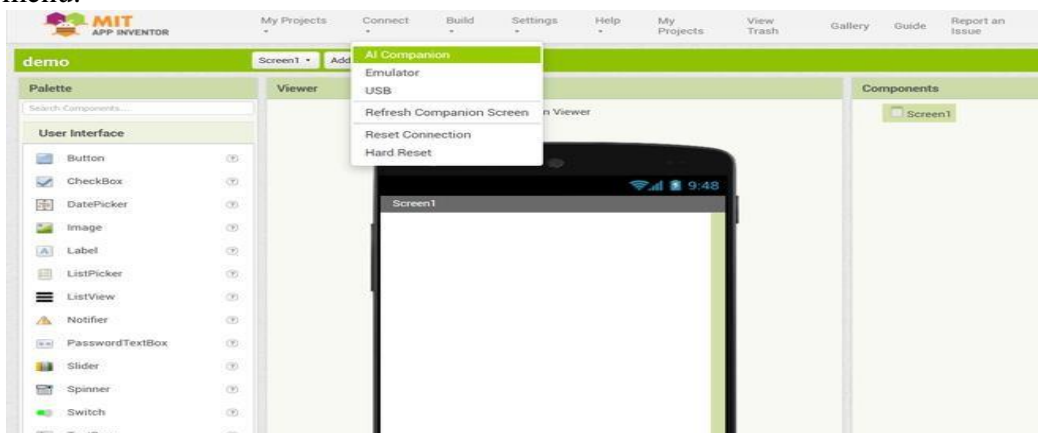


**Step 4:** We need to connect both mobile devices & laptops/desktop should be connected to the same WiFi network.

**Step 5:** To start the app-building click on “Start New Project”.

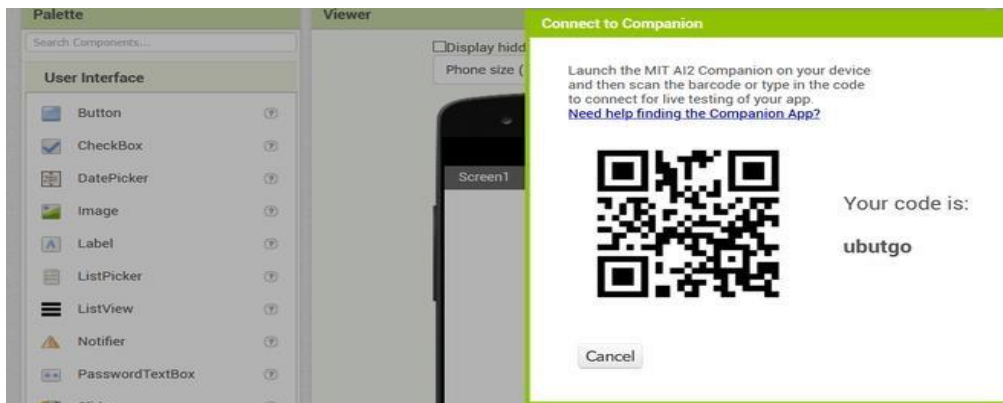


**Step 6:** To connect your mobile device, choose “Connect” and “AI Companion” from the top menu.



**Step 7:** Now to connect the MIT AI2 App on your device and desktop/laptop scan the QR code or type the 6 digit code which is appearing on your PC screen.





**Step 8:** Now you can see the app you are building on your device.

Create a new project in Android Studio, go to File ⇒ New Project and fill all required details to create a new project.

Add the following code to res/layout/activity\_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/parent"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:gravity="center"
    android:orientation="vertical">
    <Button
        android:id="@+id/customDialog"
        android:text="Custom Dialog"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```


In the above code, we have taken button. When user click on button, it will show custom dialog. To show custom dialog we have inflated a view as custom view. so create customview.xml and add the following code -

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Success"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Headline" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla eu erat tincidunt lacus fermentum rutrum."
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Medium" />
    <Button
        android:id="@+id/buttonOk"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
```

```

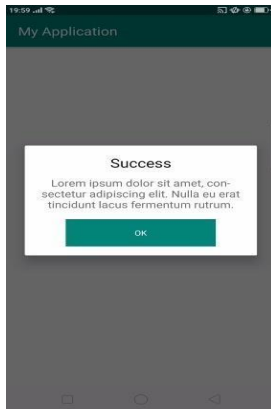
        android:layout_gravity="center"
        android:layout_marginTop="15dp"
        android:background="@color/colorPrimary"
        android:text="Ok"
        android:textColor="#FFF" />
    </LinearLayout>
</LinearLayout>

```

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from android studio, open one of your project's activity files and click Run  icon from the toolbar. Select your mobile device as an option and then check your mobile device which will display your default screen –



In the above result, it shown initial screen. Now click on button it will open custom dialog as shown below -



### **Alarm Clock Application:**

In this article, we are going to see how to build a much interesting app named **Alarm Setter**. Alarm plays a vital role in our day-to-day life. Nowadays alarm has become our wake-up assistant. Every mobile phone is associated with an alarm app. We will create this app using android studio. Android Studio provides a great unified environment to build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto because it provides a very large number of app building features and it is also very easy to use. A sample video is given below to get an idea about what we are going to do in this article. Note that we are going to implement this project using the **Java** language.

Video Player

00:00

00:27

### **Step by Step Implementation**

#### **Step 1: Create a New Project**

To create a new project in Android Studio please refer to [How to Create/Start a New Project in Android Studio](#). Note that select **Java** as the programming language.

#### **Step 2: Working with the activity\_main.xml file**

Navigate to the **app > res > layout > activity\_main.xml** and add the below code to that file. In this file, we have added two items '**TimePicker**' and '**ToggleButton**'. **TimePicker** is used to capture the alarm time and **ToggleButton** is added to set the alarm on or off. Initially, **ToggleButton** is set to off.

XML

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

```

<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <!--Added Time picker just to pick the alarm time-->
  <!--gravity is aligned to center-->
  <TimePicker
    android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center" />

  <!--Added Toggle Button to set the alarm on or off-->
  <!--ByDefault toggleButton is set to false-->
  <ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_margin="20dp"
    android:checked="false"
    android:onClick="OnToggleClicked" />

  <!--"OnToggleClicked" method will be implemented in MainActivity.java -->
</LinearLayout>

```

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called **MediaPlayer**.

### **Audio & video:**

Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, we have to call a static Method **create()** of this class. This method returns an instance of MediaPlayer class. Its syntax is as follows –

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
```

The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name **raw** and place the music file into it.

Once you have created the MediaPlayer object you can call some methods to start or stop the music. These methods are listed below.

```
mediaPlayer.start();
```

```
mediaPlayer.pause();
```

On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, you have to call **reset()** method. Its syntax is given below.

```
mediaPlayer.reset();
```

Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files. These methods are listed below –

Sr.No	Method & description
1	<b>isPlaying()</b> This method just returns true/false indicating the song is playing or not
2	<b>seekTo(position)</b> This method takes an integer, and move song to that particular position millisecond
3	<b>getCurrentPosition()</b> This method returns the current position of song in milliseconds

4	<b>getDuration()</b> This method returns the total time duration of song in milliseconds
5	<b>reset()</b> This method resets the media player
6	<b>release()</b> This method releases any resource attached with MediaPlayer object
7	<b>setVolume(float leftVolume, float rightVolume)</b> This method sets the up down volume for this player
8	<b>setDataSource(FileDescriptor fd)</b> This method sets the data source of audio/video file
9	<b>selectTrack(int index)</b> This method takes an integer, and select the track from the list on that particular index
10	<b>getTrackInfo()</b> This method returns an array of track information

#### Example

Here is an example demonstrating the use of MediaPlayer class. It creates a basic media player that allows you to forward, backward, play and pause a song.

To experiment with this example, you need to run this on an actual device to hear the audio sound.

Steps	Description
1	You will use Android studio IDE to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add MediaPlayer code.
3	Modify the res/layout/activity_main to add respective XML components
4	Create a new folder under MediaPlayer with name as raw and place an mp3 music file in it with name as song.mp3
5	Run the application and choose a running android device and install the application on it and verify the results

Following is the modified content of the xml **res/layout/activity\_main.xml**.

In the below code **abc** indicates the logo of tutorialspoint.com

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
    <TextView android:text="Music Palyer" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorials point"
        android:id="@+id/textView"
        android:layout_below="@+id/textview"
```

```

    android:layout_centerHorizontal="true"
    android:textColor="#ff7aff24"
    android:textSize="35dp" />
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:src="@drawable/abc"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/forward"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/pause"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_alignLeft="@+id/imageView"
    android:layout_alignStart="@+id/imageView" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/back"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_toRightOf="@+id/button2"
    android:layout_toEndOf="@+id/button2" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/rewind"
    android:id="@+id/button4"
    android:layout_alignTop="@+id/button3"
    android:layout_toRightOf="@+id/button3"
    android:layout_toEndOf="@+id/button3" />

<SeekBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/seekBar"
    android:layout_alignLeft="@+id/textview"
    android:layout_alignStart="@+id/textview"
    android:layout_alignRight="@+id/textview"
    android:layout_alignEnd="@+id/textview"
    android:layout_above="@+id/button" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Small Text"
    android:id="@+id/textView2"
    android:layout_above="@+id/seekBar"
    android:layout_toLeftOf="@+id/textView"
    android:layout_toStartOf="@+id/textView" />
<TextView

```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Small Text"
    android:id="@+id/textView3"
    android:layout_above="@+id/seekBar"
    android:layout_alignRight="@+id/button4"
    android:layout_alignEnd="@+id/button4" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/textView4"
    android:layout_alignBaseline="@+id/textView2"
    android:layout_alignBottom="@+id/textView2"
    android:layout_centerHorizontal="true" />
</RelativeLayout>

```

Following is the content of the **res/values/string.xml**.

```

<resources>
    <string name="app_name">My Application</string>
    <string name="back"><![CDATA[<]]></string>
    <string name="rewind"><![CDATA[<<]]></string>
    <string name="forward"><![CDATA[>>]]></string>
    <string name="pause">||</string>
</resources>


```

Following is the content of **AndroidManifest.xml** file.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sairamkrishna.myapplication" >
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.sairamkrishna.myapplication.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

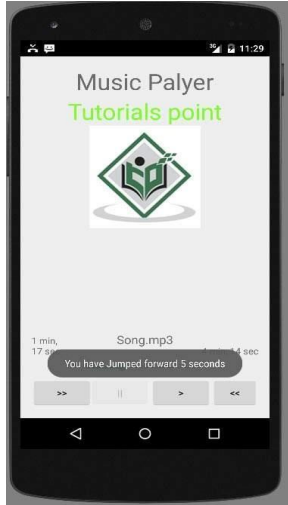
```

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Eclipse, open one of your project's activity files and click Run  icon from the toolbar. Before starting your application, Android studio will display following screens



By default you would see the pause button disabled. Now press play button and it would become disable and pause button become enable. It is shown in the picture below –

Up till now, the music has been playing. Now press the pause button and see the pause notification. This is shown below –



Your music would remain playing in the background while you are doing other tasks in your mobile. In order to stop it, you have to exit this application from background activities.

### **Drawing Application**

#### **Infinite Painter**

Infinite Painter is our top pick for the best drawing app Android can offer.

This app comes with over 160 brushes and allows you to create your own. It offers the opportunity to tailor the existing brushes and adapt them to your needs.

The app's drawing tools are designed to resemble painting over real paper textures. This feature allows the users to paint realistically using a digital device. It is practical if you prefer traditional painting, but you cannot transport your canvas everywhere.

Infinite Painter comes with several useful tools. You can paint in layers, use Photoshop blend modes, or even draw 3D cityscapes with perspective guides.

#### **Simple Draw Pro: Sketchbook**

Simple Draw is one of the best apps for beginners. It is also a good option for kids, as it is easy to use.

This paid app comes with different paint and pen sizes. It allows you to set different background colours or use a photo as the background.

Simple Draw allows you to share your sketches easily through social networks or messages. It supports JPG, PNG, or SVG vectors.

Simple Draw Pro

#### **Sketchbook**

Sketchbook is one of those all-in-one drawing apps that are worth a try. The app allows everything from simple sketches to professional artwork.

This award-winning app offers customizable tools, which are great for artists and professional illustrators. Sketchbook features different brush types, layers, and blend modes. All brushes are fully customizable.

This drawing app also offers guides and rulers that provide precision for a professional job. Its interface is elegant and simple. And it doesn't come with ads!

#### **ArtFlow**

Artflow is one of the best drawing apps available for download. It comes with over 100 brushes and tools. And it supports pressure-sensitive pens and pressure simulation. These features will make your painting experience more realistic.

This drawing app comes with some useful features such as Smudge, Fill, and Eraser tools. It features layer clipping and selection masks as well.

#### **ArtRage**

ArtRage is a paid app that offers a realistic painting experience. It features textures and painting effects that resemble physical paint.

ArtRage is one of the best drawing apps for Android. It features advanced tools that will let you measure how much paint you have used or how wet the paint is. Thanks to these tools, you can create realistic effects with watercolour and oil brushes.

#### **dotpict**

Dotpict is one of the most singular drawing apps for Android. It is designed to create pixelated drawings.



The app is very easy to use. You just need to pick a colour from the palette placed at the bottom of the screen and start drawing. After you have finished, you can share your art with other users or export your drawing in PNG format.

#### **Tayasui Sketches Lite:**

Tayasui is another one of the best drawing apps for Android devices. It comes with many tools perfect for digital artists.

Although it is sophisticated, it is also straightforward to use thanks to its user-friendly interface.

Tayasui is one of the best drawing apps you can find for creating realistic art. The app features several brushes, unlimited layers, patterns, and gradients. It comes with a feature to mix colours and get the perfect shade.

#### **Adobe Illustrator Draw:**

Adobe Illustrator Draw is a great drawing app for illustrators, graphic designers, and artists. It offers hundreds of professional tools to create vector artwork.

One of the advantages of this painting app is that it allows you to send your drawings in editable format to Photoshop, Illustrator, Capture One, or Photoshop Sketch. This feature will let you edit the picture on your desktop.

#### **Ibis Paint X:**

Ibis Paint X is one of the most versatile drawing apps for Android. It features many functional tools like layers, filters, fonts, blending modes, clipping masks and more.

This app comes with over 2000 brushes. It offers airbrushes, pencils, oil brushes, crayons, charcoal brushes and digital pens, among other professional brushes.

This app allows you to record your drawing process and share it with other users. It offers tutorial videos on its YouTube channel that will help you take full advantage of the software.

#### **InspirARTion:**

Our last recommendation for the best drawing apps for Android is InspirARTion. This app is not as popular as the rest on our list, but it's still a great option. It features a symmetry mode, a large variety of colours, and several brushes and templates.

#### **File:**

An abstract representation of file and directory pathnames.

User interfaces and operating systems use system-dependent *pathname strings* to name files and directories. This class presents an abstract, system-independent view of hierarchical pathnames.

#### **Reading and writing files in Android:**

Android separates the available storage into two parts, **internal** and **external**. These names reflect an earlier time when most devices offered built-in, non-volatile memory (internal storage) and/or a removable storage option like a Micro SD card (external storage). Nowadays, many devices partition the built-in, permanent storage into separate partitions, one for internal and one for external.

**Availability:** Internal storage is always available. External storage, on the other hand, is not. Some devices will let you mount external storage using a USB connection or other option; this generally makes it removable.

**Accessibility:** By default, files stored using the internal storage are only accessible by the app that stored them. Files stored on the external storage system are usually accessible by everything — however, you can make files private.

**Uninstall Behavior:** Files saved to the internal storage are removed when the app is uninstalled. Files saved to the external storage are not removed, even when the app is uninstalled. The exception to this rule is if the files are saved in the directory obtained by `getExternalFilesDir`.

If you're unsure which method to use, here's a hint: Use internal storage when you don't want the user or any other apps to access the files, like important user documents and preferences.

#### **Seeing the File in Device File Explorer**

**Open** lets you open the file in Android Studio.

**Save As...** lets you save the file to your file system.

**Delete** allows you to delete the file.

**Synchronize** synchronizes the file system if it's changed since the last run of the app.

**Copy Path** copies the path of the file to the clipboard.

Now that you know how to write a file to internal storage and use the Device File Explorer to see what files are available, it's time to learn how to make your app read them.

#### **Games:**

A definitive list of the best Android games you can pick up from Google Play, including some of the best free Android games on the market today

#### **Pocket Tactics**

Whether strategy, roguelikes, deck-builders, Gachas, MOBAs, or point-and-click, the range of great games available on the platform is mind-boggling. You could spend a good hundred days

playing only Android games, and after, a friend would still pop out of the woodwork to recommend another one that you missed.

### **Crashlands**

Craft, battle, and quest your way through the outlandish and vibrant world of Crashlands. You are stranded on an alien planet, and must hustle to retrieve your packages, as you become enmeshed in a nefarious plot of world domination. Learn recipes from the local sentient life, make new friends, discover ancient secrets and deadly bosses, and build your own home-away-from-home. If you love crafting and want to explore a huge world full of even bigger problems, this is the game for you.

### **Among Us:**

This social-deduction-based indie game has risen to astonishing popularity in recent months after it caught the attention of the gaming community via popular Twitch streamers. And with the developer recently reporting 200 million downloads, it doesn't look like it will slow down anytime soon. In Among Us, players must complete tasks to fix their ship, while also uprooting a murderer in their ranks before they are all wiped out. It's enjoyable, and best of all, it's free on mobile.

### **PUBG Mobile:**

The original battle royale is on mobile, but PUBG Mobile is a strange one. A lot of people enjoyed the original PC game, because of a certain level of what I shall call, jank. People liked the glitches and weird unexpected stuff that happened.

By comparison, PUBG Mobile actually plays very well, with regular PUBG Mobile updates adding fresh new content to the game, including the fresh, futuristic New Eden map. Undoubtedly one of the best Android games if you're hankering for a battle royale – and best of all, it's free to play.

### **Candy Crush Soda Saga:**

If you like match-three puzzlers and are looking for a fun, free experience, then we can't recommend Candy Crush Soda Saga enough. It's the King of the genre on mobile, providing hundreds of levels of satisfyingly crisp gameplay.

There are a load of great games like Soda Saga, make sure you check out our list of the best Candy Crush games to get your fix. Or head on over to our Candy Crush download guide to learn how to get this game on your device.

### **Device location:**

Manage your Android device's location settings

You can use location-based services such as getting better local search results, like commute predictions and nearby restaurants based on your phone's location, when you turn location on in settings.

**Google Location Accuracy for your Android device (a.k.a. Google Location Services):** To get a more accurate location for your phone, learn how to manage Location Accuracy.

**Emergency Location Service for your Android device:** Learn how to manage Android Emergency Location Service.

**Earthquake alerts for your Android device:** To get updates for nearby earthquakes on your phone, learn how to manage Earthquake alerts.

**Use location for time zone on your Android device:** To get time zone updates based on location, learn how to manage location for time zone.

**Location History for your Google Account:** To find and manage the places your phone has been, learn how to turn on Location History.

**Location Sharing for Google Maps:** To let others know where your phone is, learn how to share your real-time location via Google Maps.

**Location in Search:** To get more helpful results when you search on Google, learn how to manage location permissions for websites and apps.

**Wi-Fi scanning and Bluetooth scanning:** To help apps get better location info, learn how to scan for network or bluetooth devices.

### **Web Browser**

#### **Browser: Web3 Private Browser**

Browser is a newly launched browser which is constantly in Private Mode and nothing is tracked. It is the first browser in the world that offers the ability to purchase NFTs using Google Pay IAP (In-App Purchase). The NFTs unlock features and animations within the browser and there are usually give-aways, such as buying one NFT, you could get another one for free.



**Google Chrome:**

Google Chrome is easy to use, secure, and fast mobile browser for Android. This application offers an incognito mode to browse the internet without storing history. It is one of the best browser for Android that automatically optimizes websites when you are using a slow 2G network connection.

**Browser 4G:**

Browser 4G is a fast browser for Android devices. It enables you to surf social networking sites, a heavily loaded website with videos and photos.

**Smart Search & Web Browser:**

Smart Search & Web Browser is a portal that can be used to immediately find results from Bing, Google, YouTube, etc. This application also has a built-in QR code scanner.

**Aloha Browser Turbo:**

Aloha is a fast, free, full-featured web browser that offers maximum security and privacy. It is one of the best Android browser which offers free VPN and can block advertisements.

**Firefox Browser**

Firefox Browser is one of the best mobile browsers that allows you to surf the website privately. It can block more than 2000 trackers and prevent slowing down your speed.

**Kiwi Browser**

Kiwi Browser is an Android application that allows you to browse the internet, read the news, and watch videos with no problem. It is one of the best browsers for Android that offers a customizable mode to browse the web at night. The Browser supports private browsing.

**Microsoft Edge**

Microsoft Edge is an Android application that enables you to browse the internet by maintaining your privacy. It helps you to find and manage the needed content on the web with ease.

**Samsung Internet Browser**

Samsung Internet Browser is an internet browser that protects your privacy and security while browsing the web. It provides filters for blocking web content.

**Dolphin Browser**

Dolphin Browser is an app that can be used on Android phones. This application can load the content faster. It allows you to personalize search experience.

**Maxthon Browser**

Maxthon Browser is an app that allows you to view multiple websites in the same window. This program enables you to customize the theme. It can restore web pages in case of browser crashes.



**Web Browser & Explorer**

Web Browser & Explorer is a fast, secure, and easy application. This program offers a quick search using default search engines like Google, Bing, and Yahoo. It is one of the best mobile browser that supports flash player.

**Avast Secure Browser**

Avast Secure Browser is a fast web browsing application, VPN, and ad blocker. It can keep you hidden from ISPs, trackers, and hackers. This application can protect your password using the AES-256 encryption technique.

**Mint Browser – Video download, Fast, Light, Secure**

Mint Browser is an app for Android. It offers a faster and safer internet browsing experience. This tool allows you to download any video with one tap button.

