Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

Android Operating System and Development Environment

Study Guide

Mr. Tushar T. Jakhaniya
Assistant Professor
CSE, Parul Institute of Technology
Parul University

Parul® University
Vadodara, Gujarat
NAAC GRADE A++

Contents

1. Android Operating System and Development Environment

1.1 Introduction

The Android operating system has become the most widely used mobile platform across the globe. Today, hundreds of millions of devices in over 190 countries run on Android. By the end of 2021, it had captured nearly 71% of the global market share, and its growth continues to expand steadily.

Android was originally developed by the Open Handset Alliance (OHA) as an open-source platform based on a modified version of the Linux kernel and other open-source technologies. Google supported the project from its early stages and later acquired Android Inc. in 2005. The first Android-powered device was released in September 2008, marking the beginning of a new era in mobile computing.

Android's dominance in the mobile OS market can be attributed to its user-friendly interface, extensive customization options, strong community support, and the participation of numerous manufacturers producing Android-compatible devices. This has created a high demand for Android application development and skilled developers in the industry.

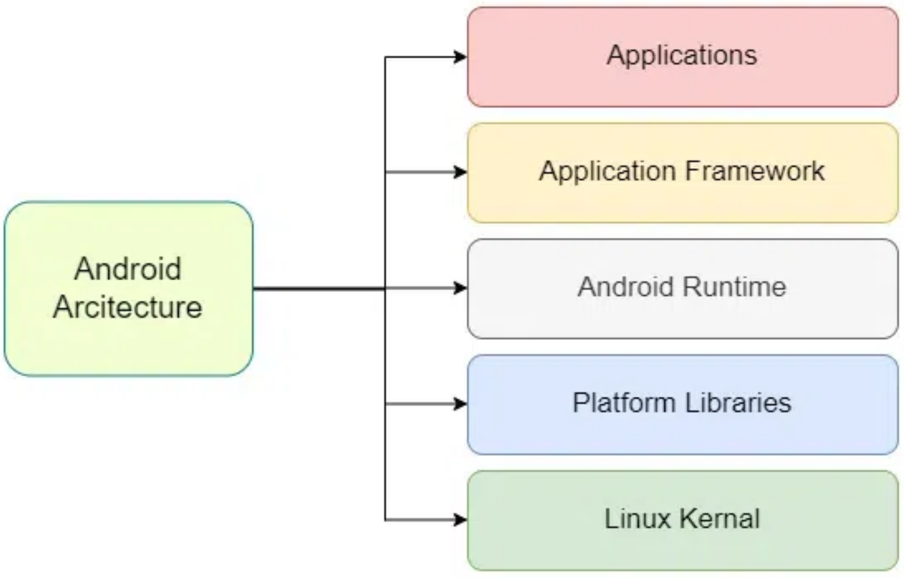
1.2 Android Architecture

Android architecture contains a different number of components to support any Android device's needs. Android software contains an open-source Linux Kernel having a collection of a number of C/C++ libraries which are exposed through application framework services. Among all the components Linux Kernel provides the main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide a platform for running an Android application.

Components of Android Architecture

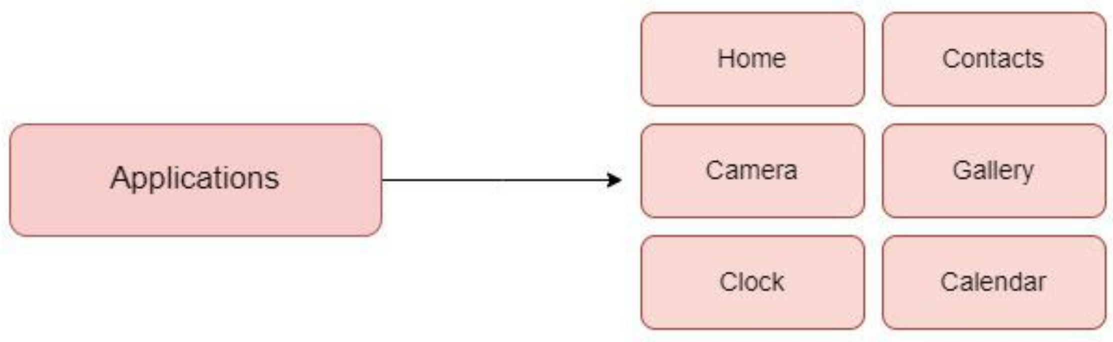
The main components of Android architecture are the following: -

1 Applications
2 Application Framework
3 Android Runtime
4 Platform Libraries
5 Linux Kernel



1. Applications

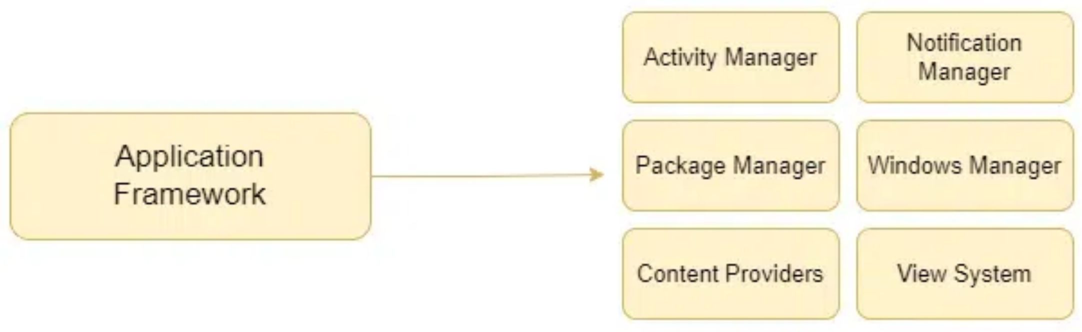
Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third-party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.



2. Application framework

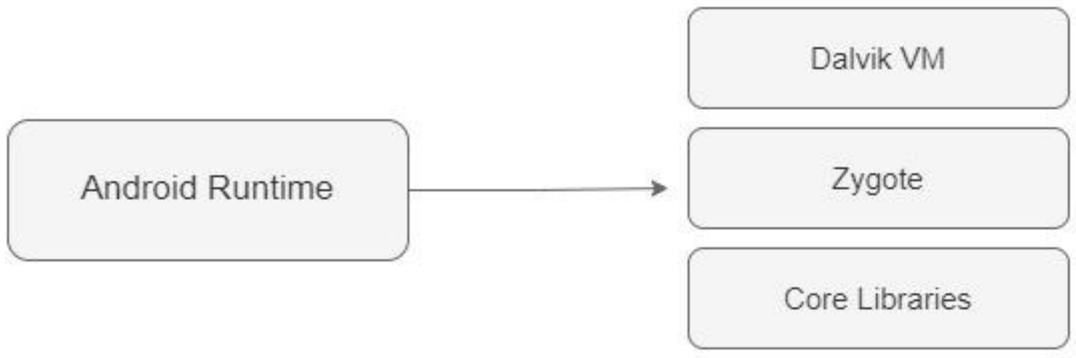
The Application Framework is a core part of Android that gives developers the tools and services they need to build apps. It provides access to device features like hardware, screen display, and system resources. It includes several important services that make it easier to build powerful and consistent Android apps without having to create everything from scratch. The services are as follows:

Activity Manager - Manages the app's activities and their life cycle (like opening, pausing, or closing screens).
Notification Manager - Allows apps to show alerts or updates to the user.
Package Manager - Keeps track of all the apps installed on the device.
Window Manager - Handles the placement and appearance of windows on the screen.
Content Providers - Help apps share data with other apps (like contacts or photos).
View System - Controls how things (like buttons or text) appear on the screen.



3. Application runtime

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine (DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries. Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.



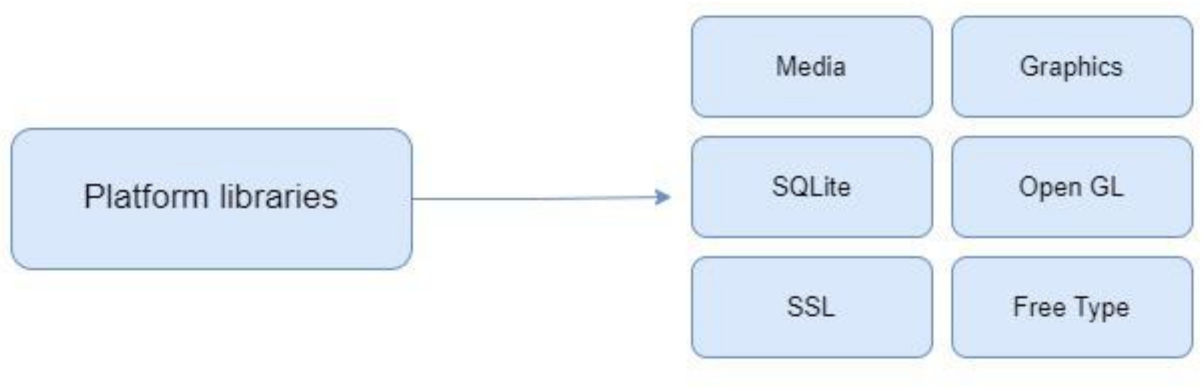
4. Platform libraries

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- Media library provides support to play and record an audio and video formats.
- Surface manager responsible for managing access to the display subsystem.
- SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.

- SQLite provides database support and FreeType provides font support.
- Web-Kit This open source web browser engine provides all the functionality to display web content and to simplify page loading.

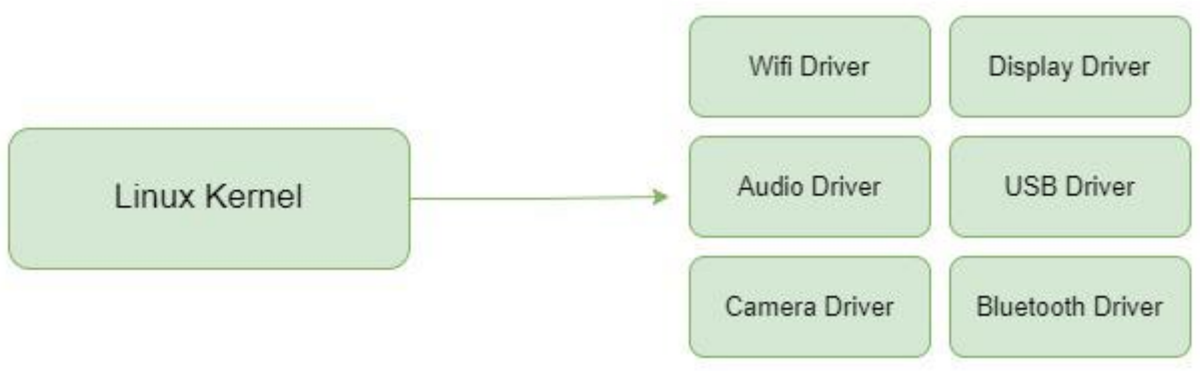
SSL (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.



5. Linux Kernel

Linux kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime. The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc. The features of Linux kernel are:

- Security: The Linux kernel handles the security between the application and the system.
- Memory Management: It efficiently handles the memory management thereby providing the freedom to develop our apps.
- Process Management: It manages the process well, allocates resources to processes whenever they need them.
- Network Stack: It effectively handles the network communication.
- Driver Model: It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.


Linux Kernel
Wifi Driver
Display Driver
Audio Driver
USB Driver
Camera Driver
Bluetooth Driver

1.3 Versions, Features

Version	Codename / Name	API Level	Release Year	Key Features / Highlights
1.0	—	1	2008	First public Android release; basic apps like Browser, Gmail, Camera, Maps, YouTube.
1.1	—	2	2009	Minor updates; bug fixes; API improvements.
1.5	Cupcake	3	2009	First named version; on-screen keyboard, widgets, video recording.
1.6	Donut	4	2009	Support for multiple screen sizes; improved camera and search.
2.0 / 2.1	Eclair	5–7	2009	Google Maps Navigation; multiple account support; live wallpapers.
2.2	Froyo (Frozen Yogurt)	8	2010	Wi-Fi hotspot; Flash support; performance boost (JIT compiler).
2.3	Gingerbread	9–10	2010	Improved UI; NFC support; better gaming performance.
3.0 / 3.1 / 3.2	Honeycomb	11–13	2011	Tablet-only version; redesigned UI for larger screens.
4.0	Ice Cream Sandwich	14–15	2011	Unified phone and tablet UI; face unlock; data usage control.
4.1 / 4.2 / 4.3	Jelly Bean	16–18	2012–2013	Google Now; smoother UI ("Project Butter"); expandable notifications.
4.4	KitKat	19–20	2013	Optimized for low-end devices; “OK Google” voice activation.
5.0 / 5.1	Lollipop	21–22	2014–2015	Material Design; 64-bit support; ART runtime introduced.
6.0	Marshmallow	23	2015	Runtime app permissions; Doze battery mode; fingerprint API.
7.0 / 7.1	Nougat	24–25	2016	Multi-window support; quick app switching; Vulkan graphics API.
8.0 / 8.1	Oreo	26–27	2017	Picture-in-picture; notification dots; background process limits.
9.0	Pie	28	2018	Gesture navigation; adaptive battery; digital wellbeing.
10	—	29	2019	Dark theme; enhanced privacy controls; foldable device support.
11	—	30	2020	Chat bubbles; screen recording; better privacy and permission handling.

Version	Codename / Name	API Level	Release Year	Key Features / Highlights
12 / 12L	—	31–32	2021	Material You design; improved haptics; optimized for large screens.
13	—	33	2022	Per-app language settings; improved notifications and privacy.
14	—	34	2023	Satellite connectivity; predictive back gestures; health connect integration.
15	—	35	2024	Improved performance; better background app management; advanced AI features.
16	— (in development / early release 2025)	36	2025	Enhanced AI integration, new privacy dashboard, seamless multi-device connectivity.

Key features of Android include:

- User Interface: The user interface of the Android operating system is straight forward, and these features make it very user friendly.
- Multiple Language Support: Android supports multiple languages in its operating system and one can change the language very easily based on one's requirement, the international languages supported are English, Germany, Chinese, Dutch, French, German, Japanese, Korean, Russian, and many more also some native language of India is also Supported Like Hindi, Marathi, Gujarati, Punjabi and many more.
- Multi-tasking: Android provides support to run apps and services in the background with ease which allows the users to use multiple apps at the same time.
- Connectivity: Android has extensive support to the connectivity and it supports connectivity such as Wi-Fi, Bluetooth, Hotspot, CDMA, GSM, NFC, VOLTE, UBB, VPN, 3G network band, and 4G Network Band.
- Extensive Application Support: Android have Play store which is used as the major tool to download and update applications on the operating system, however, one can download the installer (often called as APK file) and install it manually, but it is not much recommended as third party applications could be prone to some security breach in the smartphones.

Key Features of Android -10

1. Smart Replies
Automatic Suggestion to reply to texting app, like if a message contains a phone number, then we can click on it and save/call the contact number. In this feature, you not only get some suggested responses but also get recommended actions related to the message. For Example, if someone sends a message regarding a meeting schedule then it pops to set a reminder for the meeting in the calendar, if one gets a location in the message then it pops to open maps for it, If someone asks to join for a meeting then it prompts us to reply with yes or no. The smart reply feature was initially started by Gmail and is later on carried to android also. These features not only work for system apps but also work on some third-party applications like Signal.

2. Sound Amplifier
Native Support for the inbuilt amplifier for the users, earlier we had to download some third-party applications to do the same. With Sound Amplifier feature one can hear the better sound and can also boost the sound by fine-tuning the sound by the help of features such as "boost" and "fine-tune" on either of the ears or both of the ears of the headphone. This feature helps the user to hear the music/recorded voice note more clearly when a large amount of hindrance can be observed in the surrounding.

3. Gesture Control
Earlier this feature was not only available for IOS Applications but with the android 10 update, we can get the use it on the windows operating system. Some popular gesture control features include:

- **Go back**: In order to go back one needs to swipe from the left or right edge of the screen, and while doing so an arrow shows up on the screen showing that you are redirected to the previous page of the device.
- **Go home:** To go to the home screen one needs to swipe from the bottom edge of the screen in the upward direction and you will be redirected to the home screen.

4. Live Captions
Using machine learning algorithms to generate the live captions of any video you are watching even when the video doesn’t have subtitles. To turn Live Caption on or off we need to Press the volume button and in the volume control one needs to tap on the live caption subtitles. Once turned one, we can see the live caption appears at the bottom of the speech being played in the media on your device, and that too without ever needing Wi-Fi or Mobile Data. We can also move the caption box by touch and hold and then drag up or down to enhance the viewing experience and if we drag the caption to the extreme bottom then we can turn off Live Caption.

5. Dark Mode
The dark mode is provided in the operating system and the new dark mode support many apps that will reduce the battery drain on OLED phones. Android 10 dark theme uses true black to keep the battery alive for a longer duration, also it changes the view of the google applications like Google Drive, Google Photos, Google Calendar, to improve the viewing experience at night and to put less strains on the eyes for night users. It not only works with google apps but also it is expanding it's features to some third-party applications such as Instagram and Reddit, also the support of dark mode applications is increasing day by day.

1.4 OHA

1.
2. The acronym OHA in the context of Android refers to the **Open Handset Alliance**. It is a consortium of technology and mobile companies, led by Google, that was founded to develop open standards for mobile devices and the Android mobile operating system.
3.
4. Key facts about the OHA

- **Formation**: The OHA was established on November 5, 2007, with 34 original members.
- **Mission**: Its goal was to create a mobile platform that was open, free, and would accelerate innovation in the mobile industry.
- **Android development**: The consortium is responsible for the development of Android, an open-source mobile phone platform built on the Linux kernel.
- **Membership**: The OHA's membership includes a wide range of companies from the mobile supply chain, such as handset manufacturers, application developers, mobile network operators, and chip makers. Prominent members have included Google, HTC, Samsung, Motorola, T-Mobile, and Intel.
- **The anti-fragmentation agreement**: A crucial aspect of the alliance is that members agree to not produce devices that run on incompatible or competing "forks" of the Android operating system. This contractual obligation is designed to promote a unified Android platform and prevent fragmentation.
- **The first Android device**: The HTC Dream, also known as the T-Mobile G1, was the first commercially available phone running Android. It launched in October 2008, approximately a year after the OHA's founding

5.

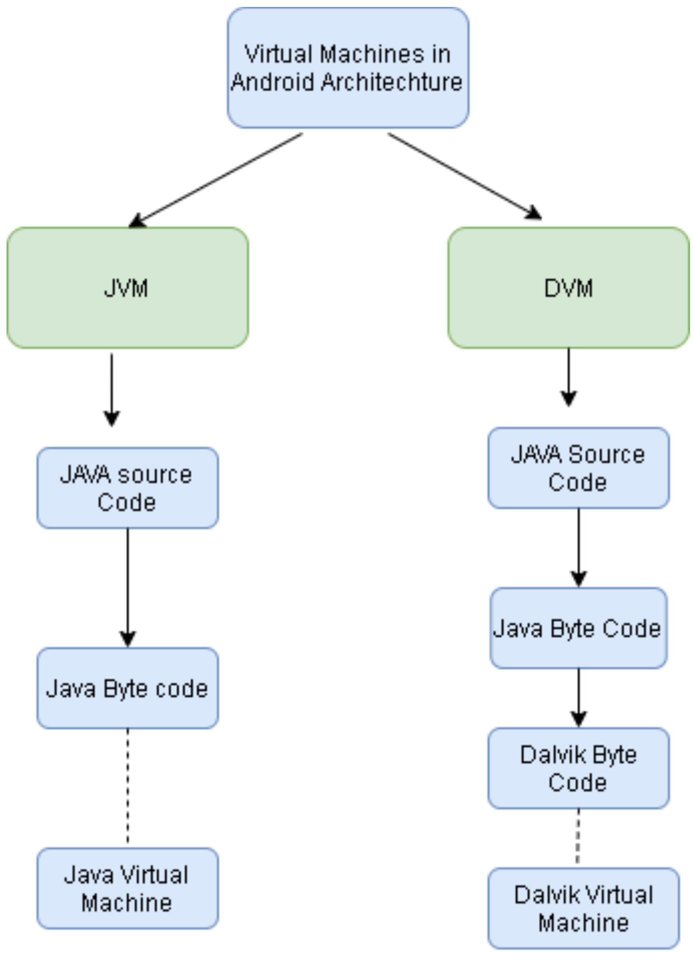
1.5 Dalvik VM (DVM)

Dalvik Virtual Machine Register Based VM The language is designed by Dan Bornstein, some cool features and nature of Apple Swift iOS Programming Language he gave it a try with the initial compiler implementations examples contributed from other Google engineers at its Android mobile phone platform. The Dalvik virtual machine was named in that way after Bornstein, a resident of the nearby village "Dalvík", born on ''Eyjafjörður''

What is Dalvik Virtual Machine?

Dalvik Virtual Machine (DVM) is the custom program introduced for Android apps. It takes the Java code and creates an optimized version of it in a file with .dex(extension) which is known as Dalvik executable. This format allows the apps to run quickly with fewer resources, i.e., on mobile phones and low-memory, slower devices. This is different from a typical Java Virtual Machine (JVM), as it optimized especially for Android to run apps that use less memory and are compatible with the millions of devices running on various versions.

Simple working of DVM



1 Java Source Code
- The original development of Android apps is mostly done using Java programming language that developers use to write the instructions your app executes.

2 Compilation
- We need to convert this textual Java code into form understandable by computer. This is commonly referred to as compilation. This Java bytecode is an executable version of your code.

3 Conversion to Dalvik Bytecode
- The Java bytecode is transformed into a different format known as Dalvik byte code. They use a tool called dx, to do this and make the code run smoothly on Android devices.

4 Dalvik Virtual Machine (DVM)
- Last, but not least the Dalvik bytecode is run on the Dalvik Virtual Machine (DVM). DVM has the special ability to run the applications on mobile devices which have limited capabilities of memory and battery.

Features of Dalvik Virtual Machine

- Register-Based Architecture: Dalvik is a register-based virtual machine unlike the traditional stack based one (like JVM). This helps minimize overhead that would be incurred by constantly pushing and popping values off the stack making execution faster, especially on low memory mobile devices.
- Optimized for Mobile Devices: It is an optimizing BSJ virtual machine that was designed specifically to run on the Android operating system, with low memory and power requirements. It virtualizes multiple apps by running a separate instance of the VM for each application as it does best in simultaneously handling many applications.
- Dex Bytecode Execution: Dalvik uses. dex (Dalvik Executable) files—these compile the source into a more compact bytecode in order to make it better-suited for low-memory environments. We compile multiple Java classes into a single. dex file so that it was just 63% of the original size, meaning a small parse and load time on mobile devices.
- Just In Time (JIT) Compilation: Dalvik was released from Android 2.2 (Froyo) and later included a JIT compiler to translate the bytecode into machine code directly at execution runtime, which could eventually boost frequently executed codes performance.
- Garbage Collection: The above code ran on the DALVIK which has automatic garbage collection to make sure that apps will not consume large number of resources and thus there is no effect in system stability.
- Support for Multithreading: Dalvik is multithreading mobile machine can have multiple threads at the same time for an application to work on. This feature is one of the most required things when it comes to mobile apps e.g., background process along with foreground user tracking.

Advantages of DVM (Dalvik Virtual Machine)

- the Android operating system only.
- In DVM executable is APK.
- Execution is faster.
- From Android 2.2 SDK Dalvik has it's own JIT (Just In Time) compiler.

- Applications are given their own instances.

Disadvantages of DVM (Dalvik Virtual Machine)

- DVM supports only Android Operating System.
- For DVM very few Re-Tools are available.
- App Installation takes more time due to dex.
- More internal storage is required.

Application of Dalvik Virtual Machine

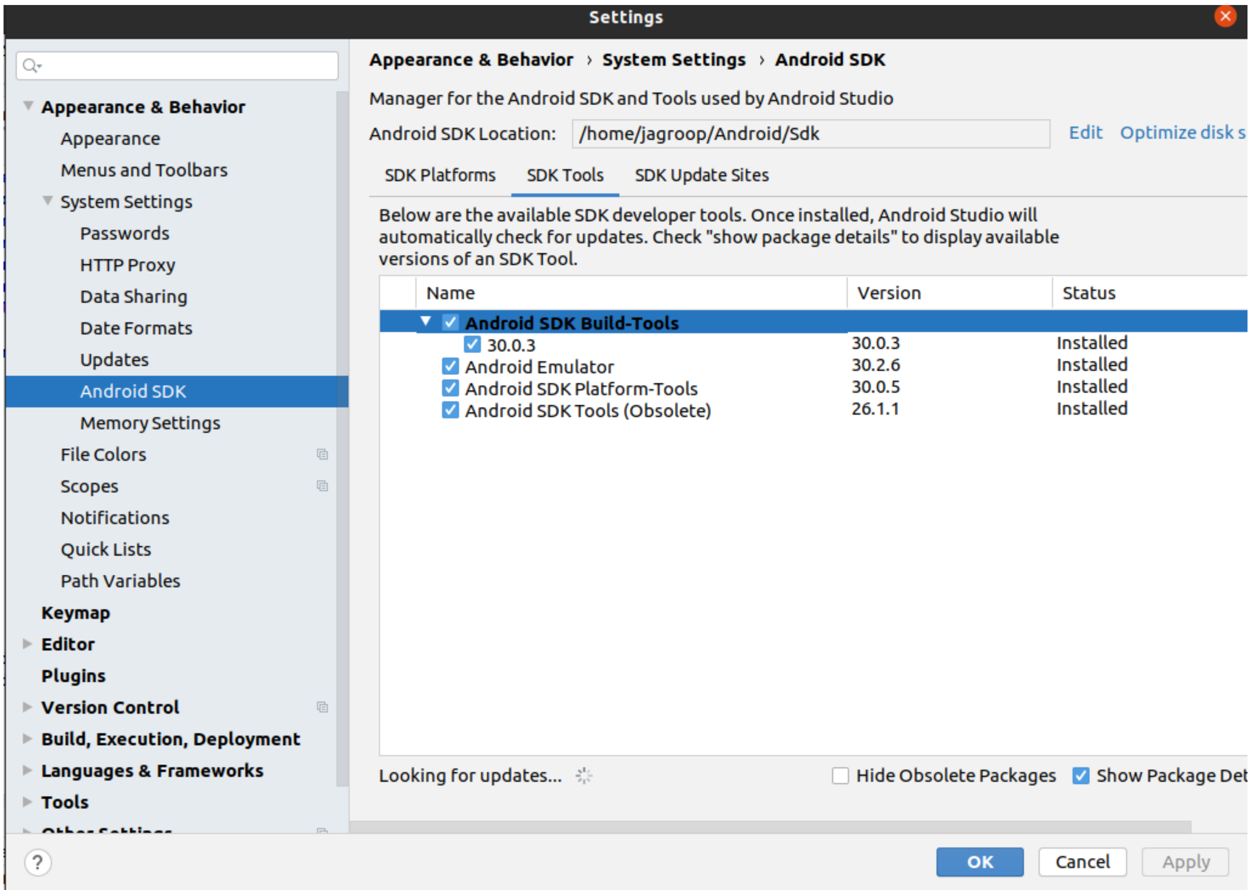
- Running Android Apps**:** It improves how apps run by using processing power and less memory, ensuring smooth work.
- Support for Multitasking**:** It allows multiple android apps to run at the same time without interrupting each other speed, making multi-tasking smoother on mobile devices.
- Memory Management**:** It manages memory more efficiently than older system, making sure that apps can run well even on devices with limited memory.

1.6 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.

About Android SDK:

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.



Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/jagroop/Android/Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		
✓ 30.0.3	30.0.3	Installed
✓ Android Emulator	30.2.6	Installed
✓ Android SDK Platform-Tools	30.0.5	Installed
✓ Android SDK Tools (Obsolete)	26.1.1	Installed

Looking for updates...

Components of Android SDK:

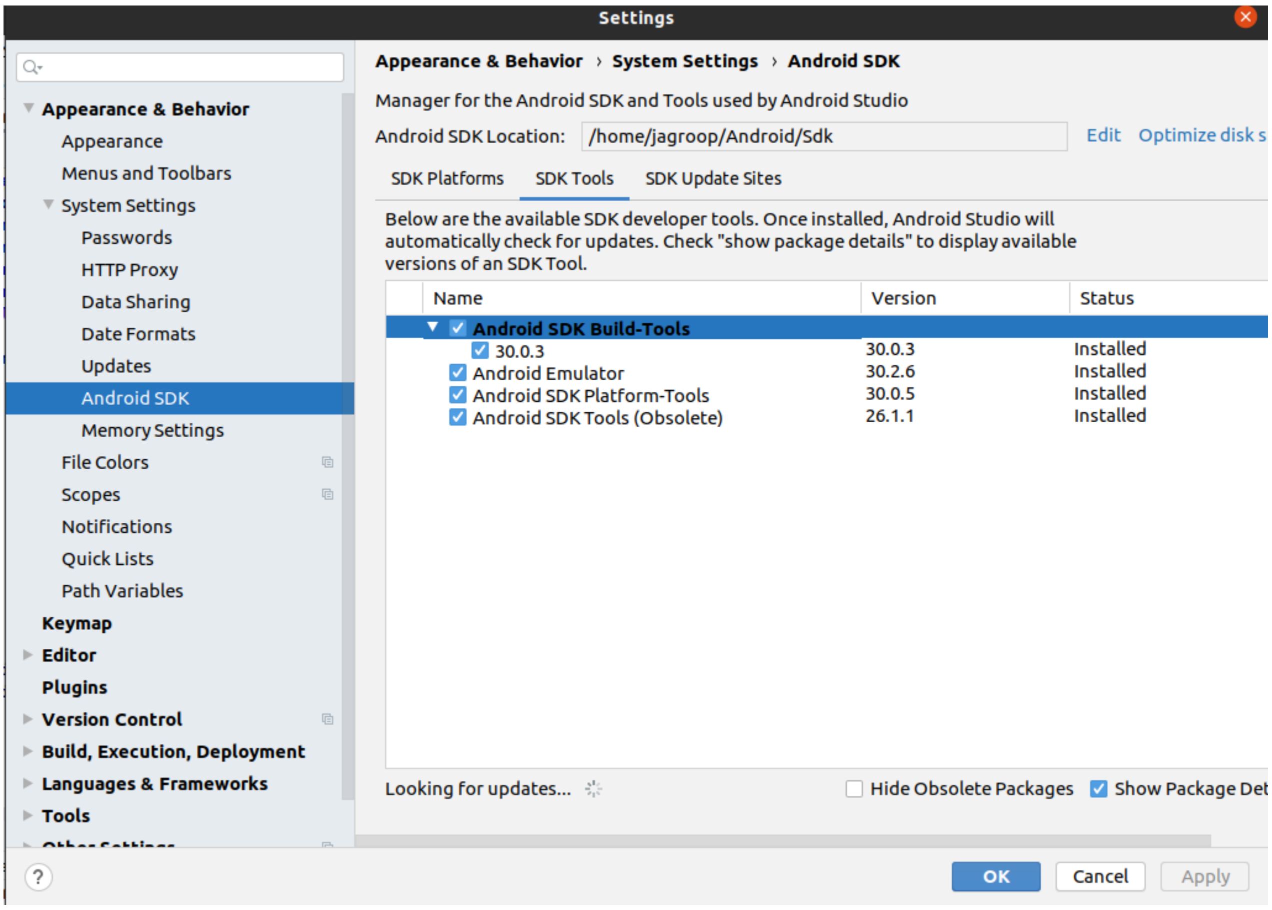
Android SDK Components play a major role in the Development of Android applications. Below are the important components:

1. 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 tool.
- Android Emulator.
- Android SDK Platform-tools.
- Android SDK Tools.

These are shown below:



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/jagroop/Android/Sdk   Edit   Optimize disk s

SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		
✓ 30.0.3	30.0.3	Installed
✓ Android Emulator	30.2.6	Installed
✓ Android SDK Platform-Tools	30.0.5	Installed
✓ Android SDK Tools (Obsolete)	26.1.1	Installed

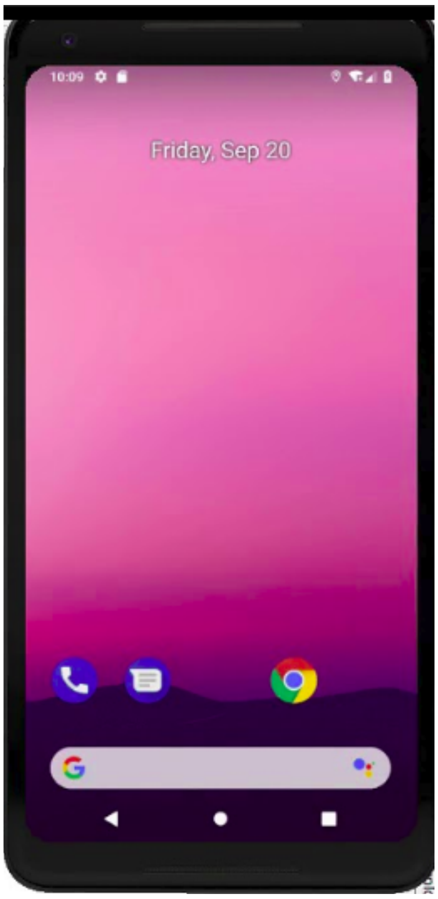
Looking for updates...   ☐ Hide Obsolete Packages   ✓ Show Package Det

2. 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.

3. 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. Thus, it simply means that without needing any physical device Android SDK component "Android Emulator" provides a virtual device on the System where we run our application. The emulator's come with the configuration for Various android phones, tablets, Wear OS, and Android TV devices.



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

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

But there is one disadvantage of this emulator is that. It is very slow when System's PC has less RAM. It works fine when a maximum GB of RAM is present on our device.

4. 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:

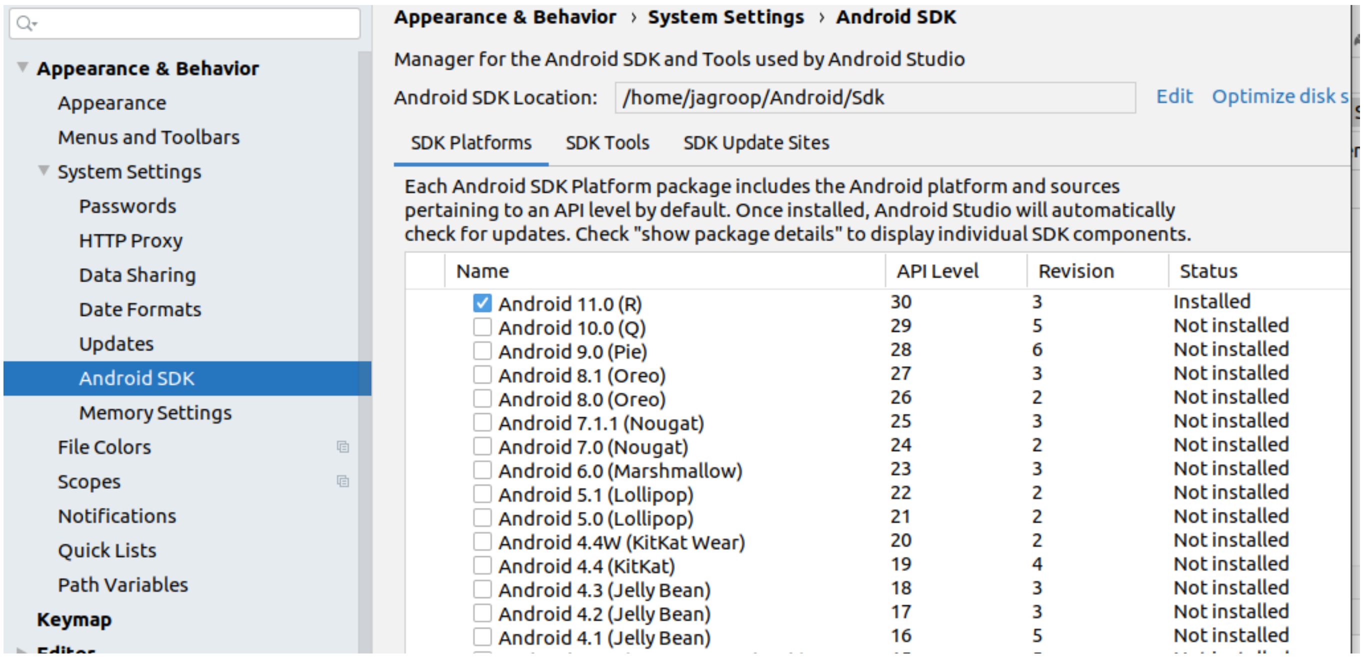
- Android Debug Bridge (ADB), is a command-line tool that helps to communicate with the device. It allows us to perform an action such as Installing App and Debugging App etc.
- Fastboot allows you to flash a device with a new system image.
- Systrace tools help to collect and inspect timing information.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

5. 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.

6. SDK Platforms

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



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/jagroop/Android/Sdk

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

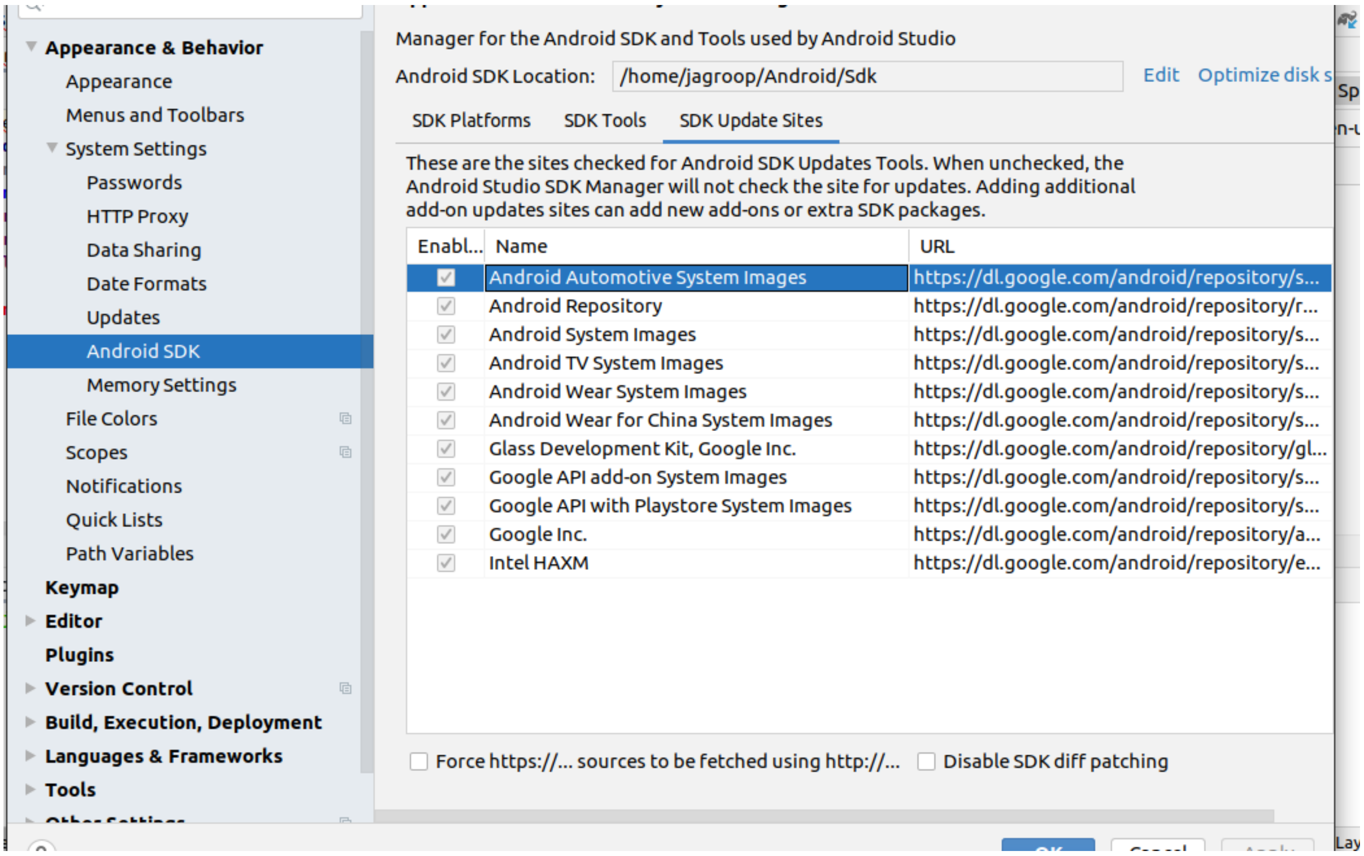
Name	API Level	Revision	Status
✓ Android 11.0 (R)	30	3	Installed
Android 10.0 (Q)	29	5	Not installed
Android 9.0 (Pie)	28	6	Not installed
Android 8.1 (Oreo)	27	3	Not installed
Android 8.0 (Oreo)	26	2	Not installed
Android 7.1.1 (Nougat)	25	3	Not installed
Android 7.0 (Nougat)	24	2	Not installed
Android 6.0 (Marshmallow)	23	3	Not installed
Android 5.1 (Lollipop)	22	2	Not installed
Android 5.0 (Lollipop)	21	2	Not installed
Android 4.4W (KitKat Wear)	20	2	Not installed
Android 4.4 (KitKat)	19	4	Not installed
Android 4.3 (Jelly Bean)	18	3	Not installed
Android 4.2 (Jelly Bean)	17	3	Not installed
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.

7. 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.



Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/jagroop/Android/Sdk

These are the sites checked for Android SDK Updates Tools. When unchecked, the Android Studio SDK Manager will not check the site for updates. Adding additional add-on updates sites can add new add-ons or extra SDK packages.

Enabl...	Name	URL
✓	Android Automotive System Images	https://dl.google.com/android/repository/s...
✓	Android Repository	https://dl.google.com/android/repository/r...
✓	Android System Images	https://dl.google.com/android/repository/s...
✓	Android TV System Images	https://dl.google.com/android/repository/s...
✓	Android Wear System Images	https://dl.google.com/android/repository/s...
✓	Android Wear for China System Images	https://dl.google.com/android/repository/s...
✓	Glass Development Kit, Google Inc.	https://dl.google.com/android/repository/gl...
✓	Google API add-on System Images	https://dl.google.com/android/repository/s...
✓	Google API with Playstore System Images	https://dl.google.com/android/repository/s...
✓	Google Inc.	https://dl.google.com/android/repository/a...
✓	Intel HAXM	https://dl.google.com/android/repository/e...

☐ Force https://... sources to be fetched using http://... ☐ Disable SDK diff patching

1.7 Android Development Tools

Integrated Development Environments (IDEs)

- **Android Studio**: The official IDE from Google, built on JetBrains' IntelliJ IDEA. It's the most comprehensive tool for Android development, offering a complete set of features. It includes:
    - **Intelligent code editor**: With advanced code completion, refactoring, and analysis for Kotlin, Java, and C/C++.
    - **Layout Editor**: A visual designer for building app user interfaces (UI) using XML or Jetpack Compose.
    - **Android Emulator**: A virtual mobile device to run and test apps on your computer without needing a physical device.
    - **Performance Profilers**: Tools for analyzing and optimizing your app's memory, CPU, network usage, and battery consumption.
    - **App signing and deployment**: Tools for building and signing your application package (.apk or .aab) for distribution on the Google Play Store.

6.

- **Visual Studio Code (VS Code)**: A popular, lightweight code editor that can be used for Android development, especially with frameworks like Flutter and React Native. Recent updates, including official Kotlin support, make it a viable alternative for native Android development as well.

7.

Languages and UI Toolkits

- **Kotlin**: The modern, preferred programming language for Android development. It is fully interoperable with Java and has a more concise syntax, improving developer productivity and code safety.
- **Java**: The original language for Android development, still widely used and supported.
- **Jetpack Compose**: A modern, declarative UI toolkit from Google for building native Android UIs. It uses Kotlin and significantly reduces the boilerplate code compared to the older XML-based layouts.

8. Core SDK components
9.
10.

11.
12.

- **Android SDK (Software Development Kit)**: A collection of tools and libraries that enable you to write apps for Android. It is typically managed and installed directly within Android Studio.
- **Android SDK Platform-Tools**: A sub-component of the SDK that contains essential command-line tools.
    - **Android Debug Bridge (adb)**: A versatile command-line tool that lets you communicate with an Android device or emulator to install apps, debug, copy files, and more.
    - **Fastboot**: A diagnostic protocol used primarily for flashing new firmware to an Android device.
- **Android Native Development Kit (NDK)**: A set of tools that allows developers to use native languages like C and C++ for certain parts of their apps, which can be useful for performance-critical components or reusing existing code libraries.

Backend services and testing

- **Firebase**: A mobile and web application development platform by Google. It offers a comprehensive suite of backend services, including real-time databases, authentication, cloud storage, crash reporting (Crashlytics), and analytics, so developers don't have to build their own backend from scratch.
- **LeakCanary**: A memory leak detection library that helps developers find and fix memory leaks in their apps, preventing performance issues and crashes.
- **Espresso**: A testing framework for writing automated UI tests. It is part of the Android Jetpack libraries and is designed for testing user interactions in your app.

Build and dependency management

- **Gradle**: The flexible and powerful build system used by Android Studio. It automates the process of compiling, packaging, and deploying your application.

13. Collaborative and productivity tools

- **GitHub**: A web-based platform for version control using Git. It is essential for managing source code, collaborating with other developers, and tracking changes.

- **Gemini Code Assist**: An AI-powered coding assistant built into Android Studio. It aids developers with code suggestions, generating documentation, and solving crashes, accelerating the development process.

1.8 Android Virtual Devices

An Android Virtual Device (AVD) is a software configuration that emulates a physical Android device on your computer. It is used with the Android Emulator to create virtual mobile, tablet, TV, or Wear OS devices for testing Android applications.

Components of an AVD

An AVD consists of several parts that define the characteristics of the virtual device:

- **Hardware profile:** This sets the physical attributes, such as screen size, resolution, and pixel density.
- **System image:** This defines the version of the Android operating system that the AVD will run.
- **Storage area:** A dedicated space on your development machine is used to store the virtual device's user data, installed applications, and settings.
- **Skin:** An emulator skin specifies the appearance of the device, with many predefined and customizable options available.
- **Hardware emulation options:** AVDs can be configured to emulate hardware features like cameras, accelerometers, gyroscopes, and GPS.

Purpose of using AVDs

AVDs are essential for Android developers and offer several key advantages:

- **Diverse testing:** Developers can test apps on a wide variety of device configurations, screen sizes, and Android versions without needing to own a corresponding physical device.
- **Cost-effectiveness:** Developers can test for many different environments with a single computer, eliminating the need to purchase multiple physical devices.
- **Consistency:** AVDs provide a controlled and consistent testing environment. This helps ensure that an app runs as expected across different Android platform versions.
- **Customization:** AVDs allow you to simulate specific hardware conditions. You can test how your app behaves on a slower network, with low battery, or when the device is rotated.
- **Accessibility:** Users or gamers can use AVDs to run apps and games on their computer that their physical device may not support due to different hardware or software configurations.

How to create and manage AVDs

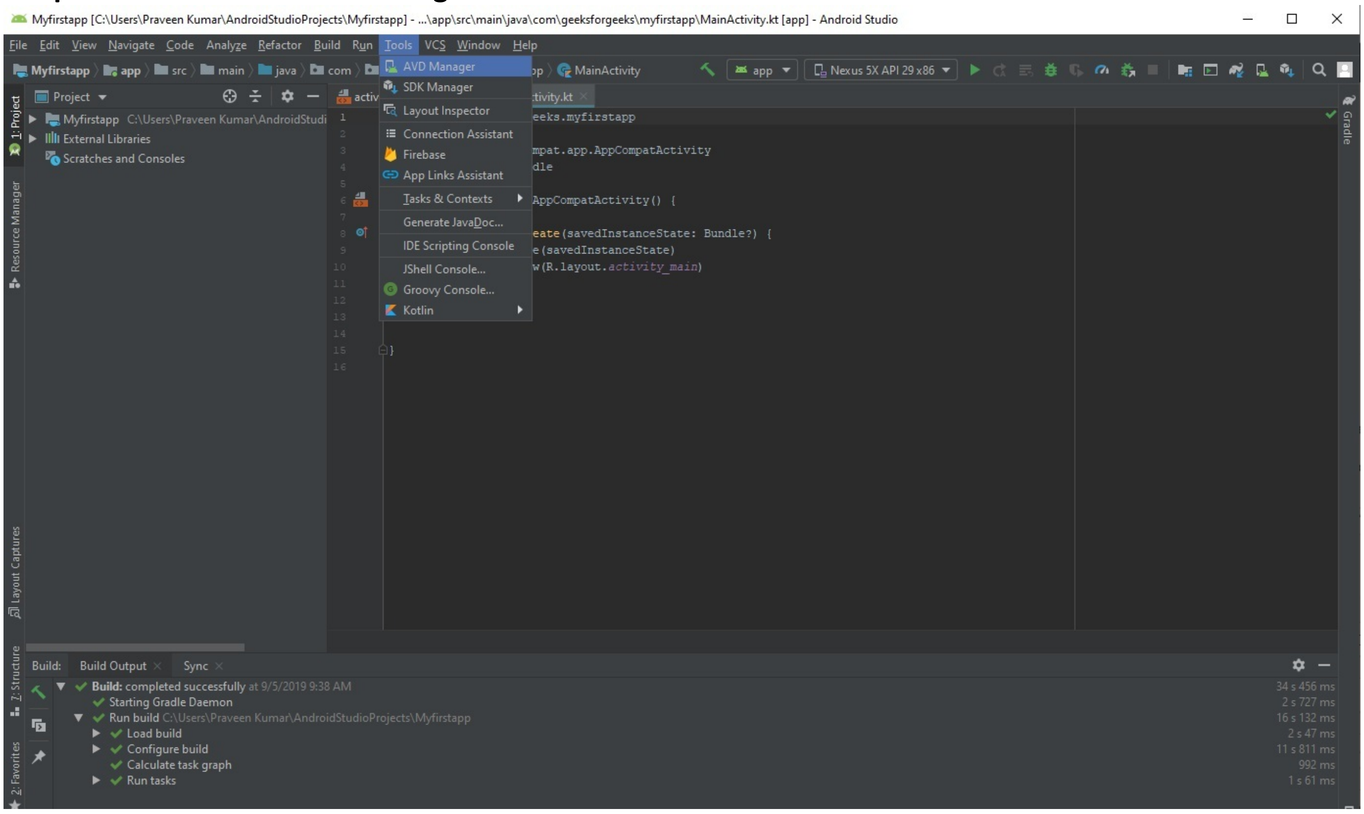
AVDs are created and managed using the Device Manager in Android Studio.

1. **Open the Device Manager** from the main menu by selecting **View > Tool Windows > Device Manager**.
2. **Click "Create Device."** This will open the **Virtual Device Configuration** dialog.

3. **Select a hardware profile** for the virtual device. For example, choose a Pixel phone or a Wear OS device.
4. **Choose a system image**, or Android version, for the device. If the system image is not already downloaded, you will need to download it first.
5. **Configure the AVD settings**, such as its name, start-up orientation, and other advanced properties, then click **Finish**.
6. **Run the AVD** by clicking the **Launch** button next to it in the Device Manager window.

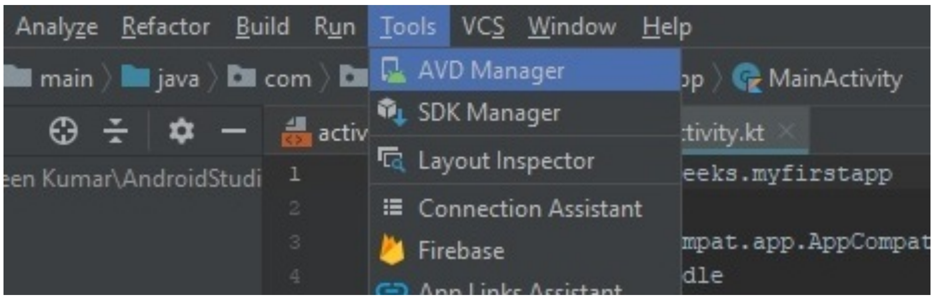
steps to install Android Virtual Device.

Step 1: Go to **Tools** > **AVD Manager**.


Myfirstapp [C:\Users\Praveen Kumar\AndroidStudioProjects\Myfirstapp] - ...\app\src\main\java\com\geeksforgeeks\myfirstapp\MainActivity.kt [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Myfirstapp app src main java com MainActivity
app
Nexus 5X API 29 x86
Project
Myfirstapp C:\Users\Praveen Kumar\AndroidStudi
External Libraries
Scratches and Consoles
AVD Manager
SDK Manager
Layout Inspector
Connection Assistant
Firebase
App Links Assistant
Tasks & Contexts
Generate JavaDoc...
IDE Scripting Console
JShell Console...
Groovy Console...
Kotlin
1: Project
Resource Manager
Layout Captures
7: Structure
2: Favorites
Gradle

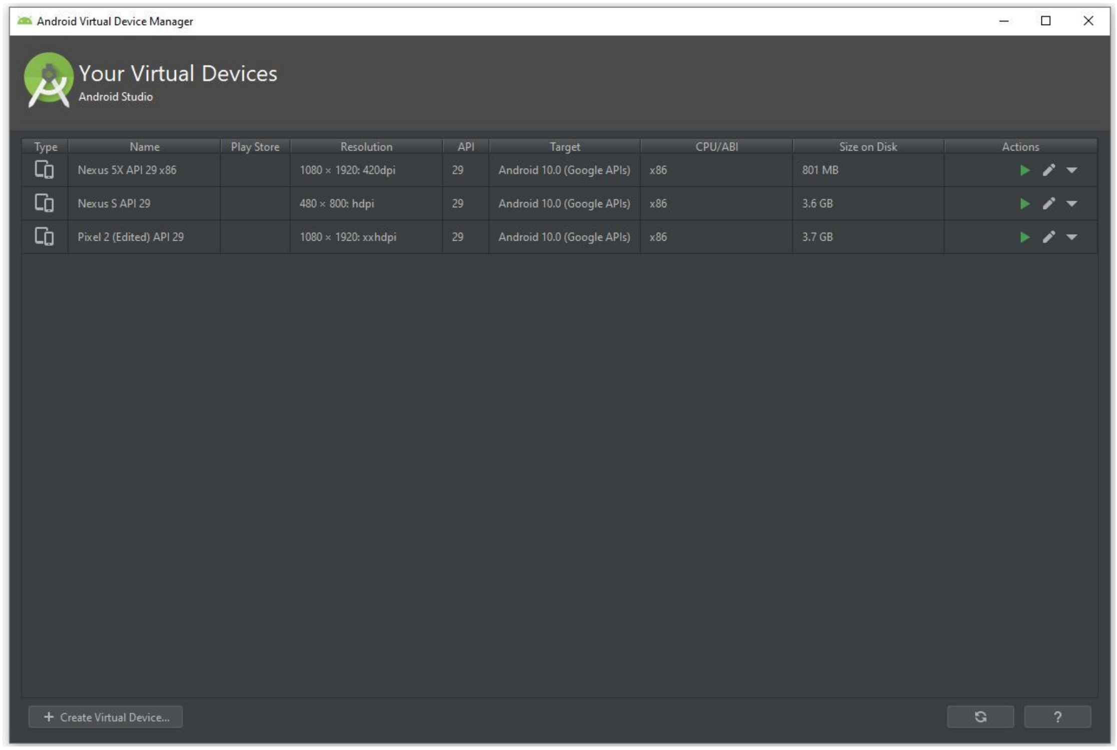
```
eeks.myfirstapp

mpat.app.AppCompatActivity
dle

AppCompatActivity() {

eate(savedInstanceState: Bundle?) {
e(savedInstanceState)
w(R.layout.activity_main)


}
```

```
Build:  Build Output  Sync
Build: completed successfully at 9/5/2019 9:38 AM                          34 s 456 ms
    Starting Gradle Daemon                                                  2 s 727 ms
  Run build C:\Users\Praveen Kumar\AndroidStudioProjects\Myfirstapp        16 s 132 ms
      Load build                                                            2 s 47 ms
      Configure build                                                      11 s 811 ms
      Calculate task graph                                                     992 ms
      Run tasks                                                              1 s 61 ms
```



```
eeks.myfirstapp

mpat.app.AppCompat
dle
```

Step 2: Now click on **Create Virtual Device**.

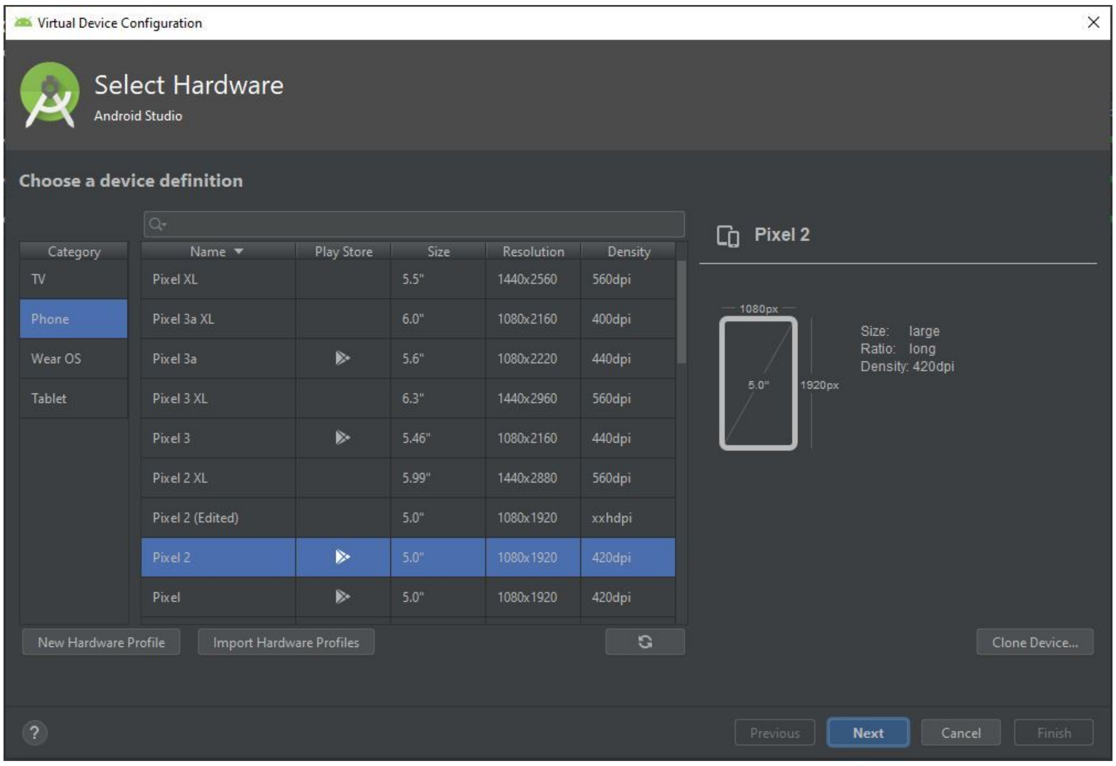


Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 29 x86		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	801 MB	
	Nexus S API 29		480 × 800: hdpi	29	Android 10.0 (Google APIs)	x86	3.6 GB	
	Pixel 2 (Edited) API 29		1080 × 1920: xxhdpi	29	Android 10.0 (Google APIs)	x86	3.7 GB	

Step 3: A pop-up window will be there and here we select the category Phone because we are creating android app for mobile and select the model of mobile phone we want to install.



Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

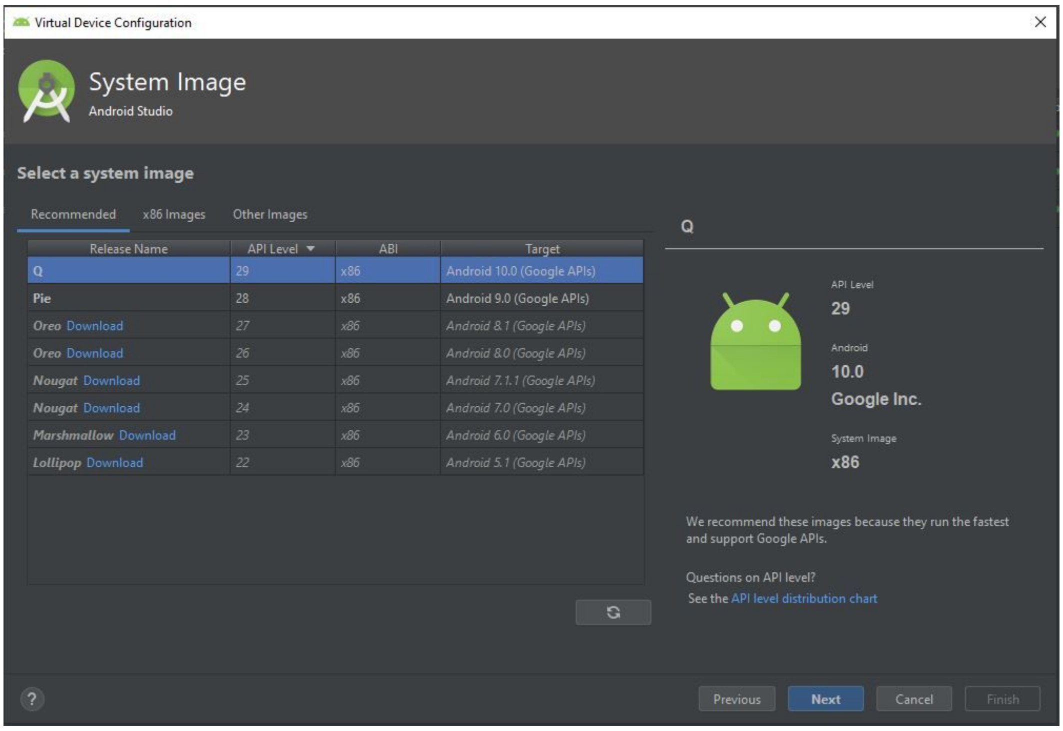
Category
TV
Phone
Wear OS
Tablet

Name	Play Store	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 3a XL		6.0"	1080x2160	400dpi
Pixel 3a	▶	5.6"	1080x2220	440dpi
Pixel 3 XL		6.3"	1440x2960	560dpi
Pixel 3	▶	5.46"	1080x2160	440dpi
Pixel 2 XL		5.99"	1440x2880	560dpi
Pixel 2 (Edited)		5.0"	1080x1920	xxhdpi
Pixel 2	▶	5.0"	1080x1920	420dpi
Pixel	▶	5.0"	1080x1920	420dpi

Pixel 2

Size: large
Ratio: long
Density: 420dpi

Step 4: Here we select the android version to download like **Q**, **Pie**, **Oreo** etc and click **Next** button.


Virtual Device Configuration

System Image
Android Studio

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
Q	29	x86	Android 10.0 (Google APIs)
Pie	28	x86	Android 9.0 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo Download	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

Q

API Level
29

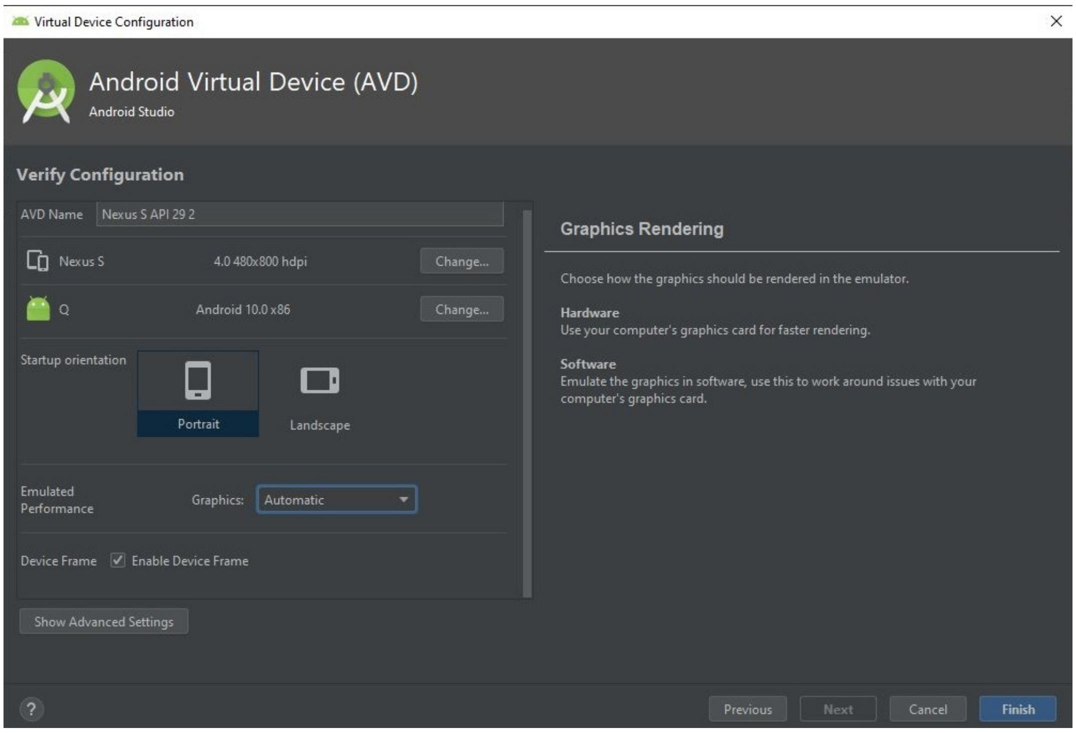
Android
10.0
Google Inc.

System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Step 5: Click the **finish** button to complete the installation.


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Nexus S API 29 2

Nexus S 4.0 480x800 hdpi Change...

Q Android 10.0 x86 Change...

Startup orientation: Portrait | Landscape

Emulated Performance Graphics: Automatic

Device Frame ✓ Enable Device Frame

Show Advanced Settings

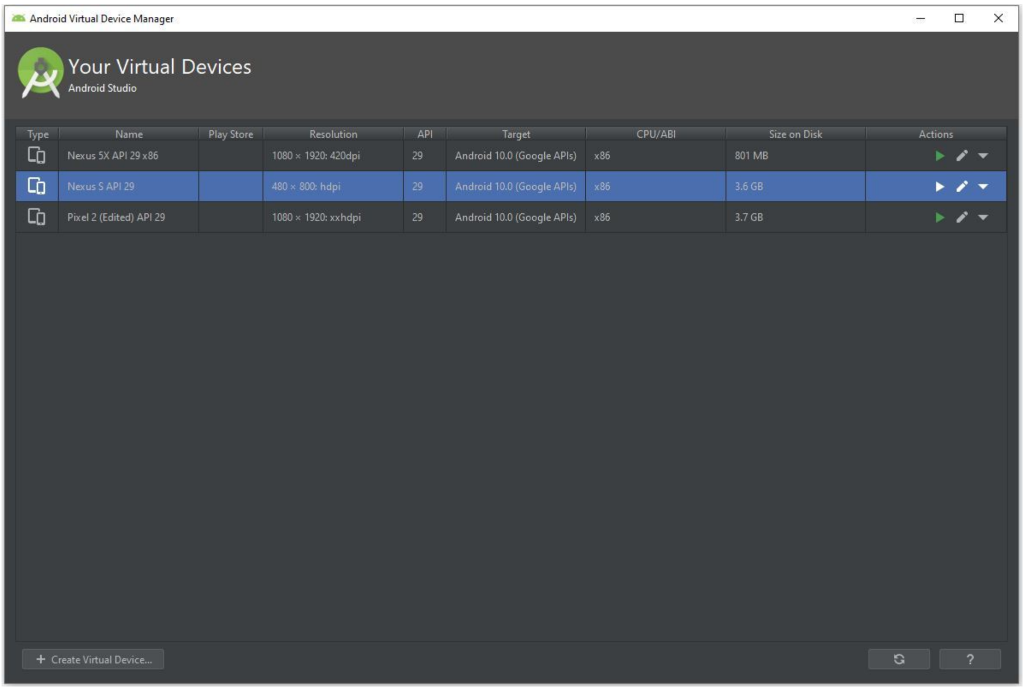
Graphics Rendering

Choose how the graphics should be rendered in the emulator.

Hardware
Use your computer's graphics card for faster rendering.

Software
Emulate the graphics in software, use this to work around issues with your computer's graphics card.

Step 6: Now we can select the virtual device we want to run as emulator can click on the **run** icon.


Android Virtual Device Manager

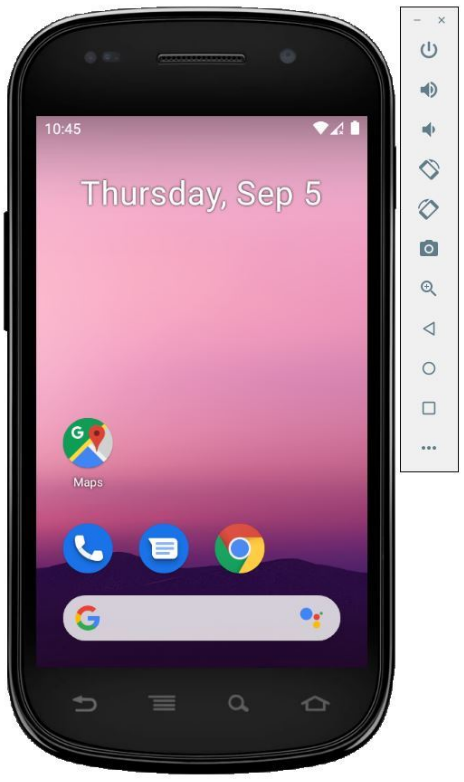
Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 29 x86		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	801 MB	
	Nexus S API 29		480 × 800: hdpi	29	Android 10.0 (Google APIs)	x86	3.6 GB	
	Pixel 2 (Edited) API 29		1080 × 1920: xxhdpi	29	Android 10.0 (Google APIs)	x86	3.7 GB	

+ Create Virtual Device...

?

Step 7: Finally our virtual device is ready to run our android app.



1.9 Development Environment

An Android development environment consists of the official Integrated Development Environment (IDE) **Android Studio**, the **Android SDK**, and other essential build and testing tools. This setup provides everything a developer needs to write, compile, debug, and package Android applications.

Core components

Android Studio
Android Studio is the official IDE for Android development, built on JetBrains' IntelliJ IDEA software. It is the central hub for your development workflow and comes with a suite of tools to increase your productivity, including:

- **Intelligent code editor:** Provides advanced features for coding in Kotlin and Java, including code completion, analysis, and refactoring.
- **Layout editor:** A visual design tool that allows you to build responsive user interfaces by dragging and dropping UI elements.
- **Built-in emulator:** For testing your applications on a variety of virtual devices with different screen sizes, hardware configurations, and Android versions without needing a physical device.
- **Performance profilers:** Help you track memory and CPU usage, detect memory leaks, and analyze network requests to optimize your app's performance.
- **GitHub integration:** Allows for seamless collaboration and version control.

Android SDK
The Android SDK (Software Development Kit) is a set of development tools essential for creating Android applications. When you install Android Studio, it automatically installs the SDK. The SDK Manager in Android Studio lets you manage and update different SDK components, including:

- **SDK Platforms:** Versions of the Android OS (e.g., API Level 36 for Android 15), necessary for compiling your app for a specific Android version.
- **SDK Tools:** Includes important command-line tools like ADB (Android Debug Bridge) for communicating with devices and emulators.
- **Build Tools:** Handles the compilation and packaging of your app.

Gradle
Gradle is the automated build system used by Android Studio. It is essential for managing dependencies, packaging your code and resources into an APK (Android Package), and automating repetitive build tasks. Gradle uses build scripts that can be written in Kotlin or Groovy.

Programming languages

Google's recommended and official programming language for Android development is **Kotlin**.

- **Kotlin:** A modern, expressive, and concise language that significantly reduces boilerplate code compared to Java. It is highly interoperable with Java, allowing for easy integration into existing projects. Google actively promotes a "Kotlin-first" approach for new Android development.

- **Java:** The original language for Android development, Java is still fully supported and widely used, especially in older and legacy codebases.

Other critical tools and libraries

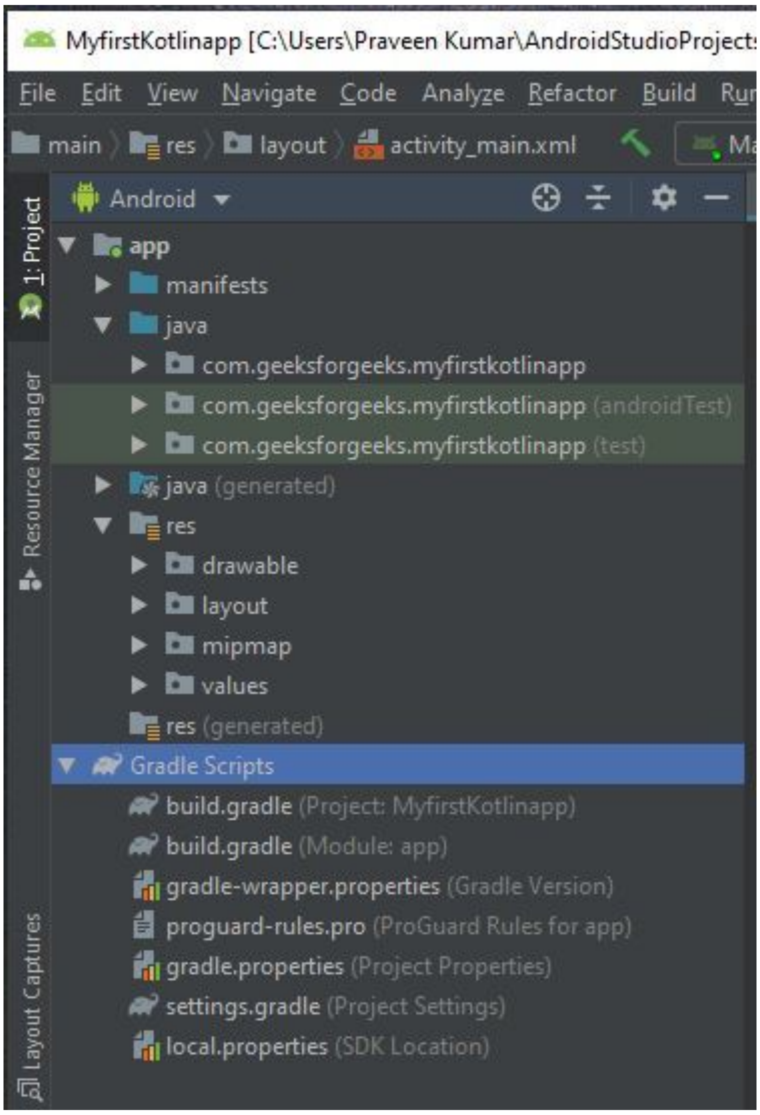
- **Android Jetpack:** A collection of libraries from Google designed to help developers follow best practices, reduce boilerplate code, and write code that works consistently across Android versions.
- **Git:** A distributed version control system for tracking changes in your code during development.
- **Firebase:** A platform from Google that provides backend services for mobile and web applications, offering tools for analytics, authentication, databases (Firestore), cloud storage, and crash reporting.

How to set up your environment

1. **Install Android Studio:** Download the official IDE from the Android Developer website and follow the installation wizard. This will also install the necessary Android SDK components.
2. **Verify components:** After installation, open the SDK Manager from the "More Actions" menu on the Android Studio welcome screen. Confirm that you have the latest Android SDK Platform and SDK Tools installed.
3. **Create a virtual device (AVD):** Use the Device Manager to create an AVD that simulates an Android device. This allows you to run and test your app directly from Android Studio.
4. **Create your first project:** Select "New Project" from the Android Studio welcome screen and choose a project template. Android Studio will automatically configure the necessary Gradle files and dependencies.

1.10 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:


MyfirstKotlinapp [C:\Users\Praveen Kumar\AndroidStudioProject
File Edit View Navigate Code Analyze Refactor Build Ru
main 〉 res 〉 layout 〉 activity_main.xml
Android
1: Project
Resource Manager
Layout Captures

```
▼ app
    ▶ manifests
    ▼ java
        ▶ com.geeksforgeeks.myfirstkotlinapp
        ▶ com.geeksforgeeks.myfirstkotlinapp (androidTest)
        ▶ com.geeksforgeeks.myfirstkotlinapp (test)
    ▶ java (generated)
    ▼ res
        ▶ drawable
        ▶ layout
        ▶ mipmap
        ▶ values
      res (generated)
▼ Gradle Scripts
      build.gradle (Project: MyfirstKotlinapp)
      build.gradle (Module: app)
      gradle-wrapper.properties (Gradle Version)
      proguard-rules.pro (ProGuard Rules for app)
      gradle.properties (Project Properties)
      settings.gradle (Project Settings)
      local.properties (SDK Location)
```

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.

1. Manifests Folder
2. Java Folder
3. res(Resources)Folder

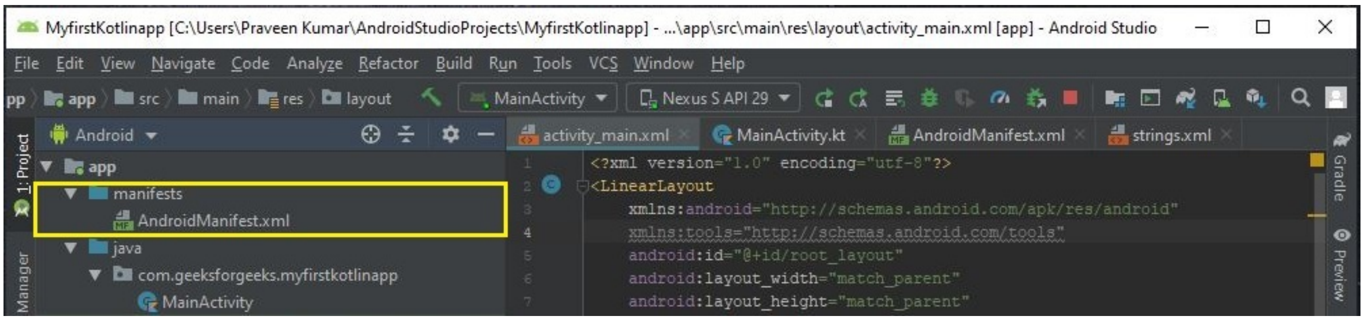
    - Drawable Folder
    - Layout Folder
    - Mipmap Folder
    - Values Folder

4. Gradle Scripts

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 intermediator between android OS and our application. Following is the manifests folder structure in the android application.

1. Manifests Folder

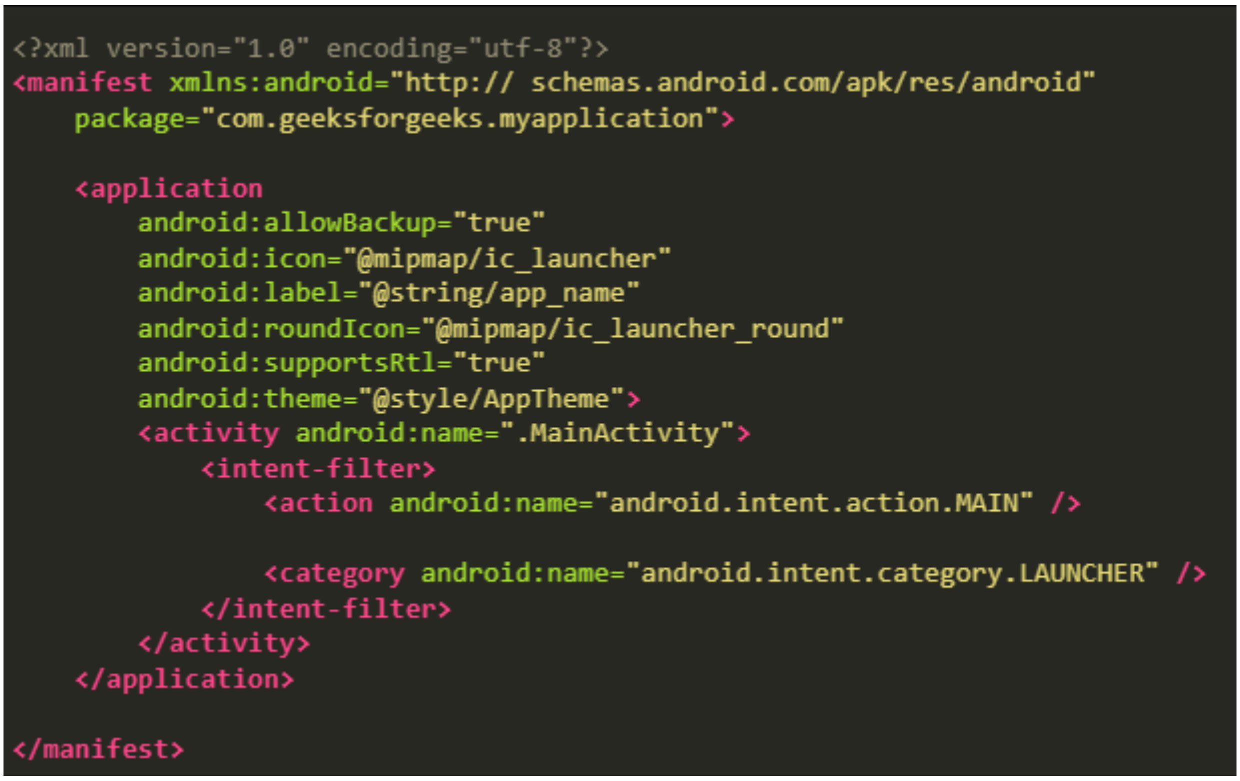
Manifests folder contains **AndroidManifest.xml** for creating our android application. This file contains information about our application such as the Android version, metadata, states package for Kotlin file, and other application components. It acts as an intermediator between android OS and our application. Following is the manifests folder structure in the android application.


MyfirstKotlinapp [C:\Users\Praveen Kumar\AndroidStudioProjects\MyfirstKotlinapp] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
pp app src main res layout
MainActivity
Nexus S API 29
Android
activity_main.xml
MainActivity.kt
AndroidManifest.xml
strings.xml
1: Project
Manager
Gradle
Preview

```
app
  manifests
    AndroidManifest.xml
  java
    com.geeksforgeeks.myfirstkotlinapp
      MainActivity
```

```
<?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:id="@+id/root_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

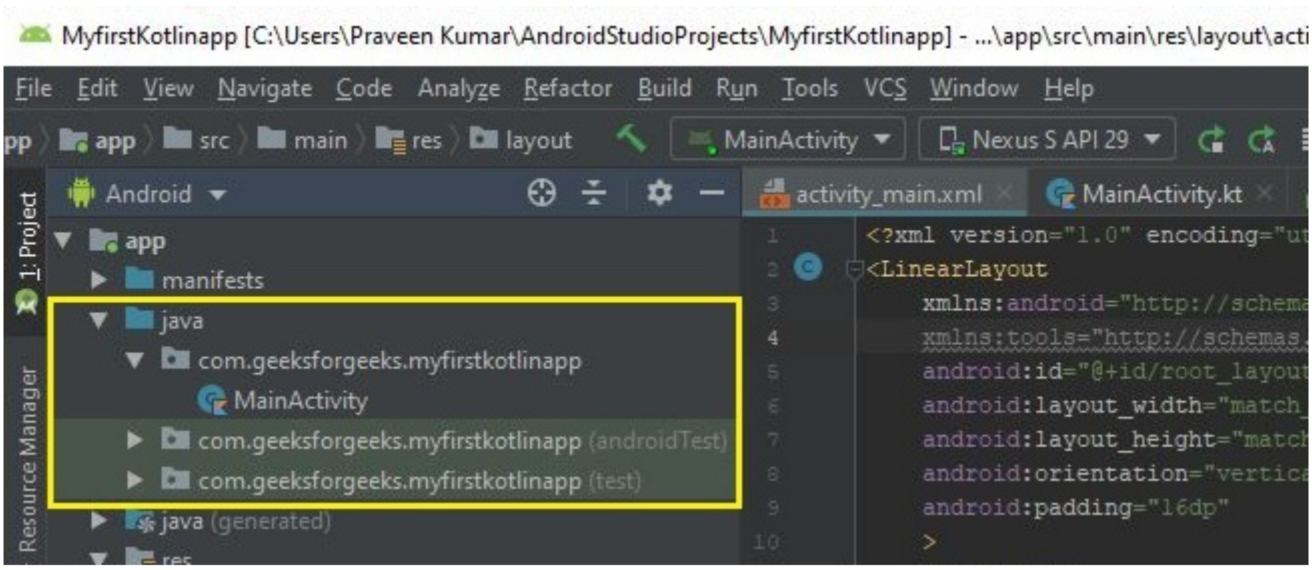
AndroidManifest.xml



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

2. 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.

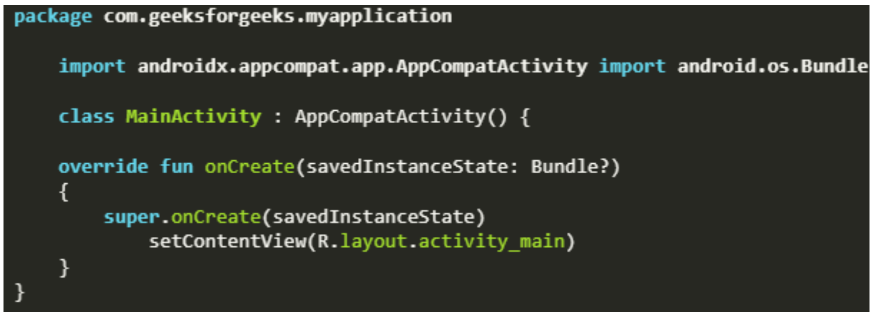

MyfirstKotlinapp [C:\Users\Praveen Kumar\AndroidStudioProjects\MyfirstKotlinapp] - ...\app\src\main\res\layout\acti
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
pp app src main res layout
MainActivity
Nexus S API 29
1: Project
Resource Manager
Android
activity_main.xml
MainActivity.kt

```
app
  manifests
  java
    com.geeksforgeeks.myfirstkotlinapp
      MainActivity
    com.geeksforgeeks.myfirstkotlinapp (androidTest)
    com.geeksforgeeks.myfirstkotlinapp (test)
  java (generated)
  res
```

```
<?xml version="1.0" encoding="ut
<LinearLayout
    xmlns:android="http://schema
    xmlns:tools="http://schemas.
    android:id="@+id/root_layout
    android:layout_width="match_
    android:layout_height="match
    android:orientation="vertica
    android:padding="16dp"
    >
```

MainActivity.kt and MainActivity.java

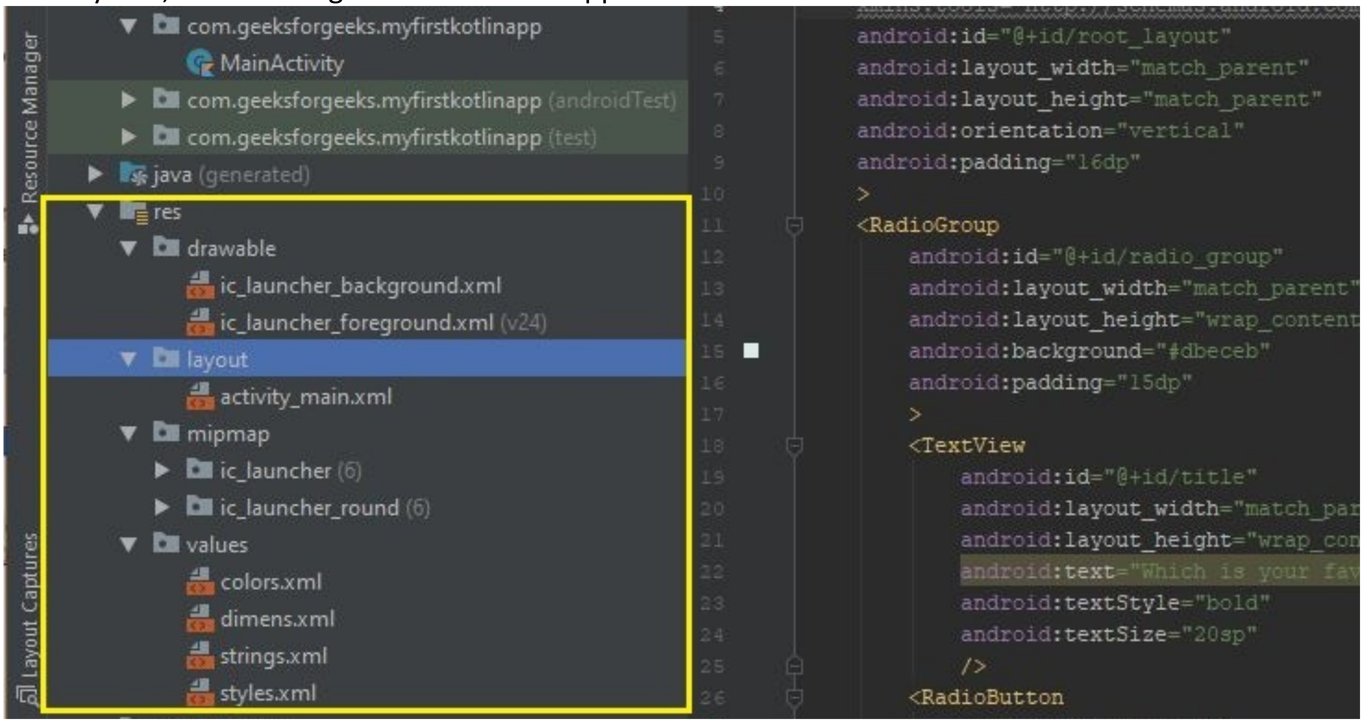
```java
package com.geeksforgeeks.myapplication;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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



```
package com.geeksforgeeks.myapplication

    import androidx.appcompat.app.AppCompatActivity import android.os.Bundle

    class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)
    }
}
```

3. 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.



```
▼ com.geeksforgeeks.myfirstkotlinapp
      MainActivity
▶ com.geeksforgeeks.myfirstkotlinapp (androidTest)
▶ com.geeksforgeeks.myfirstkotlinapp (test)
▶ java (generated)
▼ res
  ▼ drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
  ▼ layout
      activity_main.xml
  ▼ mipmap
    ▶ ic_launcher (6)
    ▶ ic_launcher_round (6)
  ▼ values
      colors.xml
      dimens.xml
      strings.xml
      styles.xml
```

```
android:id="@+id/root_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp"
>
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content
    android:background="#dbeceb"
    android:padding="15dp"
    >
    <TextView
        android:id="@+id/title"
        android:layout_width="match_par
        android:layout_height="wrap_con
        android:text="Which is your fav
        android:textStyle="bold"
        android:textSize="20sp"
        />
    <RadioButton
```

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. It contains the **activity_main.xml** file.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

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.

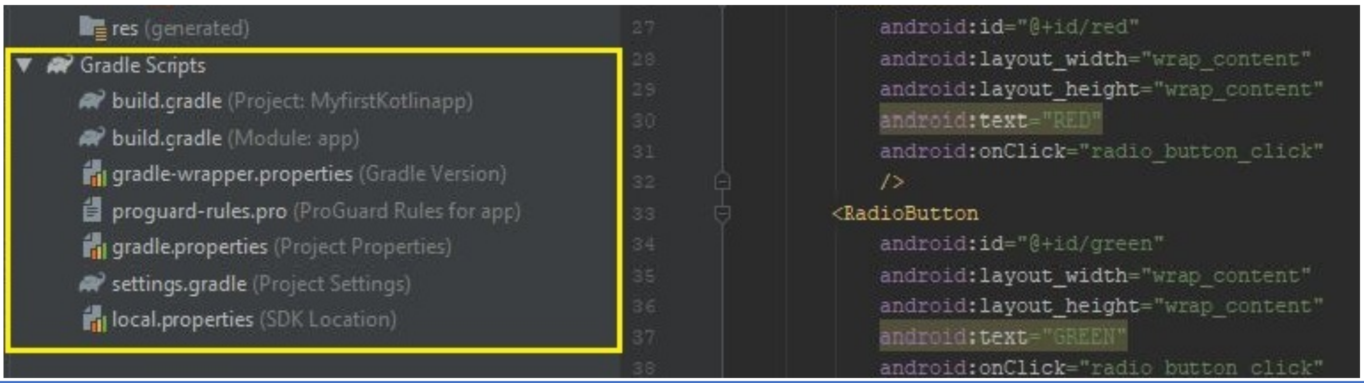
res/values folder

Values folder contains a number of XML files like strings, dimensions, colors, and style definitions. One of the most important files is the **strings.xml** file which contains the resources.

```xml
<resources>
    <string name="app_name">NameOfTheApplication</string>
    <string name="checked">Checked</string>
    <string name="unchecked">Unchecked</string>
</resources>
```

4. 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.



```
res (generated)
Gradle Scripts
    build.gradle (Project: MyfirstKotlinapp)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

```xml
    android:id="@+id/red"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="RED"
    android:onClick="radio_button_click"
    />
<RadioButton
    android:id="@+id/green"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="GREEN"
    android:onClick="radio_button_click"
```

1.11 Android Manifest File

The Android manifest file (AndroidManifest.xml) is an essential XML file located at the root of every Android app project. It acts as a blueprint, providing critical information about the app to the Android operating system, build tools, and the Google Play Store.

This file describes the app's structure, components, permissions, and compatibility requirements so the system can properly run and manage it. Without a manifest, the app cannot be recognized or installed.

Key functions of the manifest file

- **Declares app components:** Every component of your app—including activities, services, broadcast receivers, and content providers—must be declared in the manifest. For a component to be discoverable and started by the system, it must be registered with a corresponding XML tag.

- **Requests permissions:** Any sensitive data or device features your app needs to access (e.g., the internet, camera, or user contacts) must be declared with <uses-permission> tags. The system uses this information to prompt the user for permission, if required.

- **Specifies hardware and software features:** The manifest can define the specific hardware or software features your app requires, such as a camera or NFC. The Google Play Store uses this information to ensure your app is only available on compatible devices.

- **Sets the package name:** The file defines the unique package name for your app, which serves as its unique identifier on the device and in the app store.

- **Handles intent filters:** It declares which "intents" (messages) a component can respond to. An <intent-filter> for an activity can, for example, designate it as the main entry point (the launcher activity) for the app.

- **Determines device compatibility:** You can declare your app's compatibility with different screen sizes and Android API levels. For example, <uses-sdk> with a minSdkVersion attribute ensures your app will only run on devices with a compatible Android version.

- **Manages app metadata:** Defines various metadata, such as the app's icon, label (name), and theme.

Example of Manifest file:

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

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

        <service android:name=".MyService" />

    </application>
</manifest>
```

Parul® University
Vadodara, Gujarat

NAAC GRADE A++

https://paruluniversity.ac.in/