

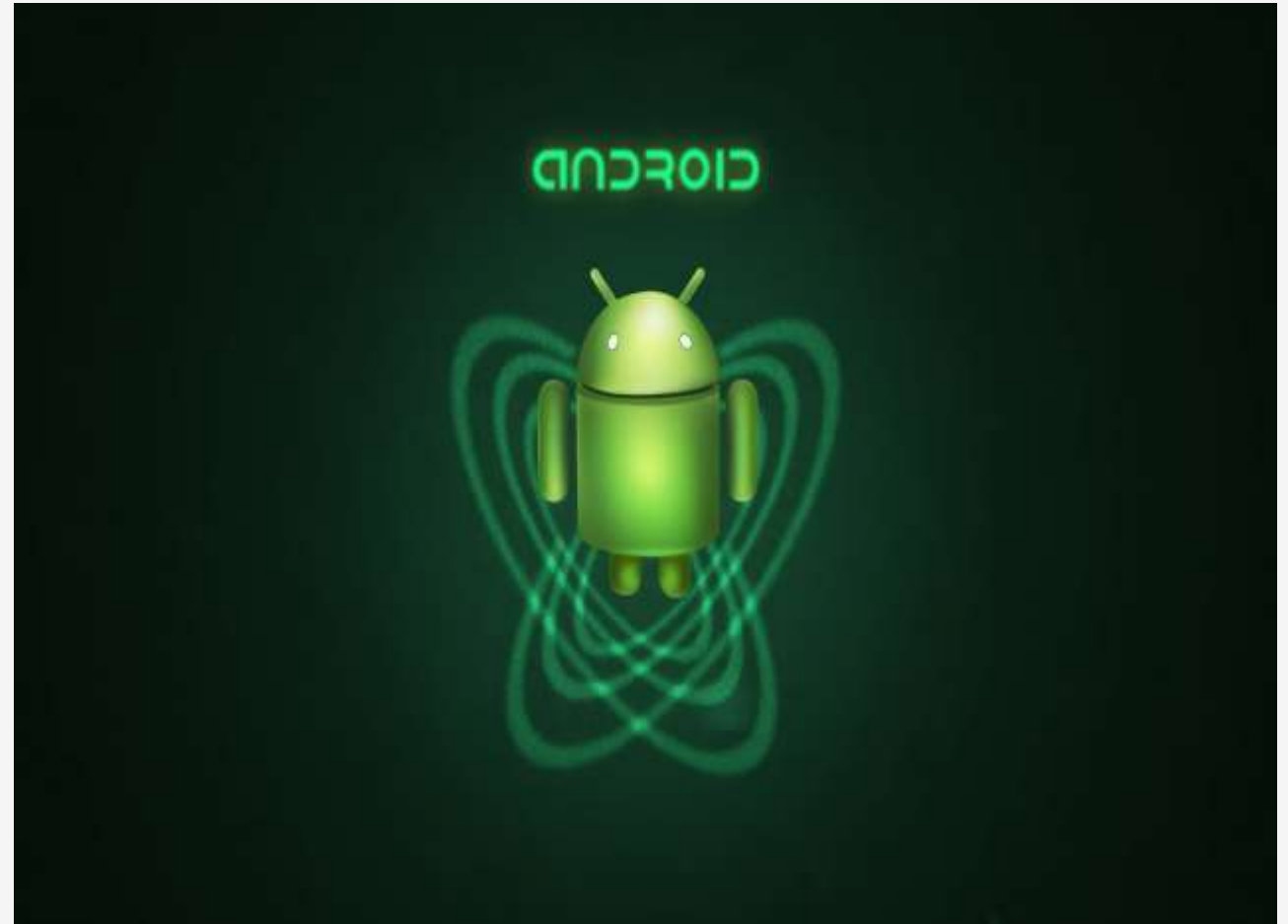
ANDROID OS & DEV ENVIRONMENT

- **Prof. Ashwani Mishra**
Assistant Professor
- **CSE Department**



Android is an open-source, Linux-based operating system designed mainly for smartphones, tablets, TVs, and wearables.

- **Complete Software Stack:** Includes OS, Middleware, and User Apps.
- **Why Popular?** Open-source, huge community, Google Play Store, Java/Kotlin support.
- **Example:** Apps like WhatsApp, Zomato, and Maps run on Android OS using built-in features like GPS and Camera.

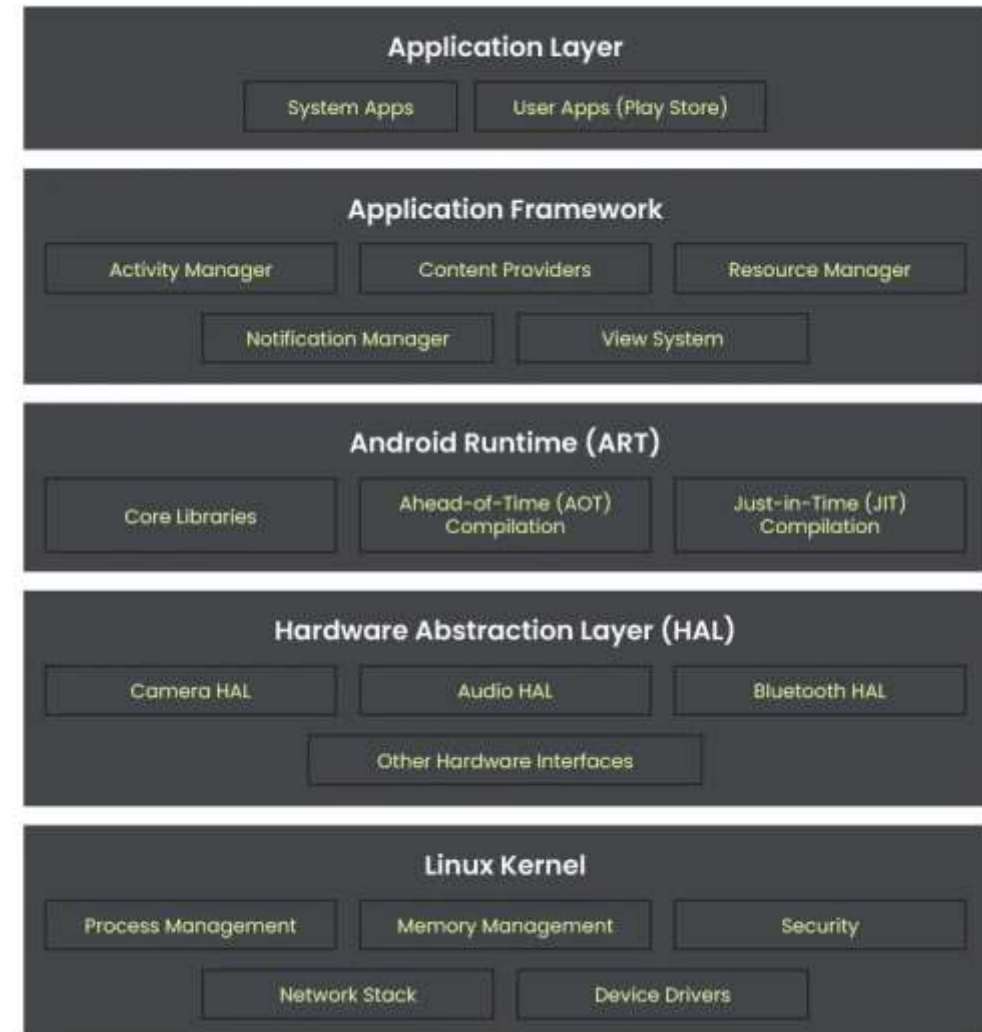


ANDROID ARCHITECTURE :

Android follows a layered architecture with 5 main components:

- 1 **Linux Kernel:** Hardware drivers, memory, power, security.
- 2 **HAL (Hardware Abstraction Layer):** Standard interfaces for hardware vendors (Camera, Audio).
- 3 **ART & Dalvik:** Runtime environment. ART converts bytecode to native code.
- 4 **Native C/C++ Libraries:** SQLite, OpenGL ES, SSL, WebKit.
- 5 **Application Framework:** High-level services (Activity Manager, Notifications)

Android Architecture Diagram



1. Linux Kernel - The Foundation

- **What is it?**
This is the lowest layer of the Android system, which interacts directly with the device's hardware.
- **Key Functions:**
 - **Hardware Drivers:** The kernel acts as a bridge between the hardware (like the camera, Bluetooth, Wi-Fi, display) and the software. It contains the drivers that allow the OS to communicate with the hardware components.
 - **Memory Management:** It manages the device's memory (RAM), deciding how much memory each application gets.
 - **Power Management:** It controls battery consumption and manages low-power states like sleep mode.
 - **Security:** It handles core-level security for the OS and enforces security permissions between applications and the system.
- **In Simple Terms:** The Linux Kernel is the "**engine**" that powers the entire Android system and controls the hardware.
-

- **2. Hardware Abstraction Layer (HAL) - The Hardware Translator**

- **What is it?**

The HAL acts as a "**middleman**" that sits between the Android framework and the device-specific hardware drivers.

- **Key Functions:**

- **Standard Interfaces:** The HAL provides a set of standard Java interfaces for hardware vendors (like Samsung, Google, OnePlus). A manufacturer building a camera for their phone writes a driver that implements the "camera" HAL interface. This tells the Android framework exactly how to talk to that specific camera hardware.
- **Example:** When an app wants to use the camera, it calls the standard "Camera" HAL API. The HAL then translates that call into commands for the specific camera driver (which is in the Linux Kernel).
- **In Simple Terms:** The HAL is like a **universal translator**. Apps speak a standard language (Java), and hardware components speak their own native languages. The HAL translates between them, so the app doesn't need to know the specific details of the hardware.



• **3. Android Runtime (ART) - The App Execution Engine**

- **What is it?**

This is the "**machine**" that runs Android applications (which are packaged in .apk or .aab files).

- **Dalvik (Legacy):** Used before Android 5.0 (Lollipop). It used a **Just-In-Time (JIT)** compiler, which compiled app code to machine code *while the app was running*, which could be slower.
- **ART (Current):** Used from Android 5.0 to the present.
- **Key Functions of ART:**
- **Ahead-of-Time (AOT) Compilation:** When you install an app, ART pre-compiles the app's bytecode into native machine code. This makes apps launch faster and run more smoothly.
- **Improved Performance and Garbage Collection:** ART is more efficient than Dalvik, leading to better battery life and a smoother user experience.
- **In Simple Terms:** ART is like a **smart interpreter** that translates an app's code into the device's native language *before* you even run it, making the app fast and e



• **4. Native C/C++ Libraries - The Brain Power**

- **What is it?**

These are powerful libraries written in C and C++ languages. Because they are "closer to the metal," they are extremely fast and efficient for complex tasks.

- **Key Libraries and Their Purposes:**

- **SQLite:** A lightweight and powerful database engine. Apps use it to store structured data locally (e.g., contacts, messages).
- **OpenGL ES:** A cross-platform API for rendering 2D and 3D graphics. Crucial for games and graphic-intensive applications.
- **WebKit/Chromium:** The rendering engine that powers web browsers (like Chrome). It displays web pages. (Note: WebKit has largely been replaced by Chromium in modern Android).
- **SSL (Secure Sockets Layer):** Provides secure communication over the internet (e.g., for HTTPS connections).
- **In Simple Terms:** These native libraries are Android's "**superpowers**" that handle complex tasks like graphics rendering and data management with high speed and efficiency.



• 5. Application Framework - The Developer's Toolkit

- **What is it?**

This is the layer that app developers directly interact with. It provides a rich set of **Java APIs** that developers use to build their apps.

- **Key Services and Managers:**

- **Activity Manager:** Manages the lifecycle of applications ("Activities") and controls how they are navigated (e.g. which app is in the foreground/background).
- **Notification Manager:** Allows all apps to display custom notifications in the status bar and notification shade.
- **Window Manager:** Manages the windows on the screen (the UI of your apps).
- **Content Providers:** Enables apps to share data with each other in a secure way (e.g., a contacts app allowing other apps to access phone numbers).
- **View System:** The building blocks for the user interface. Used to create lists, buttons, text boxes, and other UI elements.
- **Resource Manager:** Provides access to non-code resources like localized strings, images, and layout files.
- **In Simple Terms:** The Application Framework is a "**pre-built toolkit.**" Just as a carpenter has a kit with hammers and saws, an Android developer has this framework. It provides all the ready-made components needed to build an app without worrying about the complexities of the lower layers.

Versions

Version	Nickname	Release Date
1.0	Android	September 23,2008
1.1	Beta Android	February 9,2009
1.5	Cupcake	April 27,2009
1.6	Donut	September 15,2009
Android 1.x mobiles only		
Android 2.0/2.1	Éclair	October 26,2009
2.2 – 2.2.3	Froyo	May 20,2009
2.3 – 2.3.7	Gingerbread	December 6,2010
<ul style="list-style-type: none"> Android 2.x is also designed for mobiles but from 2.x android started supporting API. - By using Google API android application can interact with Google products such as Gmail, YouTube, google maps, navigation, google search engine, google Clouds 		

Version	Nickname	Release Date
Android 3.0 – 3.2.6	Honey Comb	February 22,2011
Android 3.x is specially designed for tablets. Started supporting Fragments		
Android 4.0 – 4.0.4	Ice-cream Sandwich	October 18,2011
4.1 – 4.3.1	Jellybean	July 9,2012
4.4 – 4.4.4	KitKat	October 31,2013
4.4.w supports for wearable devices like wrist-watch. From 4.x android started supporting mobiles & tablets application. It means single app can run in mobiles & tablets		

Version	Nickname	Release Date
5.0 – 5.1.1	Lollipop	November 12,2014
Android 5.x designed for Big Screens ie. TV		
Android 6.0 – 6.0.1	Marshmallow	October 5,2015
Android 6.0 designed for Automobiles(speed of car, km)		
Android 7.0 – 7.1.2	Nougat	August 22,2016
8.0 – 8.1	Oreo	August 21,2017
9.0	Pie	August 6,2018
10.0 (Official name)	Q (Internal code name)	September 3,2019
11.0	Red velvet Cake(Internal code name)	September 8,2020

Version	Nickname	Release Date
12	Snow Cone	October 04,2021
13	Android 13	August 15,2022
14	Android 14	October 04,2023
15	Android 15	September 3,2024
16	Android 16	June 10, 2025
17	Android 17	Upcoming 2026



Cupcake



Donut



Eclair



Gingerbread



Honeycomb



Ice Cream
Sandwich



Jelly Bean



KitKat



Lollipop



Marshmallow



Nougat



Oreo



Pie



Android



Android

Android 1.0 to 1.1(September 23, 2008)

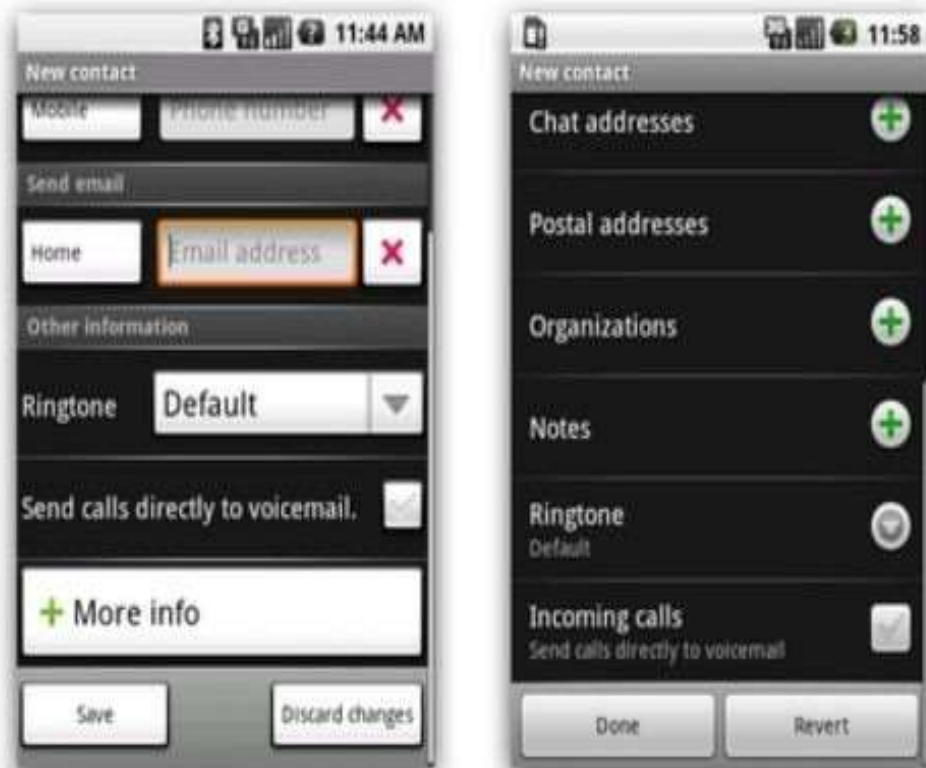


Features:

- Google Maps.
- Camera.
- Gmail, Contacts, and Google Synchronization.
- Web Browser.
- Wireless supports – Wi-Fi and Bluetooth.

Android version 1.5: Cupcake

April 20 2009



Features:

- New upload service on YouTube and Picasa like Uploading Videos and Photos.
- Supporting in MPEG-4, Video recording.
- Improving Web Browser-Copy and Paste facility.

Android version 1.6: Donut



r 15, 2009

Features:

- The main enhancement was a Power Control widget for managing Wi-Fi, Bluetooth, GPS, etc.
- Provided Gallery and Camera features with quick toggling features.
- Improve the speed in system apps.
- Introduction of the Quick Search Box.

Android versions 2.0 to 2.1: Éclair



Features:

- Update UI.
- Support Live Wallpaper.
- Support Bluetooth 2.1.
- Improve Google map.
- Minor API Changes.

Android version 2.2: Froyo May 20, 2010



Features:

- Support for Animated GIF.
- Wi-Fi Support Hotspot functionality.
- Speed improvements.
- Upload file support in the browser.
- Support numeric and alphanumeric passwords.

Android version 2.3: Gingerbread



Features:

- Improve Copy-Paste Facility.
- Updated UI design.
- VP8 and WebM video format support.
- Social Networking Supports.
- Easy use of the keyboard.
- Multiple camera support (usually known as a selfie camera nowadays).

Android 3.0/3.1/3.2 – Honeycomb



Features:

- Gmail App improvements.
- Updated 3D UI.
- Supports multiprocessors and hardware acceleration for graphics.
- Media Sync from SD Card.
- Google eBooks.
- Google Talk Video Chat.
- Support Adobe Flash in Browser.
- High-performance Wi-Fi Connections and Lock.
- Chinese handwriting.

Android version 4.0: Ice Cream



9, 2011

Features:

- Improved text input and spelling check.
- Wi-Fi direct (Sharing information using NFC).
- Photo Decor facility.
- Improved keyboard correction.
- Unlocking with face-fixing.
- Improved video recording resolution.
- Camera performance.
- Up to 16 tabs in the web browser.

Android versions 4.1 to

4.3: Jelly Bean



Features:

- Voice search.
- Panorama.
- Daydream as a screensaver.
- Power control.
- Improve camera application.
- Security enhancement.
- Voice typing.
- Multiple user accounts on tablets only.
- 4k resolution support.
- Supporting Bluetooth Low Energy.
- Bi-directional text and other language support.
- Support USB audio.
- Set the volume of incoming calls and show a message alert.
- Native emoji support.

Android version 4.4: KitKat

September 2, 2013.



Features:

- Screen Recording.
- KitKat adds a feature in 'Google now'. Its name is 'OK Google'. "OK, Google" allows access to Google to the users without touching your mobile phone.
- GPS Support.
- Offline music support.
- UI updates for google map navigation and alarm.
- Introduction of 'Emoji' to the google keyboard.

Android versions 5.0 and 5.1: Lollipop



Features:

- Support ART(Android Runtime).
- Improvement in UI.
- New material design.
- Notifications on the Lock screen.
- Revamped navigation bar.
- Multiple sim card support.
- High definition voice call.

Android version 6.0: Marshmallow



Features:

- Fingerprint authentication to unlock the screen.
- USB Type C support.
- Multi-window experiments (user can use two different apps in one screen).
- Save battery- 'Sleep Mode'.
- App permission model-OPT(send a request for permission).

Android versions 7.0 and 7.1: Nougat March 2016



Features:

- Provide multitasking.
- Inline reply to messages and notifications so you won't have to open up your Messenger application for quick replies.
- Providing multi-window mode.
- Improvements in storage manager.
- Display touch improvement.

Android version 8.0 and 8.1: Oreo



Features:

- Support PIP(Picture-in-Picture).
- Multi-display support.
- Google Play support.
- Adaptive icons.
- Revamped notification section(Users can set which notifications you want to show).

Android version 9: Pie August

0010

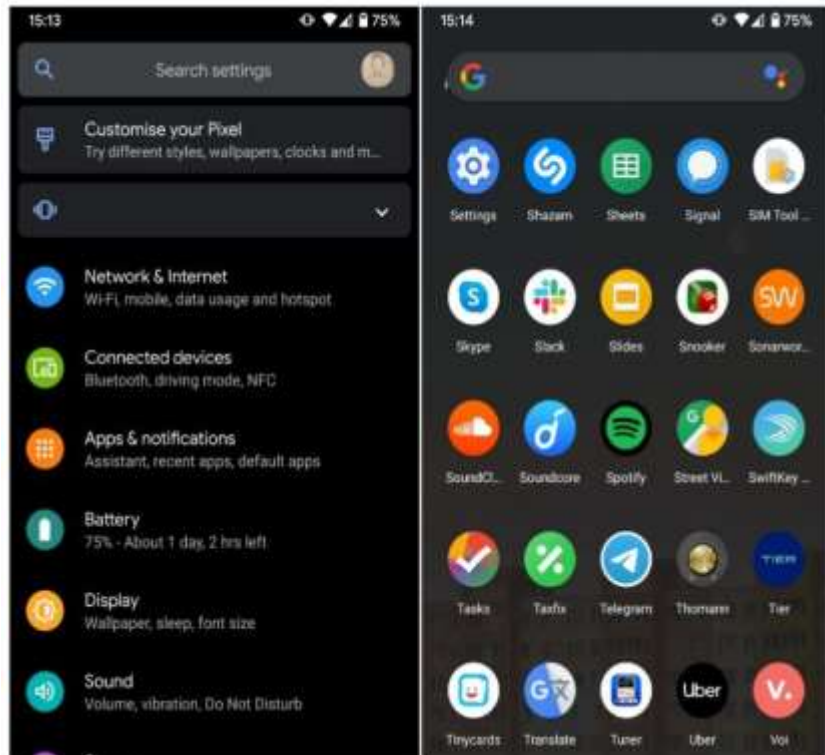


Features:

- New Gesture Navigation.
- Artificial intelligence (AI) Compatible.
- Adaptive Battery and Brightness.
- App Actions.
- New Screenshot Shortcuts.
- Easier Screen Rotation.
- Volume and Sound Improvement.
- Selectable Dark Mode.
- Slices.
- Improved Security Features.
- Digital Wellbeing.
- New Accessibility Menu.
- Easier Text Selection.
- More Notification Information.

Android version 10: Android Q

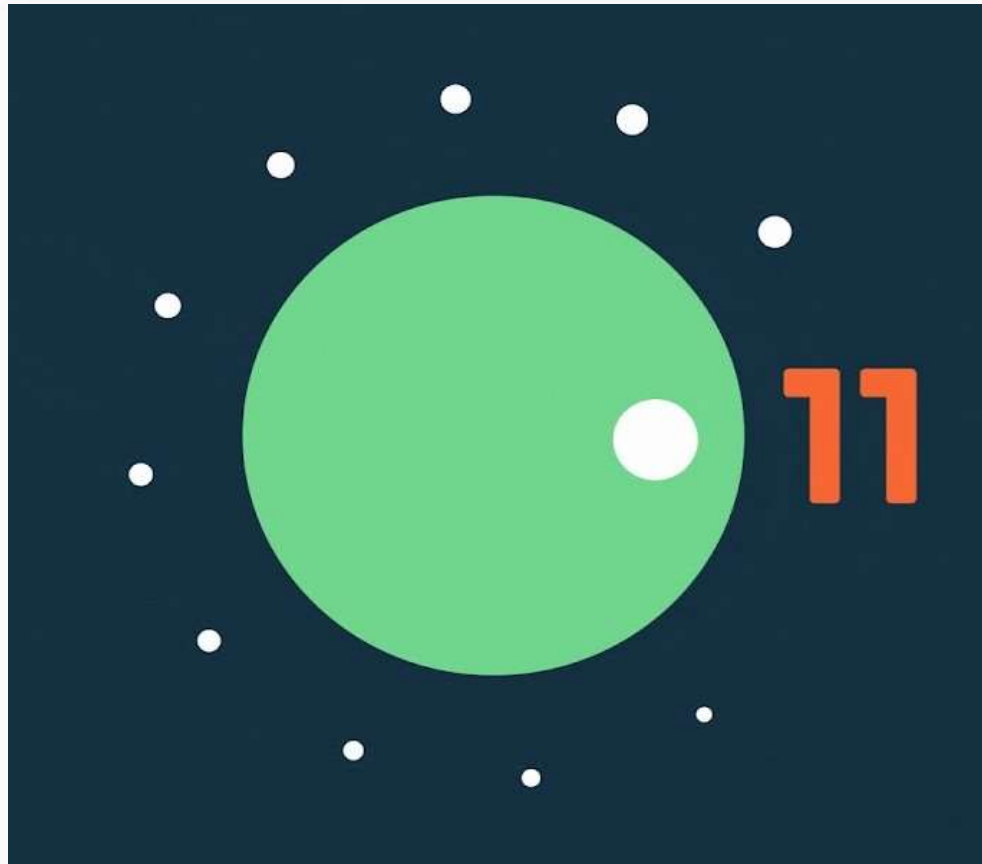
September 2, 2019



Features:

- Support for the upcoming foldable smartphones with flexible displays which is an upcoming rush.
- System-wide dark mode.
- Navigation control over gesture.
- Smart reply for all messaging apps.
- Support for Live caption.
- Better notification control.

Android version 11: Red Velvet Cake (Sept 2020)



Features:

- **Conversations & Bubbles:** Dedicated section in notifications for messaging apps.
- **Built-in Screen Recorder:** Record screen with audio without 3rd party apps.
- **Smart Home Controls:** Power menu includes quick access to smart home devices.
- **One-time Permissions:** Grant permission for location/mic/camera only once.
- **Wireless Android Auto:** Works wirelessly on all devices with 5GHz Wi-Fi.
- **Media Controls:** Moved to Quick Settings for easier access.
- **Notification History:** View dismissed notifications from the last 24 hours.

Android version 12: Snow Cone (Oct 2021)



Features:

- **Material You:** UI colors adapt automatically based on your wallpaper.
- **Privacy Dashboard:** View which apps accessed location, camera, or mic recently.
- **Mic & Camera Indicators:** Green dot status bar indicator when recording.
- **Scrolling Screenshots:** Capture full pages natively.
- **Approximate Location:** Option to share approximate instead of precise location.
- **Performance Improvements:** Faster core system services and reduced CPU usage.
- **One-handed Mode:** Shrinks screen for easier reachability.

Android version 13: Tiramisu (Aug 2022)



Features:

- **Themed Icons:** Third-party app icons can now match the system theme.
- **Per-App Language:** Set different languages for specific apps individually.
- **Photo Picker:** Grant access to specific photos rather than the entire gallery.
- **Clipboard Editor:** Visual preview and edit option after copying text/images.
- **Notification Permissions:** Apps must ask permission before sending notifications.
- **Bluetooth LE Audio:** Better audio quality and lower power consumption.
- **Spatial Audio:** Head tracking support for immersive sound.

Android version 14: Upside Down Cake (Oct 2023)



Features:

- **Lock Screen Customization:** Custom clocks, shortcuts, and wallpaper styles.
- **Ultra HDR Support:** Support for 10-bit HDR images.
- **Health Connect:** Built-in hub for storing health and fitness data securely.
- **Flash Notifications:** Camera flash or screen flash for incoming alerts.
- **Non-Linear Font Scaling:** Better readability for larger text sizes (up to 200%).
- **Regional Preferences:** Set system-wide units (Temperature, First day of week).
- **Predictive Back Gesture:** See destination before completing the back gesture.

Android version 15: Vanilla Ice Cream (Oct 2024)



Features:

- **Private Space:** A separate, locked profile for hiding sensitive apps.
- **Partial Screen Sharing:** Record or share only a single app window, not the full screen.
- **Edge-to-Edge Default:** Apps span the entire screen height by default.
- **Notification Cooldown:** Lowers volume for successive notifications from the same app.
- **Satellite Connectivity:** Native UI support for satellite messaging.
- **Battery Health:** More detailed insights into battery cycle count and health.
- **Low Light Boost:** Enhances camera preview brightness in dark conditions.

OHA

- Open Handset Alliance(OHA) was formed in November 2007.
- The OHA is the group that is in **charge of the Android Smartphone Operating System**. It was created by **Google**.

Purpose:

- Develop open standards for mobile devices.
- Provide a free operating system (Android).
- Promote innovation and reduce cost in the mobile industry



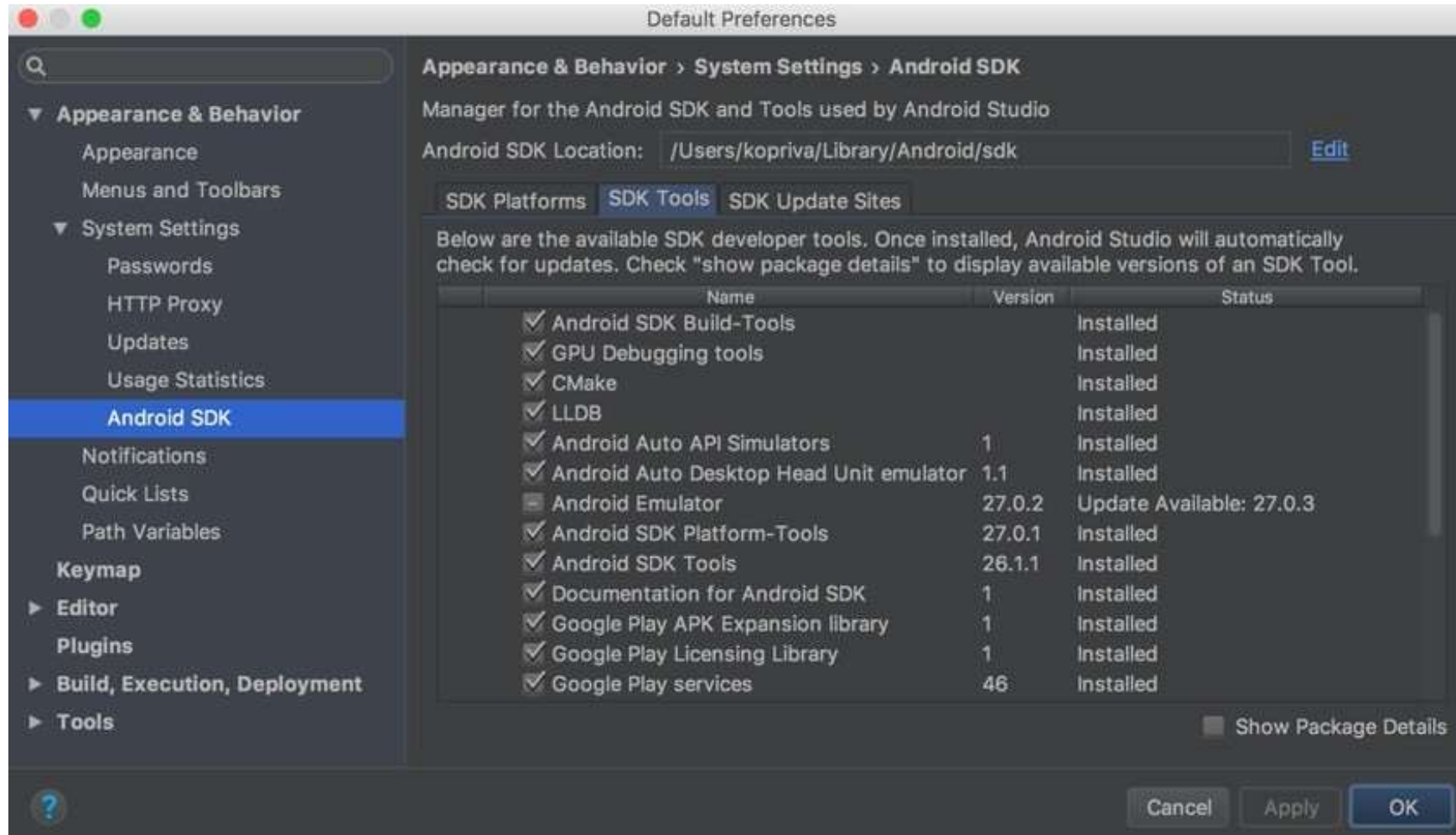
Dalvik VM

- Runtime environment for running android application.
- JVM is used to run high-end applications while DVM is used for small-end applications.
- DVM was first written by "Dan Bornstein"
- Unlike JVM, the DVM does not run .class files but it runs .dex files.
- .dex files are built from .class file at the time of compilation and provide higher efficiency in low resource environments.

Android SDK

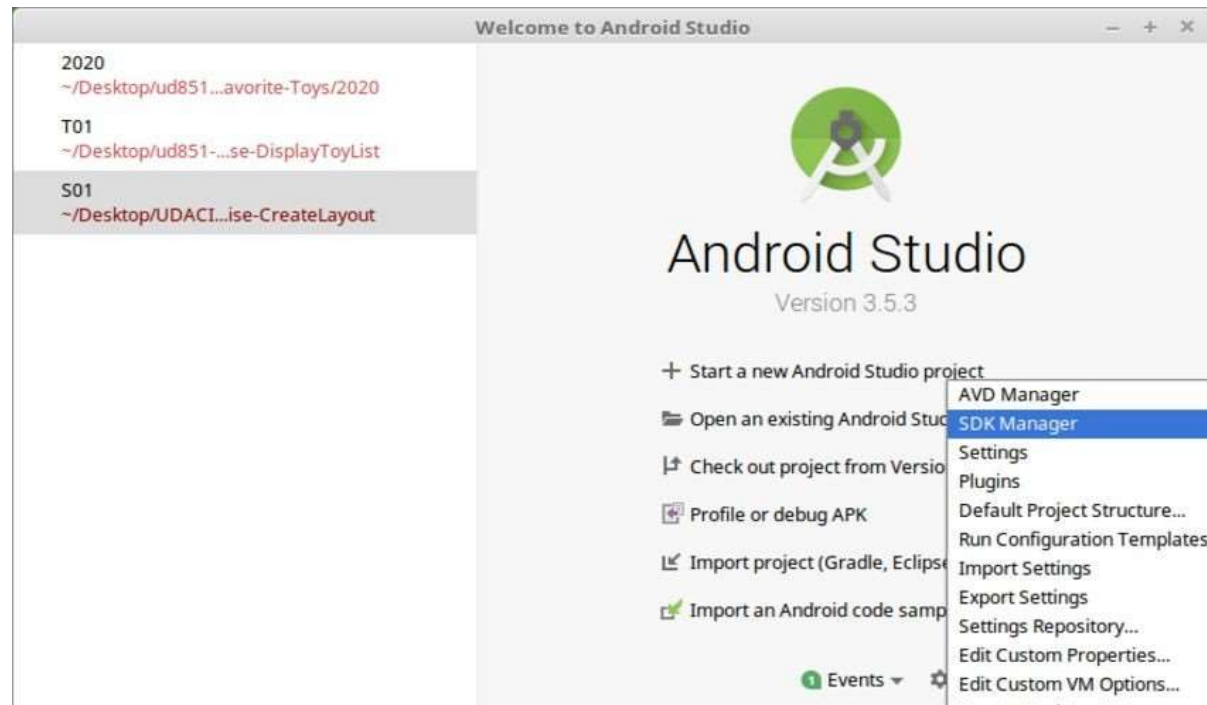
- Android SDK is a software development kit developed by Google for the Android platform.
- Android SDK comes bundled with Android Studio, Google's official integrated development environment (IDE) for the Android operating system
- The Android SDK is a **collection of software development tools and libraries required** to develop Android applications.
- Every time Google releases a new version of Android or an update, a corresponding SDK is also released which developers must download and install.
- The Android SDK comprises all the tools necessary to code programs from scratch and even test them. These tools provide a smooth flow of the development process from developing and debugging, through to packaging.
- The Android SDK is compatible with **Windows, macOS, and Linux**, so you can develop on any of those platforms.

Android SDK



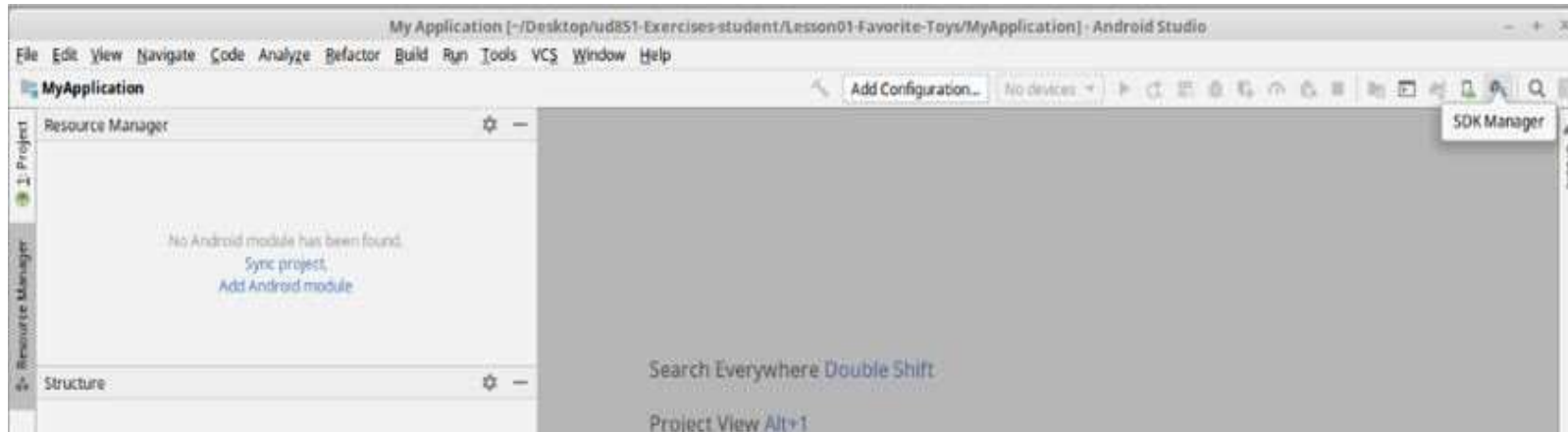
Android SDK

- To install the Android SDK from within Android Studio, first start Android Studio.
- From the Android Studio start page, select **Configure > SDK Manager**.



Android SDK

- If you already have Android Studio open, the SDK Manager icon is found on the top right corner, as shown below.



Android SDK

- The Android SDK consists of **an emulator, development tools, sample projects with source code, Google API, and the required libraries** to build Android applications

Android SDK

- Provided in the **android.jar** file that made up of several important packages.

Top-Level Package Name	Description
android.*	Android application fundamentals
dalvik.*	Dalvik Virtual machine support fundamentals
java.*	Core java classes and generic utilities for networking, security, math and so on
javax.*	Encryption support
junit.*	Unit-Testing support
org.apache.http.*	HTTP Protocol support
org.json	JavaScript Object Notation Support

Android SDK

➤ Few Popular Third-Party Android APIs:

- Com.google.android.gms.ads.* = Google Mobile Ads SDK Package.
- Com.google.android.gms.analytics.* = Google Analytics SDK for Android Package.
- Com.google.android.gms.gcm = Google Cloud Messaging for Android.
- Com.google.android.gms.appindexing = Google App Indexing Package.
- Com.google.android.gms.appinvite = Google App Invites Package
- Com.google.android.gms.games = Google Play Games Services Package
- Com.google.android.gms.fitness = Google Fit Package

Android SDK

- Android SDK provides many tools to design, develop, debug and deploy your applications.
 - Android Studio
 - Android SDK and AVD Managers
 - Android Emulator

Android Virtual Devices

- An **Android Virtual Device (AVD)** is a configuration that defines the characteristics of an Android device you want to simulate in the Android Emulator. This includes the device's hardware profile, system image, storage area, skin, and other properties

Android Virtual

Devices

➤ Key Components of AVD

- **Hardware Profile:** Defines the characteristics of a device as shipped from the factory. The Device Manager in Android Studio comes pre-loaded with certain hardware profiles, such as Pixel devices, and you can define or customize the hardware profiles as needed.
- **System Image:** A system image labeled with Google APIs includes access to Google Play services. The API level of the target device is crucial because your app doesn't run on a system image with an API level lower than the one required by your app¹.
- **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 card¹.
- **Skin:** An emulator skin specifies the appearance of a device. The Device Manager provides some predefined skins, and you can also define your own or use skins provided by third parties

Android Virtual Devices

➤ Creating and Managing AVDs

- To create a new AVD, you can use the Device Manager in Android Studio. Here are the steps:
- Open the Device Manager.
- Click "Create Device."
- Select a hardware profile and click "Next."
- Select the system image for a particular API level and click "Next."
- Change the AVD properties as needed and click "Finish"

Android Virtual Devices

- **Advantages of Using AVD**

- **Testing and Debugging:**

- The Android emulator provides almost all the functionality of a real device, making it easier to test and debug applications without needing physical hardware.

- **Predefined Configurations:**

- The emulator comes with predefined configurations for several Android phones, Wear OS, tablet, and Android TV devices.

- **Faster Data Transfer:**

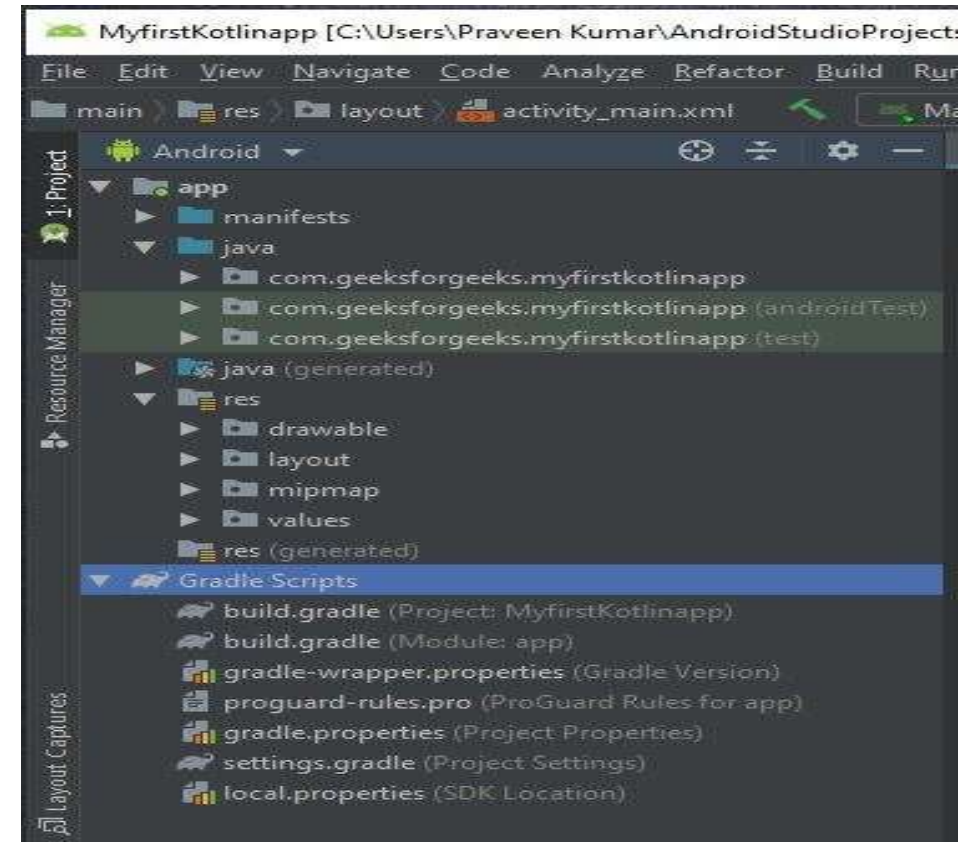
- Transferring data to the emulator is faster than to a real device connected through USB.

Android Virtual Devices

- **Limitations of AVD**
- **Performance:** Emulators can be slower compared to actual physical devices.
- **Accuracy:** Testing on an emulator is not as accurate as using a real device, especially for network and hardware-related activities.
- In summary, an AVD is a powerful tool for Android developers, allowing them to simulate various Android devices and test their applications in a controlled environment

Directory Structure of Android Application

- Android Studio is the official IDE (Integrated Development Environment) developed by the JetBrains community which is freely provided by Google for android app development. After completing the setup of Android Architecture we can create an android application in the studio. We need to create a new project for each sample application and we should understand the folder structure. It looks like this:



Directory Structure of Android Application

The android project contains different types of app modules, source code files, and resource files. We will explore all the folders and files in the android app.

Manifests Folder

Java Folder

res (Resources) Folder

 Drawable Folder

 Layout Folder

 Mipmap Folder

 Values Folder

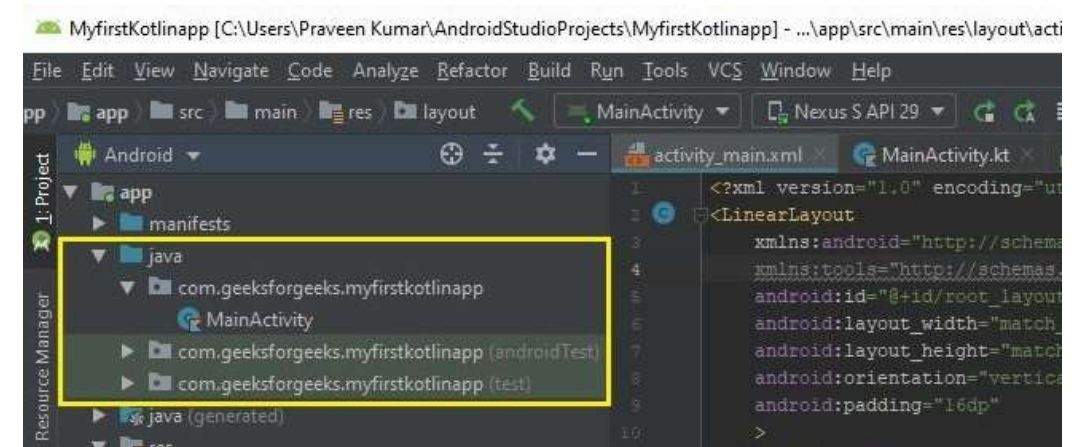
Gradle Scripts

Directory Structure of Android Application

- **Manifests Folder**
- Manifests folder contains **AndroidManifest.xml** for creating our android application. This file contains information about our application such as the Android version, metadata, states package for Kotlin file, and other application components. It acts as an intermediary between android OS and our application.

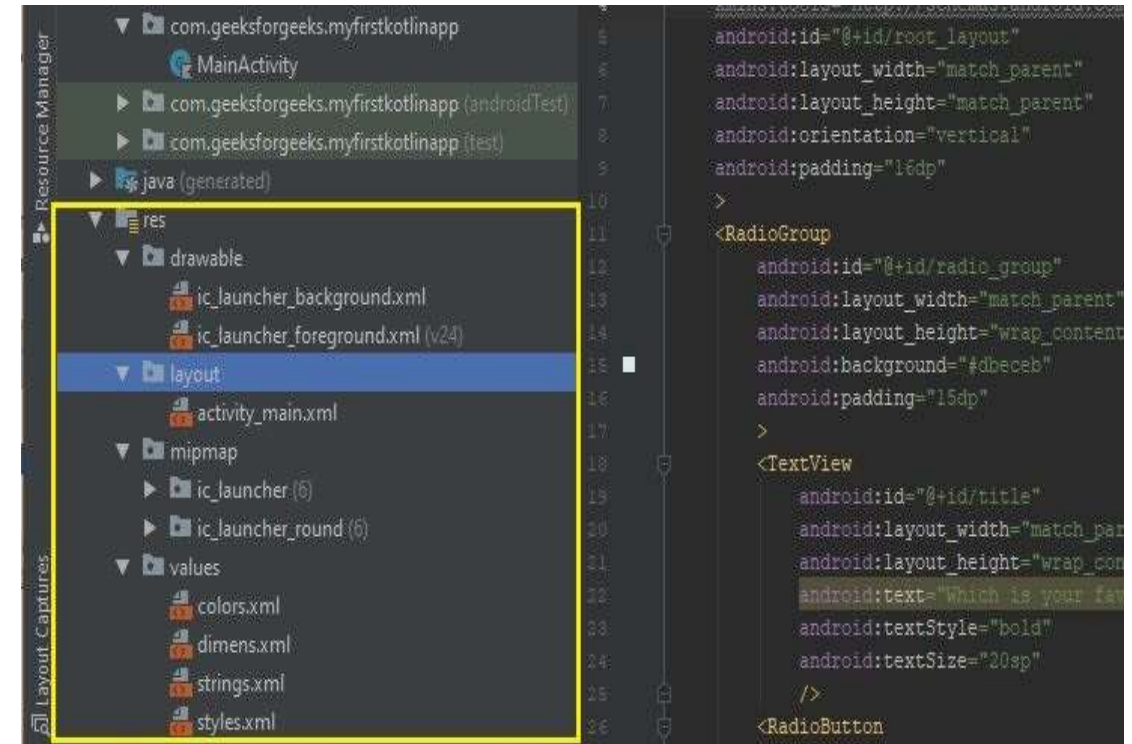
Directory Structure of Android Application

- **Java folder**
- The Java folder contains all the java and Kotlin source code (.java) files that we create during the app development, including other Test files. If we create any new project using Kotlin, by default the class file MainActivity.kt file will create automatically under the package name “com.geeksforgeeks.myfirstkotlinapp” as shown below.



Directory Structure of Android Application

- **Resource (res) folder**
- The resource folder is the most important folder because it contains all the non-code sources like images, XML layouts, and UI strings for our android application.



Directory Structure of Android Application

- **res/drawable folder**

It contains the different types of images used for the development of the application. We need to add all the images in a drawable folder for the application development.

- **res/layout folder**

The layout folder contains all XML layout files which we used to define the user interface of our application.

Directory Structure of Android Application

- **res/mipmap folder**

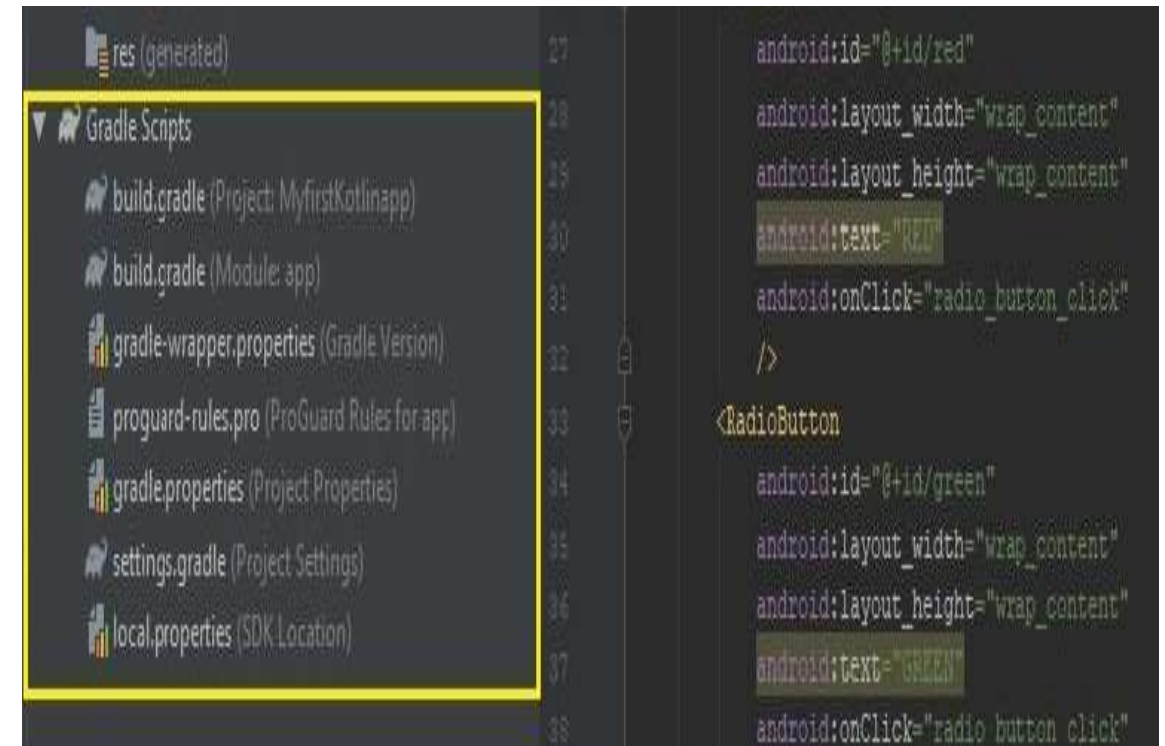
This folder contains launcher.xml files to define icons that are used to show on the home screen. It contains different density types of icons depending upon the size of the device such as hdpi, mdpi, xhdpi.

- **res/values folder**

Values folder contains a number of XML files like strings, dimensions, colors, and style definitions.

Directory Structure of Android Application

- **Gradle Scripts folder**
- Gradle means automated build system and it contains a number of files that are used to define a build configuration that can be applied to all modules in our application. In build.gradle (Project) there are buildscripts and in build.gradle (Module) plugins and implementations are used to build configurations that can be applied to all our application modules.



Android Manifest file

- Every project in Android includes a Manifest XML file, which is **AndroidManifest.xml**, located in the root directory of its project hierarchy. The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements.
- This file includes nodes for each of the [Activities](#), [Services](#), [Content Providers](#), and [Broadcast Receivers](#) that make the application, and using [Intent Filters](#) and Permissions determines how they coordinate with each other and other applications.

Android Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.geeksforgeeks"
    android:versionCode="1"
    android:versionName="1.0"
    android:installLocation="preferExternal">

    <uses-sdk
        android:minSdkVersion="18"
        android:targetSdkVersion="27" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Android Manifest file

- **1. manifest**
- The main component of the AndroidManifest.xml file is known as manifest. Additionally, the packaging field describes the activity class package name. It must contain an <application> element with the xmlns:android and package attribute specified.

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

    <!-- manifest nodes -->

    <application>

    </application>

</manifest>
```


Android Manifest file

- **2. uses-sdk**
- It is used to define a minimum and maximum SDK version by means of an API Level integer that must be available on a device so that our application functions properly, and the target SDK for which it has been designed using a combination of minSdkVersion, maxSdkVersion, and targetSdkVersion attributes, respectively. It is contained within the <manifest> element.

```
<uses-sdk  
    android:minSdkVersion="18"  
    android:targetSdkVersion="27" />
```

Android Manifest file

- **3. uses-permission**
- It outlines a system permission that must be granted by the user for the app to function properly and is contained within the `<manifest>` element. When an application is installed (on Android 5.1 and lower devices or Android 6.0 and higher), the user must grant the application permissions.

```
<uses-permission  
    android:name="android.permission.CAMERA"  
    android:maxSdkVersion="18" />
```

Android Manifest file

- **4. application**
- A manifest can contain only one application node. It uses attributes to specify the metadata for your application (including its title, icon, and theme). During development, we should include a debuggable attribute set to true to enable debugging, then be sure to disable it for your release builds. The application node also acts as a container for the Activity, Service, Content Provider, and Broadcast Receiver nodes that specify the application components. The name of our custom application class can be specified using the android:name attribute.

```
<application
    android:name=".GeeksForGeeks"
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@drawable/gfgIcon"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@android:style/Theme.Light"
    android:debuggable="true"
    tools:targetApi="31">

    <!-- application nodes -->

</application>
```

Android Manifest file

- **5. uses-library**
- It defines a shared library against which the application must be linked. This element instructs the system to add the library's code to the package's class loader. It is contained within the <application> element.

```
<uses-library  
    android:name="android.test.runner"  
    android:required="true" />
```

Android Manifest file

- **6. activity**
- The Activity sub-element of an application refers to an activity that needs to be specified in the AndroidManifest.xml file. It has various characteristics, like label, name, theme, launchMode, and others. In the manifest file, all elements must be represented by <activity>. Any activity that is not declared there won't run and won't be visible to the system. It is contained within the <application> element.

```
<activity  
    android:name=".MainActivity"  
    android:exported="true">  
</activity>
```

Android Manifest file

- **7. intent-filter**
- It is the sub-element of activity that specifies the type of intent to which the activity, service, or broadcast receiver can send a response. It allows the component to receive intents of a certain type while filtering out those that are not useful for the component. The intent filter must contain at least one <action> element.

```
<intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
  
    <category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>
```


Android Manifest file

- **8. action**
- It adds an action for the intent-filter. It is contained within the <intent-filter> element.
- **9. category**
- It adds a category name to an intent-filter. It is contained within the <intent-filter> element.

```
<action android:name="android.intent.action.MAIN" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
```

Android Manifest file

- **11. uses-features**
- It is used to specify which hardware features your application requires. This will prevent our application from being installed on a device that does not include a required piece of hardware such as NFC hardware, as follows:

```
<uses-feature android:name="android.hardware.nfc" />
```

Android Manifest file

- **12. permission**
- It is used to create permissions to restrict access to shared application components. We can also use the existing platform permissions for this purpose or define your own permissions in the manifest.

```
<permission  
    android: name="com.paad.DETONATE_DEVICE"  
    android:protectionLevel="dangerous"  
    android:label="Self Destruct"  
    android:description="@string/detonate_description">  
</permission>
```

THANK YOU