



ANDROID PROGRAMMING

IT-606 [Lab]



0103IT201111
NAVEEN SINGH

Information Technology

IT 606 [ANDROID PROGRAMMING]

1. Introduction to Android:

A little Background about mobile technologies, Overview of Android, An Open Platform for Mobile development, Open Handset Alliance, What does Android run On – Android Internals, Why to use Android for mobile development

2. Developing for Android:

My First Android Application, How to setup Android Development Environment, Android development Framework - Android-SDK, Eclipse, Emulators – What is an Emulator / Android AVD, Creating & setting up custom Android emulator, Android Project Framework, My First Android Application.

3. Android Activities and UI Design

Understanding Intent, Activity, Activity Lifecycle and Manifest, Creating Application and new Activities, Expressions and Flow control, Android Manifest, Simple UI -Layouts and Layout properties, Fundamental Android UI Design, Introducing Layouts Creating new Layouts, Drawable Resources, Resolution and density independence (px,dip,dp,sip,sp), XML Introduction to GUI objects viz., Push Button Text / Labels, EditText, ToggleButton, WeightSum, Padding, Layout Weight

1. Introduction to Android:

A little Background about [mobile technologies](#), Overview of [Android](#), An Open Platform for Mobile development, Open Handset Alliance, What does Android run On – [Android Internals](#), Why to use Android for mobile development

A little Background about mobile technologies

- **Mobile technology** is a type of technology in which a user utilizes a mobile phone to perform communications-related tasks, such as communicating with friends, relatives, and others.
- Mobile technology refers to any technology that's "portable."
- It consists of portable two-way communications devices, computing devices and the networking technology that connects them.
- It is used to send data from one system to another.
- Some of the most popular forms of mobile technology include: **Smartphones, Laptops, Tablets, Smartwatches, Hotspots, IoT Devices**

Types of Mobile Networks

- **Cellular Networks:** Cellular networks operate through radio networks distributed via cell towers, which allows mobile devices to automatically switch frequencies to their nearest geographical tower without interruption. Cellular networks have the capability to service mass amounts of users at a single time and are currently in their fifth generation of service.
- **4G:** Referring to the fourth generation of cellular service, 4G operates on packet switching technology and organizes data into smaller groupings for fast transmission before reassembling at the destination.
- **5G:** The fifth generation of network service has led to new infrastructure for widespread adoption and operates at higher frequencies in aggregated bands. This allows networks to access more bandwidth and increase transmission speed. 5G is reported to be up to 100 times faster at sending and receiving signals than 4G.
- **Wi-Fi:** Wi-Fi utilizes radio wave technology to connect devices to localized hotspot routers. Internet providers allow users to connect to their network but will not automatically pass signals to a device without a WiFi connection. Users will often have the ability to make their WiFi network available for public or private use.
- **Bluetooth:** Rather than connect devices to the internet, Bluetooth networks connect devices to other devices via short-wavelength radio waves.

Use of Mobile technology

- To connect people from anywhere.
- complete collaborative work.
- one may check the current traffic situation on their phone and make appropriate decisions to arrive on time. The weather is also a factor.
- Mobile phones are being used for a variety of legitimate tasks, including meeting schedules, sending and receiving documents, providing introductions, warnings, and job applications, among others.
- These days, mobile phones are also used as a wallet to make payments.
- Mobile phones eliminate the need for costly technology like landline carrier services.

BENEFITS OF MOBILE TECHNOLOGY

- Scalability
- Onboarding
- Cloud-based development
- Reusability
- Edge computing

OPTIONAL/ NOTE:-

Types of Mobile Technologies

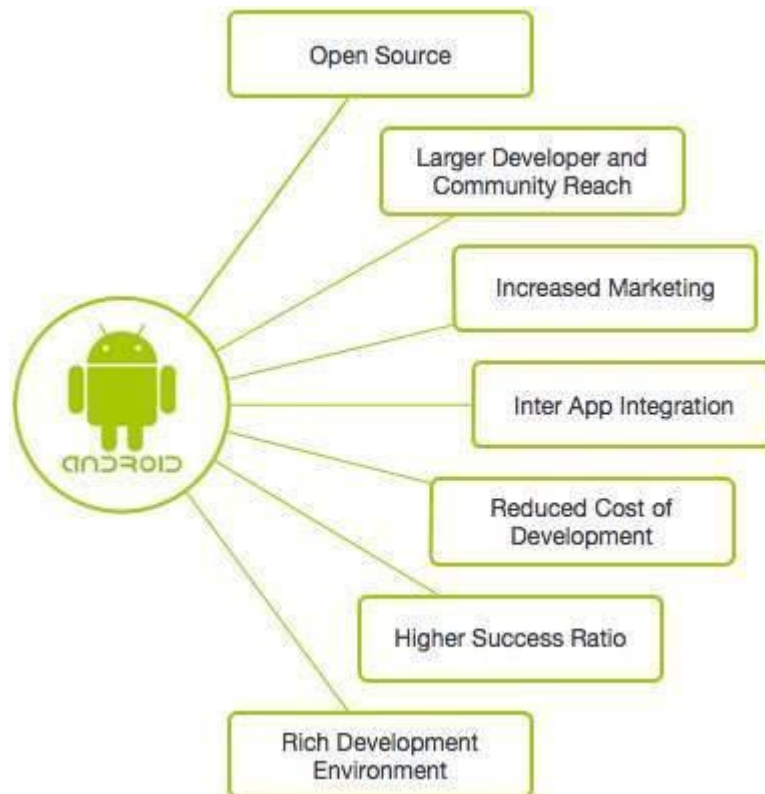
Followings are the few famous mobile technologies:

- SMS
 - MMS
 - 5G
 - 4G
 - 3G
 - GSM
 - CDMA
 - Wi-Fi
1. **SMS:** "SMS" stands for "Short Message Service." It is now the most widely used and oldest text messaging service. SMS are also sent over cellular networks, therefore you'll need a wireless plan and a wireless carrier. SMS is fast gaining popularity in the world as a low-cost messaging medium. Every text message delivered to a cell phone has become known as SMS. Messages can usually be up to 140 characters long. SMS was originally developed for GSM phones, although it is now supported by all major cellular phone networks.
 2. **MMS:** MMS (Multimedia Messaging Service) messaging is a standard method of delivering multimedia material, including messages. MMS, as opposed to SMS, can send up to forty seconds of video, one picture, a multi-image slideshow, or audio. MMS texting will be supported by the majority of contemporary devices. MMS capability is typically embedded within the text message interface and is turned on automatically when needed. If you enter in a text-only message, for example, it will be transmitted by SMS. If you include a graphic or video, the multimedia part will be sent via MMS. Similarly, if someone sends you a multimedia message, your phone will automatically receive the file via MMS.
 3. **3G:** The third letter in the designation 3G stands for third-generation access technology, which allows mobile phones to connect to the internet. Every new technology introduces new frequency bands and data transmission rates.
The first generation emerged in the 1980s. First-generation uses large phones that had to be mounted on top of cars because they were too heavy to hold. Text messaging was made possible by the second-generation network, which became available in the 1990s. This huge and game-changing advancement also provided a more secure network and laid the path for today's ubiquitous 3G and 4G technology.
The development of 3G connection-based networks in 2001 marked the start of mainstream Internet use on mobile phones. The signals are transmitted by a network of telephone towers, ensuring robust and relatively rapid long-distance communication. The user's mobile phone is receiving data from the tower nearest to it. Upload speeds of up to 3 Mbps are possible on 3G networks. The fastest 2G phones, on the other hand, may get up to 144Kbps.
 4. **4G:** The fourth generation of mobile networking technology is known as 4G, which comes after the 2G and 3G networks. Although it's commonly referred to as 4G LTE, this isn't exactly right because LTE is just one sort of 4G. 4G was launched in the United Kingdom in 2012.
Premium 4G offers download speeds of around 14 Mbps, which is over five times quicker than the 3G network's predecessor. 4G networks can currently attain speeds of up to 150 Mbps, allowing users to download gigabytes of data in minutes, if not seconds, rather than hours as with 3G networks. Uploading data is also significantly faster with 4G – normal upload speeds are over 8 Mbps, with theoretical rates of up to 50 Mbps, whereas 3G upload speeds are under 0.5 Mbps.
 5. **5G** is the fifth generation of cellular wireless technology. Like 4G, it uses frequencies that are part of the radio spectrum, but 5G uses very high frequencies that offer more bandwidth. Means Greater bandwidth with ultra-low latency.
This means more data delivered at higher speeds to more devices. Imagine video streaming to a smartphone. According to IBM, 5G will "improve that experience by making it 10x better not just for one individual, but for anyone streaming a video at the same time."
 6. **Global System for Mobile technology:** The (GSM) is an acronym for Global System for Mobile Communication. GSM is a cellular technology that is open and digital and is used for mobile communication. It operates on the 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz frequency ranges. It employs a hybrid of FDMA and TDMA.
 7. **Code Division Multiple Access:** (CDMA) is an acronym for code division multiple access. It is a channel access mechanism that also serves as an example of multiple access. Multiple access simply means that data from multiple transmitters can be delivered onto a single communication channel at the same time.
 8. **Wi-Fi (Wireless Fidelity):** Wi-Fi is a wireless networking technology that allows us to connect to a network or to other computers or mobile devices across a wireless channel. Data is delivered in a circular region over radio frequencies in Wi-Fi. Wi-Fi (Wireless Fidelity) is a generic acronym for a communication standard for a wireless network that functions as a Local Area Network without the use of cables or other types of cabling.

Overview of Android

- Android is an **open source and Linux-based Operating System** for mobile devices.
- It is designed primarily for touchscreen mobile devices such as mobile phones and tablets.
- Android was developed by the **Open Handset Alliance**, a group of companies led by **Google**, and other companies.
- The **first beta version** of the Android Software Development Kit (SDK) was released by **Google in 2007** where as the first commercial version, **Android 1.0, was released in September 2008**.
- The source code for Android is available under **free and open source** software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.
- Android has a large user base and is used on a variety of devices, including phones, tablets, televisions, and even automobiles.
- Developers create apps for a variety of reasons. They may need to address business requirements or build new services or businesses, or they may want to offer games and other types of content for users.
- The SDK includes software libraries of prewritten code, a debugger, a device emulator, documentation, sample code, and tutorials.

Why Android ?



Features of Android

Sr.No.	Feature & Description
1	Beautiful UI Android OS basic screen provides a beautiful and intuitive user interface.
2	Connectivity GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

3	Storage SQLite, a lightweight relational database, is used for data storage purposes.
4	Media support H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
5	Messaging SMS and MMS
6	Web browser Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
7	Multi-touch Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
8	Multi-tasking User can jump from one task to another and same time various application can run simultaneously.
9	Resizable widgets Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	Multi-Language Supports single direction and bi-directional text.
11	GCM Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
12	Wi-Fi Direct A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
13	Android Beam A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Applications

- Android applications are usually developed in the Java language using the Android Software Development Kit.

- Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play, SlideME, Opera Mobile Store, Mobango, F-droid** and the **Amazon Appstore**.

History of Android

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.



1. Android 1.0: Alpha (API 1)

The initial version of Android was released on September 23, 2008, known as Android 1.0. There were many loopholes in this alpha version including the necessity of physical keyboards or hardware buttons. It has an android API (application program interface) level 1. API level is the integer value that identifies the API framework revision provided by the version of the android platform. API framework includes the core set of classes, packages, XML elements, intents, and permissions that the application can request.

Features

- Google Maps, browser, calendar
- Camera and scroll down the notification bar
- Gmail integration, Contacts, and Google Synchronization.
- Wireless supports – Wi-Fi and Bluetooth.

2. Android 1.1: Beta (API 2)

Android Beta version was released on February 9, 2009, with API changes and resolved issues encountered in 1.0. This android version was released for T-Mobile G1 devices only.

Features

- Display details and reviews for locations
- Add Save attachment in the message
- Provide detailed information by clicking on the business

3. Android 1.5: Cupcake (API 3)

It was released on April 30, 2009, with the first official public code name and amazing features as compared to the old versions. Moreover, it also brought the third-party app widgets that was the most distinguishing and valuable feature.

Features

- on-screen keyboard and search function
- Uploading videos and images
- Copy and paste facility and video recordings
- Support for MPEG4 and 3GP formats

4. Android 1.6: Donut (API 4)

The android 1.6 version was released on September 15, 2009, was many valuable changes including the ability to operate on a number of different screen resolutions and sizes. Donut provides the voice and text entry search including bookmark history, and the ability to select multiple images for deletion.

Features

- Power Control widget for handling Wi-Fi, Bluetooth, GPS, etc.
- Gallery and Camera quick toggling features
- WVGA screen resolution speed
- Technology support for CDMA/EVDO, 802.1x, VPNs, and a text-to-speech engine
- Speed improvements for camera and searching applications
- Quick Search Box

5. Android 2.0: Eclair (API 5)

Éclair 2.0 – 2.1 was released on December 3, 2009, with multiple account synchronizations of email and contacts. Moreover, there are many new features were added including flash support, scene mode, white balance, color effect, macro focus and digital zoom. Android 2.1 version brought bug fixes and stability improvements to the Éclair. The most valuable feature was real-time traffic information and voice-guided turn by turn navigation.

Features

- Update UI
- Support Bluetooth 2.1
- Improve Google map
- Minor API Changes
- Support Live and animated Wallpapers
- Ability to add contact's photo and select to call, message or email

6. Android 2.2: Froyo (API 8)

Android version 2.2 was released on May 20, 2010, with speed, memory, and performance improvements. It was introduced with the enhanced Bluetooth functionality and compatibility with docks, portable WiFi hotspot for sharing the 3G connection. Google Nexus One was the first smartphone having an android 2.2 android version. Android version list provides you the complete historical knowledge of android operating system updates and improvements.

Features

- Support for Animated GIF and multiple keyboard languages
- Speed and performance improvements
- Upload file support in the browser
- Support numeric & alphanumeric passwords to enhance security
- Increased Compatibility with car kits and headsets
- Wi-Fi Support Hotspot functionality

7. Android 2.3: Gingerbread (API 9)

Gingerbread exists in the 7th position in the android version list that was released on December 6, 2010. The main enhanced feature was the introduction of gaming API with improved graphical intense gaming, so it has boomed the mobile games.

Features

- Improve Copy and Paste Facility
- Updated UI design
- Support for VP8 and WebM video format
- Video calling and Social Networking Supports

- Easy to use a keyboard with faster and intuitive typing

8. Android 3.0: Honeycomb (API 11)

Honeycomb from the android version list was released in 2011 that was tablet-only release to launch the Motorola Xoom. It was also suitable for those mobiles having a larger view than current smartphones.

Features

- Gmail, contacts, camera and gallery improvements
- Support for passwords with complex characters
- encrypted storage and updated 3D UI
- Supports multiprocessors and recent apps for easy visual multitasking
- Media Sync from SD Card
- Action bar for application control
- System bar for global status and notifications
- Google eBooks and Talk Video Chat
- Support Adobe Flash in Browser
- More sensor support
- High-performance Wi-Fi Connections and Lock
- Chinese handwriting and redesigned keyboard

9. Android 4.0: Ice Cream Sandwich (API 14)

The ice cream sandwich was released on October 19, 2011, with many enhanced features to enter the era of modern design. The snapshot was introduced to take screenshots by holding the volume and power button. Android version list has a variety of android operating systems but as compared to all older versions, ice cream sandwich widgets are more robust and resizable.

Features

- Spelling check feature
- Wi-Fi direct
- Photo Decor facility and on-screen buttons
- Unlocking with face-fixing.
- Card-like appearance for app-switching
- Improved video recording with high resolution
- Better Camera performance
- Ability to open up to 16 tabs in the web browser

10. Android 4.1: Jelly Bean (API 16)

The better version of android known as Jelly Bean was released in June 2012 with Google Digital Assistant technology accessible from the home screen. The spectacular predictive intelligence utility provides expandable and interactive notifications. Moreover, users can enjoy the multi-user support into play and many other valuable features.

Features

- Voice search and typing
- Panorama
- Project Butter
- Expandable notifications
- Daydream as a screensaver
- Power control
- Support USB audio
- Improved camera app
- Security improvements
- New gestures and accessibility features
- Multiple user accounts (Only for tablets)
- 4k resolution support
- Supporting Bluetooth with low energy
- Bi-directional text and different language support
- Set or adjust the volume of incoming calls and show a message alert

- Google displays relevant content based on your search history
- Native emoji support

11. Android 4.4: KitKat (API 19)

Android version 4.4 was released on September 3, 2013, with more focused on better user experience. KitKat is optimized to run at a larger range of old versions from the android version list. The smartphone must have a minimum of 512 MB of RAM.

Features

- Screen Recording
- Contact Prioritization
- GPS Support
- Smarter Caller ID
- Offline music support
- UI updates for alarm and google map navigations
- Cartoonish ideograms and emojis to the Google keyboard
- KitKat has 'OK Google' feature that allows access to Google to the users without touching your smartphones.

12. Android 5.0: Lollipop (API 21)

Lollipop or android version 5.0 was released on November 12, 2014, with a redesigned user interface and built with "material design". It gives a new and modern look extended across all of android, applications, and google products. Lollipop from the android version list comes with many amazing features including Support for better notification management.

Features

- Support ART
- Better device protection
- Notifications can be flicked away from the lock screen
- Better and improved UI
- Built-in battery saver feature
- New material design
- Revamped navigation bar
- Support for Multiple sim cards
- High definition of voice call.

13. Android 6.0: Marshmallow (API 23)

In the year 2015, Google used "Macadamia Nut Cookie" to describe android version 6.0 before the Marshmallow official announcement.

Features

- Support for Fingerprint readers
- Type C USB support
- Multi-window experience
- 'Sleep Mode' for saving battery life
- Clear permission system
- Custom google tabs and improved Copy-pasting

14. Android 7.0: Nougat (API 24)

The android 7.0 was released in 2016 with a native split-screen mode, data saver functionality, and a "bundled-by-app" system to organize notifications.

Features

- Provide multitasking and split-screen mode
- Storage manager enhancements
- Quick setting toggles
- Display touch enhancements
- Better setting application
- Inline reply to messages and notifications without opening applications

15. Android 8.0: Oreo (API 26)

Oreo was released in the year 2017 having native picture-in-picture mode, notification snoozing options and better control over how applications can alert you by notifications.

Features

- Password autofill
- Auto-enable Wi-Fi
- Downloadable fonts
- Multi-display support
- Support Picture-in-Picture
- Notification channels and snooze notification
- Google Play support and new emoji styling
- Adaptive icons and smart text selection

16. Android 9: Pie (API 28)

The pie was publicly released on August 6, 2018, with plenty of amazing features according to the users' interests and requirements. According to a [report](#), Pie is the most widely used operating system in the android version list.

Features

- Sound amplifier with select to speak options
- [Artificial intelligence \(AI\)](#) compatibility
- Adaptive Battery and Brightness with background restrictions
- Multi-camera support with the external camera compatibility
- New Gesture Navigation and App Actions
- New Screenshot Shortcut key and accessibility menu
- Easier Screen Rotation and edge to edge screens support
- Volume and Sound enhancements
- Selectable Dark Mode
- HDR, HD audio, multiple Bluetooth connections
- Slices and long press to overview selection
- Improved Security features for extra protection
- Digital Wellbeing with app timers, dashboard and do not disturb options
- Android backups and privacy enhancements
- More Notification Information and easier text selection

17. Android 10: Android Q (API 29)

Android version 10 is officially released on September 3, 2019, with enhanced features and functionalities with higher API levels.

Features

- Support for foldable smartphones with flexible displays
- Dark mode for eyes comfortability
- Navigation control over gesture quicker and intuitive ever
- Sound amplifier with more clear sound
- Smart reply suggestions for all messaging apps
- Live caption for media playing on a smartphone
- Undo app removal
- Better notification control with many options

18. Android 11: (API 30)

Android developers are continually working to provide more advanced applications as per the user requirements. Most of the developers are searching [Android Developer vs Web Developer](#) to choose a trending field.

The following are the new features that you will experience in the new android 11 that is going to release and will be the latest version in the android version list.

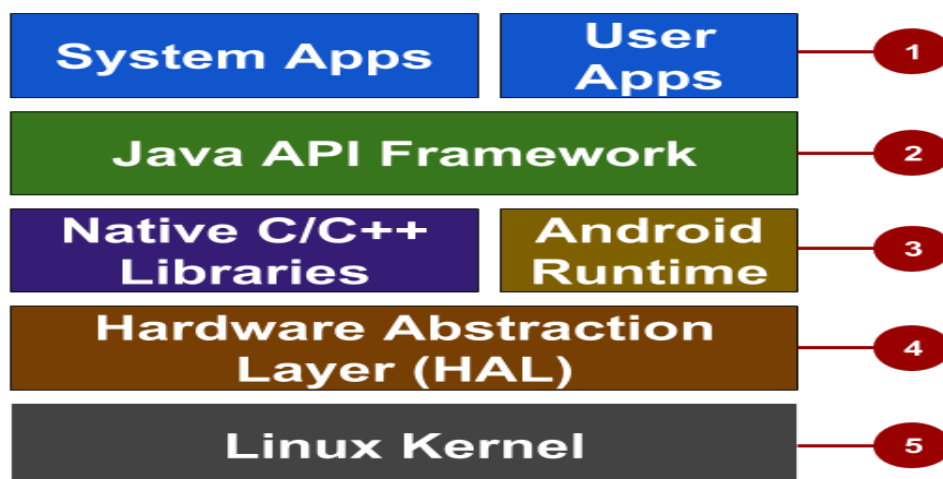
- Native screen recording

- Muting notifications during video
- Increase touch sensitivity
- Notification History
- Auto revoke app permissions

Following are the Revamped or updated features:

- Revamped menu and screenshot shortcuts
- New text selection mode from one app to another
- Undoing recently cleared applications
- Airplane mode doesn't kill Bluetooth anymore
- Face Unlock will require you to open your eyes in pixel 4
- App pinning in the share menu
- Improved notification conversation shades
- Conversation bubbles and context-aware dark mode
- Improved one-time permissions

The following diagram shows the major components of the Android *stack*—the operating system and development architecture.



An Open Platform for Mobile development

There are several platforms that you can use to develop mobile apps, including:

1. **Android**: You can use Java or Kotlin to build native Android apps that can run on devices that use the Android operating system.
2. **iOS**: You can use Swift or Objective-C to build native iOS apps that can run on devices that use the iOS operating system.
3. **Cross-platform**: There are several frameworks and tools that allow you to build apps that can run on both Android and iOS, including:
 - **Flutter**: A mobile app development framework created by Google.
 - **React Native**: A JavaScript framework for building native mobile apps.
 - **Xamarin**: A cross-platform development platform that allows you to build native apps for Android, iOS, and Windows using C#.

4. Hybrid: You can use technologies such as HTML, CSS, and JavaScript to build apps that are essentially websites that are wrapped in a native app shell. These apps can run on multiple platforms, but may not have the same performance and capabilities as native apps.

Which platform you choose will depend on your specific requirements and preferences.

Android is a popular open platform for mobile app development, with a large and active developer community. Some of the key features of the Android platform for mobile development include:

1. Open source: Android is open source, meaning that developers can access and modify the source code. This makes it easy for developers to customize the platform and build innovative apps.
2. Wide range of devices: Android is used on a variety of devices, including phones, tablets, televisions, and even automobiles. This allows developers to reach a large and diverse user base.
3. Strong developer community: There is a strong and active developer community around Android, with a wealth of resources and support available online. This makes it easy for developers to get help and find solutions to problems they encounter.
4. Flexibility: Android allows developers to build native apps using languages such as Java and Kotlin, or to build cross-platform apps using frameworks such as Flutter, React Native, and Xamarin. It is also possible to build hybrid apps using technologies such as HTML, CSS, and JavaScript.
5. Extensive API support: Android provides a comprehensive set of APIs (Application Programming Interfaces) that allow developers to access a wide range of device features and capabilities, including the camera, GPS, and more.

Overall, the Android platform provides a powerful and flexible environment for mobile app development.

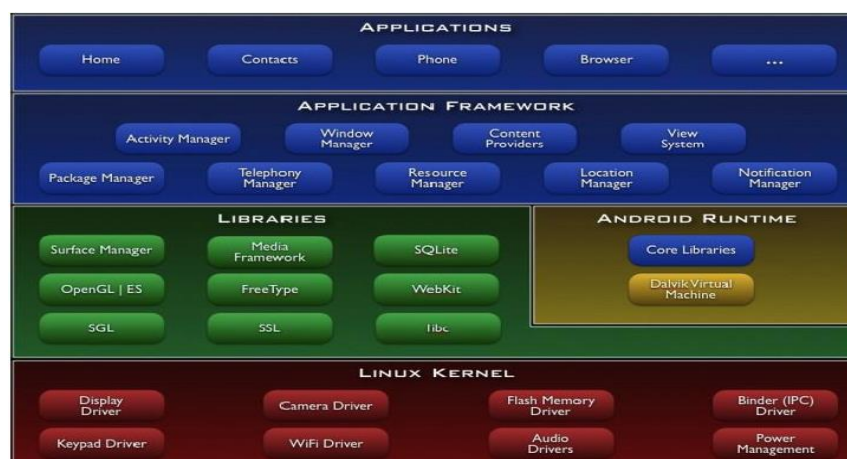
Open Handset Alliance

- The Open Handset Alliance (OHA) is a group of companies that have come together to support and promote the Android mobile operating system.
- The OHA was founded in 2007 by Google, and currently consists of 84 member companies.
- The goal of the OHA is to create a standardized, open platform for mobile devices that allows developers to build innovative and useful apps.
- The OHA also works to promote the adoption of Android by manufacturers of mobile devices, and to encourage the use of Android in the mobile industry.

What does Android run On – Android Internals

Android is a mobile operating system that is designed to run on a wide variety of devices.

At a high level, the internals of the Android operating system can be divided into the following components:



1. **Linux kernel:** Android is based on the Linux kernel, which provides the underlying system-level functionality for the operating system. The kernel handles tasks such as memory management, process management, and networking.
2. **Hardware Abstraction Layer (HAL):** The HAL provides a interface between the hardware and the Android operating system. It enables the operating system to access the hardware resources of the device in a consistent and abstracted manner.
3. **Libraries:** Android includes a set of native libraries that provide a wide range of functionality, including support for graphics, media, and database access.
4. **Android Runtime:** The Android Runtime (ART) is the runtime environment in which Android apps are executed. It includes the Dalvik Virtual Machine, which is responsible for executing code written in languages such as Java and Kotlin.
5. **Application framework:** The Android application framework provides the tools and APIs that developers can use to build Android apps. It includes components such as activity manager, resource manager, and notification manager, which provide core functionality for Android apps.
6. **Applications:** Android includes a set of pre-installed apps that provide core functionality for the operating system, such as the home screen, phone, and messaging apps. It is also possible to install third-party apps from the Google Play Store or other app stores.

Why to use Android for mobile development

There are several reasons why you might choose to use Android for mobile app development:

1. **Wide user base:** Android is the most widely used mobile operating system in the world, with a large and diverse user base. This means that your app has the potential to reach a large audience if it is developed for Android.
2. **Open source:** Android is open source, which means that you have complete access to the source code and can customize it to meet your specific needs. This makes it easy to build innovative and unique apps.
3. **Strong developer community:** There is a strong and active developer community around Android, with a wealth of resources and support available online. This makes it easy to get help and find solutions to problems you may encounter while developing your app.
4. **Flexibility:** Android allows you to build native apps using languages such as Java and Kotlin, or to build cross-platform apps using frameworks such as Flutter, React Native, and Xamarin. It is also possible to build hybrid apps using technologies such as HTML, CSS, and JavaScript.
5. **Extensive API support:** Android provides a comprehensive set of APIs (Application Programming Interfaces) that allow you to access a wide range of device features and capabilities, including the camera, GPS, and more.

Overall, Android is a powerful and flexible platform for mobile app development that offers a wide range of tools and resources for developers.

Developing for Android:

My First Android Application, How to setup Android Development Environment, Android development Framework - [Android-SDK, Eclipse, Emulators](#) – What is an Emulator / Android AVD, Creating & setting up custom Android emulator, [Android Project Framework](#), [My First Android Application](#).

How to setup Android Development Environment

- Android Application Development can be done using Android Studio as well as Eclipse IDE.(We will be using Eclipse IDE to set up Android App Development.)
- We can create android applications in Eclipse IDE using the ADT plugin.
- Eclipse is preferred for creating small android applications.
- Eclipse IDE is an open-source software used by developers, it contains a variety of plugins to develop software in different programming languages.

To set up an Android development environment, (in android studio) you need to do the following:

1. Install the [JDK \(Java Development Kit\)](#) (JDK). You can download the latest version of the JDK from the Oracle website.
2. Download and install [Android Studio](#). You can download Android Studio from the Android website (<https://developer.android.com/>)..
3. Launch Android Studio and go through the setup wizard to install the Android SDK and any necessary dependencies.
4. When the setup wizard finishes, you should be able to see the "Welcome to Android Studio" window. Click "Start a new Android Studio project" to create a new project.
5. In the "Create New Project" window, enter the name of your project, package name, and location where you want to save your project. Then, click "Next".
6. In the "Select the form factors your app will run on" window, select the form factors you want your app to support (e.g. phone, tablet, Android Wear). Then, click "Next".
7. In the "Select a minimum API level" window, select the minimum API level that you want to support. The API level determines the version of Android that your app can run on. Then, click "Next".
8. In the "Add an activity to Mobile" window, select the type of activity you want to create (e.g. Empty Activity, Login Activity). Then, click "Next".
9. In the "Configure activity" window, enter the name of your activity and layout. Then, click "Finish".
10. Android Studio will create a new project and open the main activity file. You are now ready to start developing your Android app!

To set up an Android development environment, (in Eclipse IDE) you need to do the following:

1. Install the [JDK \(Java Development Kit\)](#) (JDK) if it is not already installed on your system from the Oracle website.
2. To install **Eclipse IDE**, click on [Download Eclipse](#). Make sure to install the correct version for your operating system (32-bit or 64-bit).
3. **Install the Android SDK** (Software Development Kit).
 - ✓ Go to the Android Developer website (<https://developer.android.com/>).
 - ✓ Click on the "Download Android Studio" button to download the latest version of Android Studio.

- ✓ When the download is complete, open the downloaded file and follow the instructions to install Android Studio.
 - ✓ During the installation process, you will be asked to choose the components that you want to install. Make sure to select the "Android SDK" component.
 - ✓ Once the installation is complete, you can find the Android SDK in the following directory:
 - ✓ Windows: C:\Users\<user>\AppData\Local\Android\Sdk
 - ✓ Mac: /Users/<user>/Library/Android/sdk
 - ✓ Linux: /home/<user>/Android/sdk
4. Once the Android SDK is installed, you need to configure the **ADT (Android Development Tools) plugin** in Eclipse. To do this, follow these steps:
- ✓ Launch Eclipse, then select "Help" > "Install New Software".
 - ✓ Click "Add", in the top-right corner.
 - ✓ In the "Add Repository" dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location: <https://dl-ssl.google.com/android/eclipse/>
 - ✓ Click "OK".
 - ✓ In the "Available Software" dialog, select the checkbox next to "Developer Tools" and click "Next".
 - ✓ In the next window, you will see a list of the tools to be downloaded. Click "Next".
 - ✓ Read and accept the license agreements, then click "Finish".
 - ✓ If you get a security warning saying that the authenticity or validity of the software can't be established, click "OK".
 - ✓ When the installation completes, restart Eclipse.
5. After the restart, a dialog box will appear for setting up the Preferences. Click on Open Preferences then Click on Proceed. If the dialog box does not appear then go to Eclipse -> Window -> Preferences. In Eclipse, select "Window" > "Preferences" (on Mac, "Eclipse" > "Preferences").

- ✓ Browse the SDK Location of Android (eg. C:\Users\<user>\AppData\Local\Android\Sdk) and Click Apply.

Note 1: SDK Path is also present in Android Studio -> Tools -> SDK Manager -> Copy the Android SDK Location path and paste it here.

or

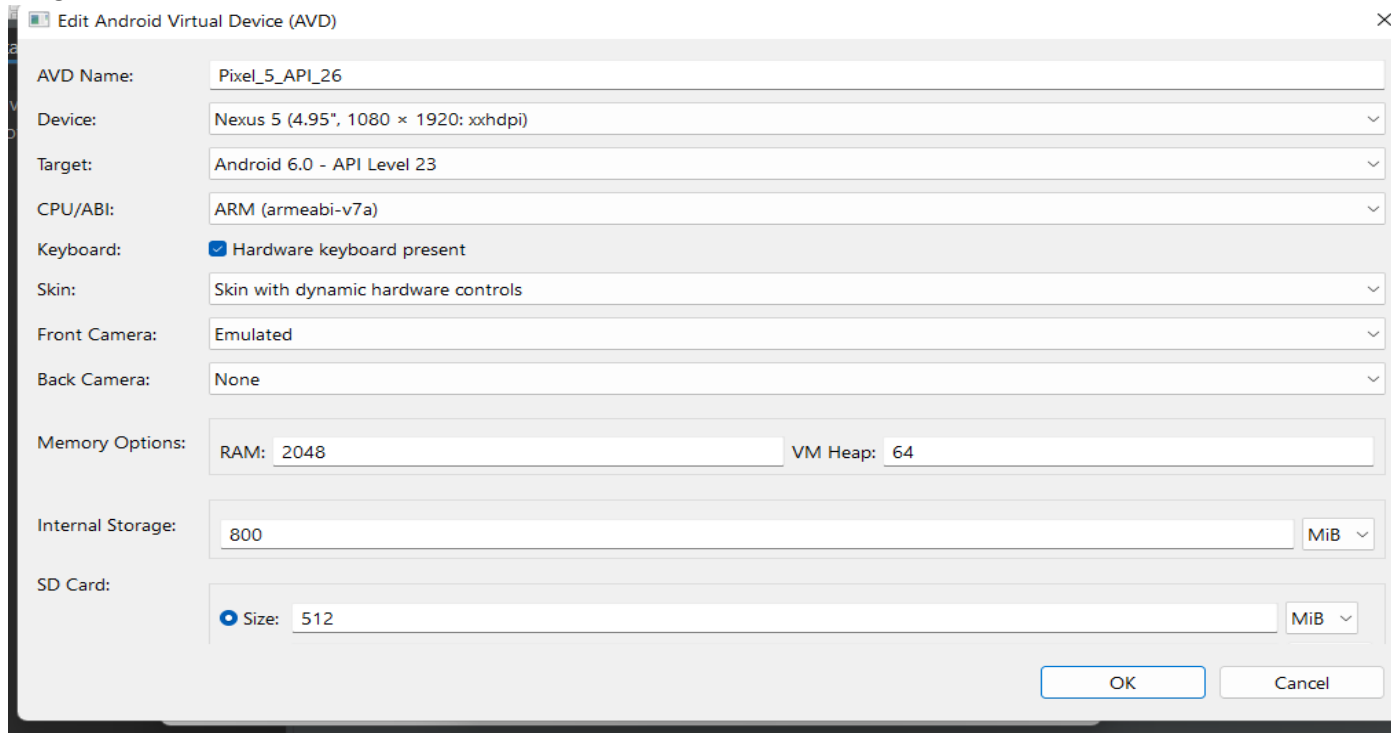
- ✓ If there is no android SDK(means you not install android SDK, AS STEP-3), A new dialog box also appear ,like this: Click on Install new SDK then Next. Another dialog box will appear, Accept all three packages and Click on Install.



or

- ✓ In the left panel, expand "Android" and select "SDK Manager". In the "SDK Manager" window, click "Browse" and locate the directory where you installed the Android SDK. Click "Apply", then "OK".

6. After installation is completed, Go to Eclipse then Select Window then Click on Android Virtual Device Manager. A dialog box will appear, Select existing AVD and Click on Edit. Fill in all the details as per the below image. Click OK.



AVD Name: Pixel_5_API_26

Device: Nexus 5 (4.95", 1080 × 1920: xxhdpi)

Target: Android 6.0 - API Level 23

CPU/ABI: ARM (armeabi-v7a)

Keyboard: ☒ Hardware keyboard present

Skin: Skin with dynamic hardware controls

Front Camera: Emulated

Back Camera: None

Memory Options: RAM: 2048 VM Heap: 64

Internal Storage: 800 MiB

SD Card: ☒ Size: 512 MiB

OK Cancel

7. You are now ready to create your first Android project in Eclipse!
8. To create an android application, Select File -> New -> Other, and then the below dialog box will appear Select Android -> Android Application Project then Click on Next. Follow the steps and then click on Finish.

That's it! You should now be ready to start developing Android apps.

Android development Framework

Android provides a development framework that includes the following components:

1. The **Android operating system** itself, which provides the foundation for running Android apps.
2. The **Android SDK** (Software Development Kit), which includes tools and APIs for building Android apps.
3. The **Android NDK** (Native Development Kit), which allows developers to use native code (such as C and C++) in their Android apps.
4. The Android Studio **Integrated Development Environment** (IDE), which is a specialized software tool for developing Android apps.
5. The **Android Emulator**, which allows developers to test their apps on a variety of different devices and Android versions without the need for physical hardware.
6. The Android support **libraries**, which provide a set of reusable components and utilities that can be used to build Android apps.

Together, these components form a comprehensive development framework that allows developers to create powerful and innovative Android apps.

Android-SDK

- The Android SDK (Software Development Kit) is a set of tools and APIs for building Android apps.

- The Android SDK is an essential component of the Android development environment, and is required for building and testing Android apps. It can be downloaded from the Android website.

It includes the following components:

1. The Android Emulator, which allows you to test your app on a variety of different devices and Android versions without the need for physical hardware.
2. The Android Debug Bridge (ADB), which is a command-line tool that allows you to communicate with an emulator or a physical device.
3. The Android Asset Packaging Tool (AAPT), which is a command-line tool that allows you to package and build Android app packages (APKs).
4. The Android Library, which is a set of code libraries that provide core functionality for Android apps, such as data storage, network communication, and user interface components.
5. The Android Build Tools, which are a set of command-line tools that allow you to build and sign Android app packages.
6. The Android Support Libraries, which provide a set of reusable components and utilities that can be used to build Android apps.

Eclipse

- Eclipse is a popular open-source Integrated Development Environment (IDE) that can be used for developing Android apps.
- It includes a variety of tools for building and testing Android apps, such as a code editor, a debugger, and an emulator.

To use Eclipse for Android development, you need to install the Android Development Tools (ADT) plugin. The ADT plugin adds support for Android projects to Eclipse, and allows you to build and run Android apps from within the Eclipse environment.

- To install the ADT plugin in Eclipse:
- Open Eclipse and click on "Help" in the menu bar.
- Click on "Install New Software".
- In the "Work with" field, enter the URL for the ADT plugin: <https://dl-ssl.google.com/android/eclipse/>.
- Click "Add" to add the repository to the list.
- Select the checkbox next to "Developer Tools" and click "Next".
- Follow the prompts to complete the installation.

After the ADT plugin is installed, you can create a new Android project in Eclipse by going to "File" > "New" > "Android Application Project".

Emulators – What is an Emulator / Android AVD

- An emulator is a software program that allows a computer or device to imitate the functions of another device.
- In the context of mobile development, **an emulator is a tool that allows developers to test their app on a variety of different devices and Android versions without the need for physical hardware.**

There are two types of emulators that are commonly used in Android development:

1. Hardware emulators: These emulators use hardware acceleration to imitate the functions of a specific device. Hardware emulators are generally more accurate and faster than software emulators, but they require more resources and are slower to set up.

2. Software emulators: These emulators use software to imitate the functions of a specific device. Software emulators are generally slower and less accurate than hardware emulators, but they are easier to set up and require fewer resources.

An **Android Virtual Device** (AVD) is an emulator that allows you to test your Android app on a variety of different devices and Android versions.

You can create and configure an AVD in the AVD Manager, which is included in the Android Studio IDE.

Creating & setting up custom Android emulator

To create a custom Android emulator in Eclipse:

1. Open the Android Virtual Device Manager by going to "Window" > "AVD Manager" in the menu bar.
2. Click the "New" button to open the "Create new Android Virtual Device (AVD)" dialog.
3. Enter a name for the AVD and select a device definition.
4. Select the desired Android version and click "Next".
5. (Optional) Customize the hardware and device options for the emulator, such as the screen size, the amount of memory, and the device skin. You can also create a custom skin by clicking the "Create Skin" button.
6. Click "Finish" to create the emulator.

To set up a custom emulator:

1. Open the Android Virtual Device Manager and select the emulator you want to set up.
2. Click the "Edit" button to open the "Edit Android Virtual Device (AVD)" dialog.
3. Modify the hardware and device options as desired.
4. Click "OK" to save the changes.

After you set up the custom emulator, you can start it by clicking the "Start" button in the Android Virtual Device Manager. This will launch the emulator, allowing you to test your app on the virtual device.

To create a custom Android emulator in Android Studio:

1. Open the AVD Manager by going to "Tools" > "AVD Manager" in the menu bar.
2. Click the "Create Virtual Device" button to open the "Select Hardware" screen.
3. Select the type of device you want to emulate and click "Next".
4. Select the Android version you want to run on the emulator and click "Next".
5. (Optional) Customize the hardware and device options for the emulator, such as the screen size, the amount of memory, and the device skin. You can also create a custom skin by clicking the "Create Skin" button.
6. Click "Finish" to create the emulator.

To set up a custom emulator:

1. Open the AVD Manager and select the emulator you want to set up.
2. Click the "Edit" button to open the "Edit Android Virtual Device (AVD)" dialog.
3. Modify the hardware and device options as desired.
4. Click "OK" to save the changes.

After you set up the custom emulator, you can start it by clicking the "Play" button in the AVD Manager. This will launch the emulator, allowing you to test your app on the virtual device.

My First Android Application

To create your first Android app in **Android Studio**, follow these steps:

1. Launch Android Studio and create a new project by going to "File" > "New" > "New Project".
2. Choose a name and location for your project, and select the "Phone and Tablet" option under "Form Factors". Then, choose the "Empty Activity" option and click "Next".
3. On the "Configure Activity" page, keep the default settings and click "Finish".
4. Android Studio will now create a new project for you with the basic files and structure needed for an Android app.
5. To start building your app, open the "activity_main.xml" file located under "app > res > layout" in the project tree on the left side of the Android Studio window. This file defines the layout of your app's user interface.
6. In the "activity_main.xml" file, you can use the visual designer or the XML code to add UI elements to your app's layout. You can also use the "Design" tab at the bottom of the window to preview your layout as you design it. To add functionality to your app, open the "MainActivity.java" file located under "app > java" in the project tree. This file contains the Java code that runs when your app is launched.
7. In the "MainActivity.java" file, you can write Java code to respond to user actions, such as button clicks, and perform tasks, such as connecting to the internet or saving data to a database.
8. When you're ready to test your app, click the "Run" button in the toolbar to build and run your app. You can choose to run your app on an emulator or on a physical device.

Congratulations, you've just created your first Android app!

To create your first Android app in **Eclipse**, follow these steps:

1. Launch Eclipse and create a new Android project by going to "File" > "New" > "Android Project".
2. Enter a name for your project and select the desired Android version.
3. Select the "Create new project in workspace" option and click "Finish".
4. Eclipse will now create a new project for you with the basic files and structure needed for an Android app.
5. To start building your app, open the "activity_main.xml" file located under "res > layout" in the project tree on the left side of the Eclipse window. This file defines the layout of your app's user interface.
6. In the "activity_main.xml" file, you can use the visual designer or the XML code to add UI elements to your app's layout.
7. To add functionality to your app, open the "MainActivity.java" file located under "src" in the project tree. This file contains the Java code that runs when your app is launched.
8. In the "MainActivity.java" file, you can write Java code to respond to user actions, such as button clicks, and perform tasks, such as connecting to the internet or saving data to a database.
9. When you're ready to test your app, click the "Run" button in the toolbar to build and run your app. You can choose to run your app on an emulator or on a physical device.

Congratulations, you've just created your first Android app in Eclipse!

Android Project Framework

An Android project is a project that is created and managed using the Android Studio Integrated Development Environment (IDE). An Android project contains all of the files and resources that are needed to build an Android app, including source code, resource files, and an AndroidManifest.xml file.

The basic structure of an Android project is as follows:

1. The "app" directory contains the source code and resource files for your app. It includes the following subdirectories:
 - "src": This directory contains the Java source code files for your app.
 - "res": This directory contains the resource files for your app, such as layouts, strings, and images.
 - "manifests": This directory contains the AndroidManifest.xml file, which defines the components and permissions of your app.
2. The "build" directory contains the build output for your app, including the compiled Java code and resources.
3. The "gradle" directory contains the build files and dependencies for the Gradle build system, which is used to build and package your app.
4. The "libs" directory contains any third-party libraries that your app depends on.
5. The "local.properties" file contains the local configuration for your project, such as the path to the Android SDK.
6. The "build.gradle" file contains the build configuration for your project, including the dependencies and build options.

Overall, the Android project framework provides a structure for organizing and building.

Android Activities and UI Design

Understanding Intent, Activity, Activity Lifecycle and Manifest, Creating Application and new Activities, Expressions and Flow control, Android Manifest,

Fundamental Android UI Design, Simple UI -Layouts and Layout properties, Creating new Layouts, Drawable Resources, Resolution and density independence (px,dip,dp,sip,sp), XML Introduction to GUI objects viz., Push Button Text / Labels, EditText, ToggleButton, WeightSum, Padding, Layout Weight

Intent

- Intents are an important part of the Android operating system, as they provide a way for apps to communicate and interact with each other.
- In Android, an Intent is an object that represents an intention to perform some action.
- Intents are used to request an action from another app component, such as starting an activity, delivering a broadcast, or starting a service. Intents can also be used to transfer data between apps.
- Intents can specify a desired action to perform, such as VIEW, EDIT, or SEND.
- They can also include additional data to be used in performing the action, such as a URI or a MIME type.

There are two types of intents in android

1. Implicit
2. Explicit

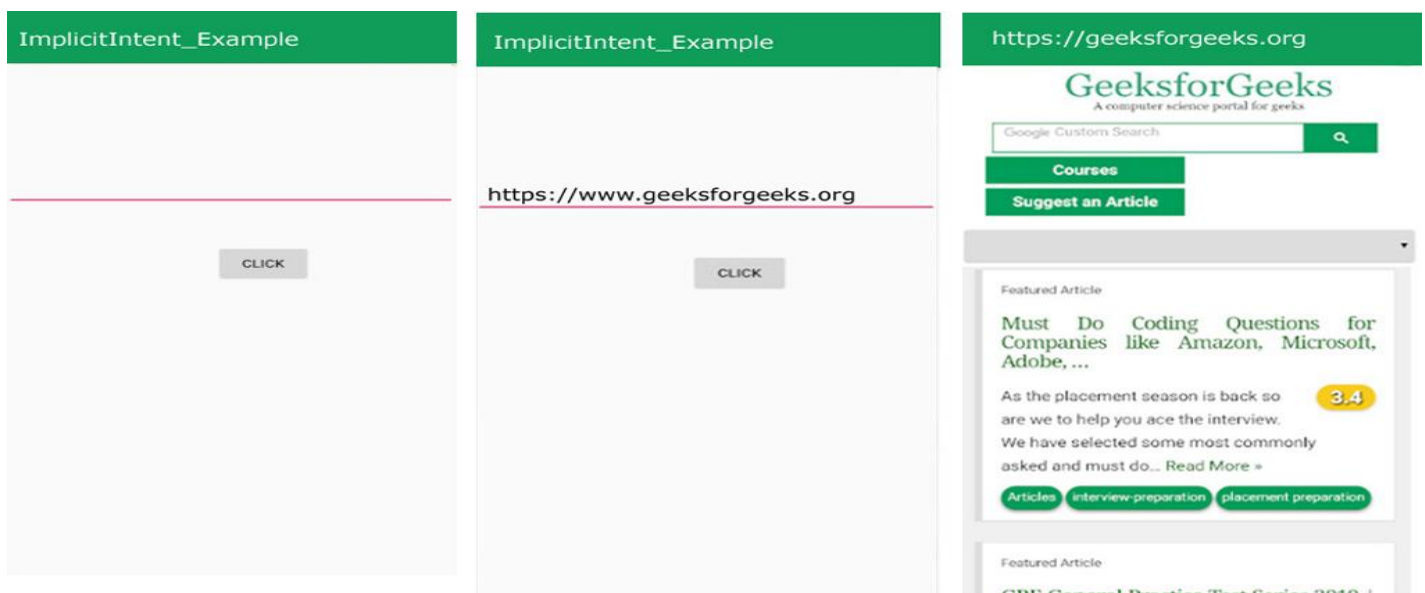
Implicit Intent

[Implicit Intent](#) doesn't specify the component. In such a case, intent provides information on available components provided by the system that is to be invoked. For example, you may write the following code to view the webpage.

Syntax:

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("https://www.geeksforgeeks.org/"));
startActivity(intent);
```

For Example: In the below images, no component is specified, instead, an action is performed i.e. a webpage is going to be opened. As you type the name of your desired webpage and click on the 'CLICK' button. Your webpage is opened.



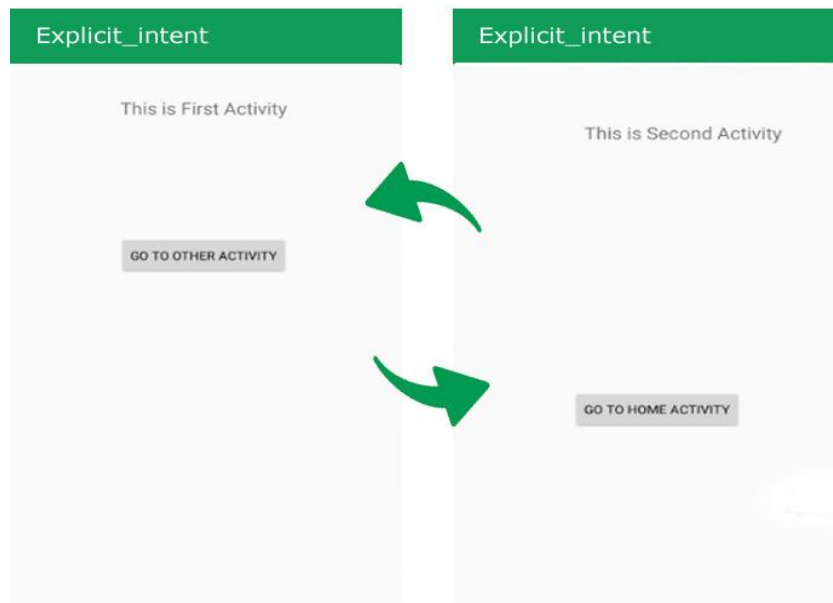
Explicit Intent

[Explicit Intent](#) specifies the component. In such a case, intent provides the external class to be invoked.

Syntax:

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
startActivity(i);
```

For Example: In the below example, there are two activities (FirstActivity, and SecondActivity). When you click on the 'GO TO OTHER ACTIVITY' Button in the FirstActivity, then you move to the SecondActivity. When you click on the 'GO TO HOME ACTIVITY' button in the SecondActivity, then you move to the first activity. This is getting done through Explicit Intent.



Activity

- An activity is a single, focused thing that a user can do. or
- In Android, an **activity** is referred to as one screen in an application. It is very similar to a single window of any desktop application. An Android app consists of one or more screens or activities.
- For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity to view an email.
- Activities in Android are used to represent the UI of an app.
- An activity is implemented as a subclass of the **Activity** class and typically consists of a layout file and corresponding Java code. The layout file defines the UI of the activity, and the Java code defines the behavior of the activity.

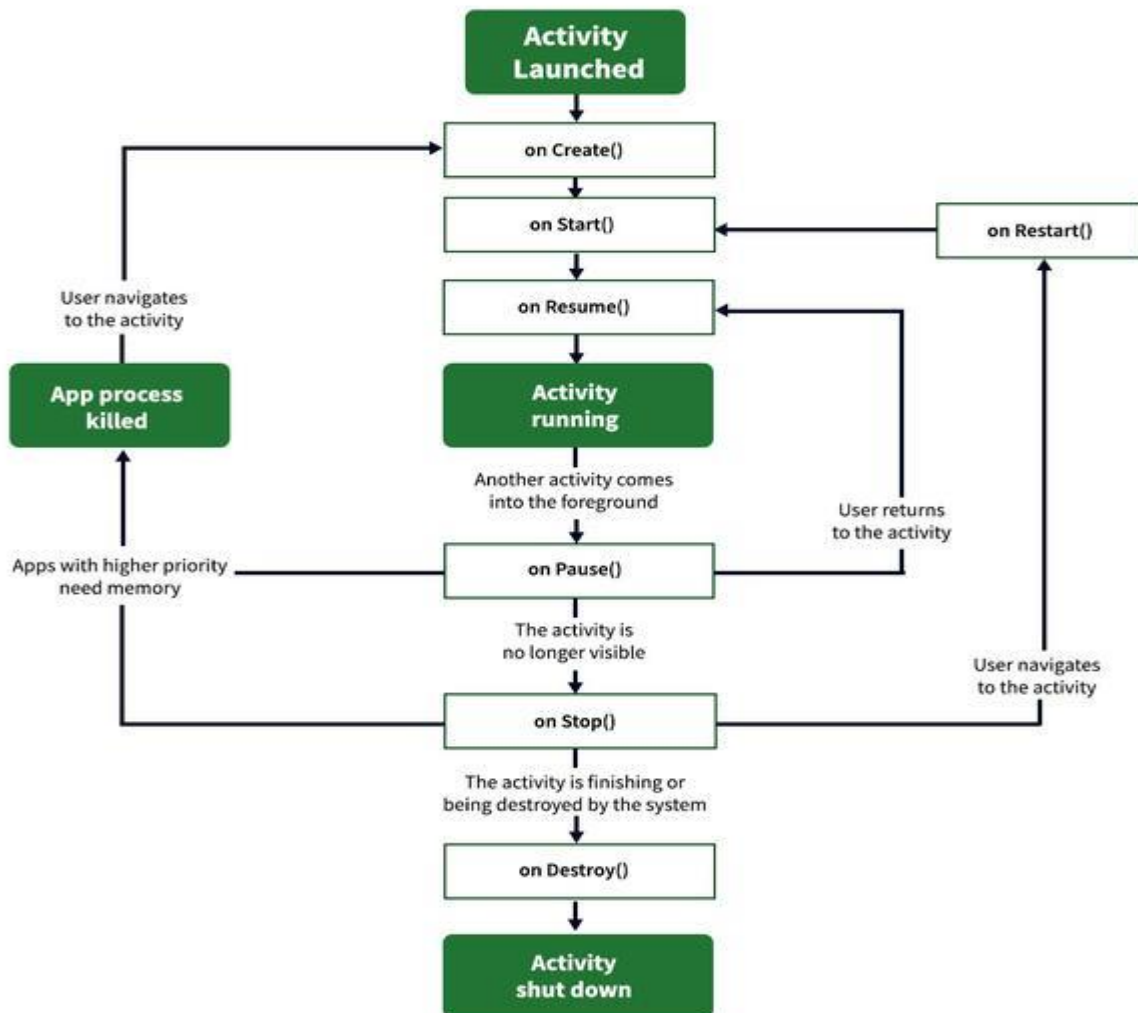
Activity Lifecycle

Each activity goes through various stages or a lifecycle and is managed by activity stacks. So when a new activity starts, the previous one always remains below it. There are four stages of an activity.

1. If an activity is in the foreground of the screen i.e at the top of the stack, then it is said to be active or running. This is usually the activity that the user is currently interacting with.
2. If an activity has lost focus and a non-full-sized or transparent activity has focused on top of your activity. In such a case either another activity has a higher position in multi-window mode or the activity itself is not focusable in the current window mode. Such activity is completely alive.

3. If an activity is completely hidden by another activity, it is stopped or hidden. It still retains all the information, and as its window is hidden thus it will often be killed by the system when memory is needed elsewhere.
4. The system can destroy the activity from memory by either asking it to finish or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

For each stage, android provides us with a set of 7 methods that have their own significance for each stage in the life cycle. The image shows a path of migration whenever an app switches from one state to another.



Activity Lifecycle in Android

Detailed introduction on each of the method is stated as follows:

1. onCreate()

It is called when the activity is first created. This is where all the static work is done like creating views, binding data to lists, etc. This method also provides a Bundle containing its previous frozen state, if there was one.

2. onStart()

It is invoked when the activity is visible to the user. It is followed by onResume() if the activity is invoked from the background. It is also invoked after onCreate() when the activity is first started.

3. onRestart()

It is invoked after the activity has been stopped and prior to its starting stage and thus is always followed by `onStart()` when any activity is revived from background to on-screen.

4. `onResume()`

It is invoked when the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, with a user interacting with it. Always followed by `onPause()` when the activity goes into the background or is closed by the user.

5. `onPause()`

It is invoked when an activity is going into the background but has not yet been killed. It is a counterpart to `onResume()`. When an activity is launched in front of another activity, this callback will be invoked on the top activity (currently on screen). The activity, under the active activity, will not be created until the active activity's `onPause()` returns, so it is recommended that heavy processing should not be done in this part.

6. `onStop()`

It is invoked when the activity is not visible to the user. It is followed by **`onRestart()`** when the activity is revoked from the background, followed by `onDestroy()` when the activity is closed or finished, and nothing when the activity remains on the background only. Note that this method may never be called, in low memory situations where the system does not have enough memory to keep the activity's process running after its `onPause()` method is called.

7. `onDestroy()`

The final call received before the activity is destroyed. This can happen either because the activity is finishing (when `finish()` is invoked) or because the system is temporarily destroying this instance of the activity to save space. To distinguish between these scenarios, check it with **`isFinishing()`** method.

Manifest

- The Android Manifest is a file that is included in every Android app and contains essential information about the app.
- It is used by the Android operating system to install and manage the app, and to declare the app's components, such as activities, services, and broadcast receivers.
- The manifest file is an XML file called **`AndroidManifest.xml`** and is located in the root directory of the app.
- It contains elements that provide information about the app's package name, version, and target API level, as well as the app's required permissions and features.
- This manifest file declares an **`application`** element with a single **`activity`** element and a **`service`** element. It also declares a **`receiver`** element, which is a component that listens for and responds to broadcast messages. The manifest file also includes a **`uses-permission`** element, which indicates that the app requires the INTERNET permission to access the internet.
- Here is an example of the structure of an Android Manifest file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">
    <application android:label="@string/app_name">
        <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>
```

```

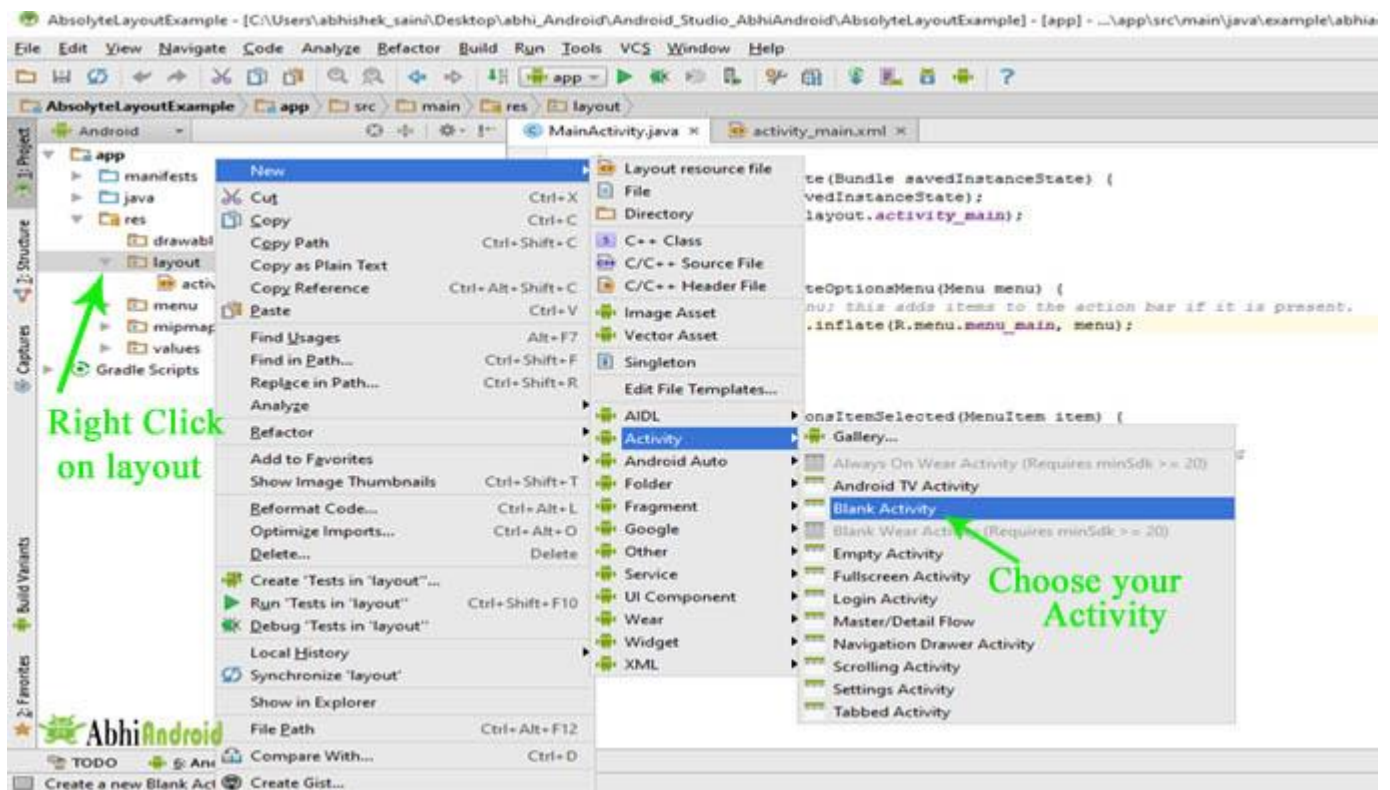
</intent-filter>
</activity>
<service android:name=".MyService"> </service>
<receiver android:name=".MyReceiver"> </receiver>
</application>
<uses-permission android:name="android.permission.INTERNET" />
</manifest>

```

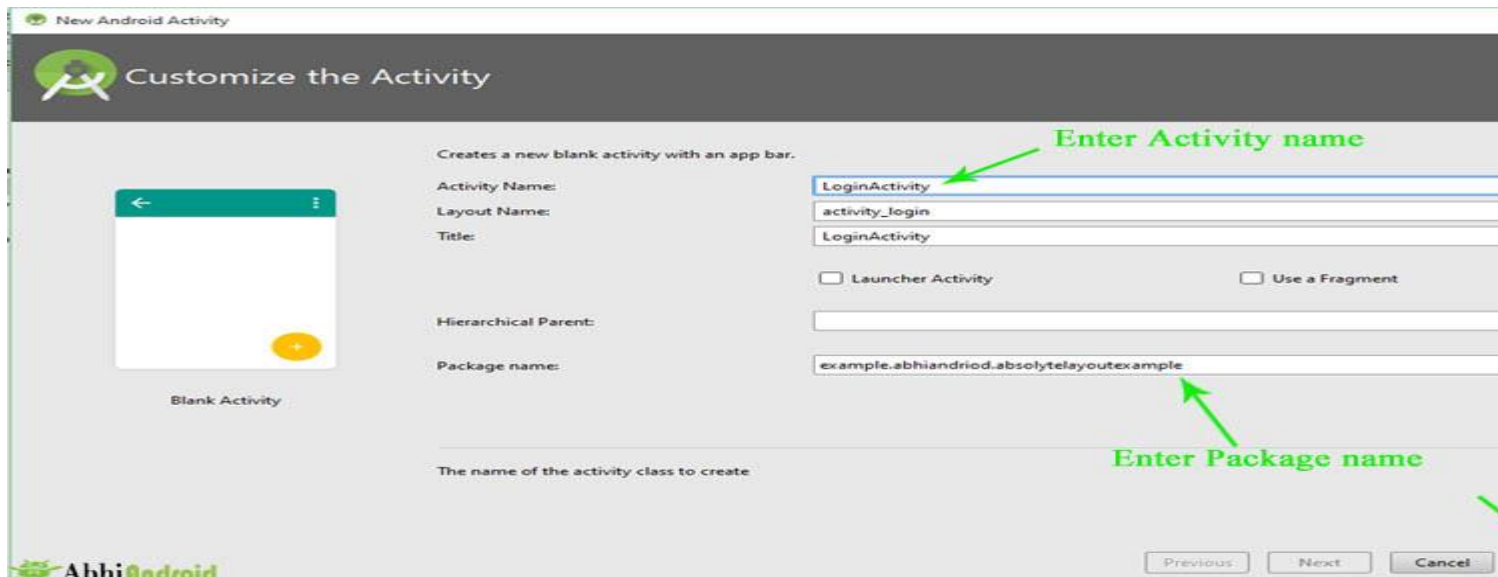
Creating Application and new Activities

How To Create New Activity in Android Studio:

Step 1: Firstly, click on app > res > layout > Right Click on layout. After that Select New > Activity and choose your Activity as per requirement. Here we choose Blank Activity as shown in figure below.



Step 2: After that Customize the Activity in Android Studio. Enter the "Activity Name" and "Package name" in the Text box and Click on Finish button.



Step 3: After that your new Activity in Layout will be created. Your XML Code is in Text and your Design Output is in Design.

Expression and flow control

- In Android, you can use expressions and flow control statements to control the flow of execution of your code.
- An expression is a combination of values, variables, and operators that represents a value. In Android, you can use expressions to perform calculations, assign values to variables, and to specify conditions in flow control statements.
- Flow control statements allow you to specify the order in which your code is executed. In Android, you can use the following flow control statements:

- `if` statements: used to execute a block of code if a condition is true
- `if-else` statements: used to execute a block of code if a condition is true, and another block of code if the condition is false
- `for` loops: used to execute a block of code multiple times, with a counter variable
- `while` loops: used to execute a block of code multiple times, as long as a condition is true
- `do-while` loops: used to execute a block of code at least once, and then repeat the block as long as a condition is true

Fundamental Android UI Design

Designing a good user interface (UI) for an Android app is important, as it can improve the user experience and make the app more enjoyable to use. Here are some fundamental principles to consider when designing a UI for an Android app:

1. Keep it simple: Avoid clutter and unnecessary elements, and focus on the most important features and functionality of the app.
2. Use consistent design patterns: Follow Android design guidelines and use standard UI elements and design patterns to make the app feel familiar and easy to use.
3. Use appropriate layout and hierarchy: Use appropriate layouts, such as `LinearLayout` or `ConstraintLayout`, and organize UI elements in a logical hierarchy to help guide the user through the app.
4. Provide clear feedback: Use visual and auditory feedback to let the user know what's happening in the app and to confirm that their actions have been successful.
5. Test and iterate: Test the app with real users and gather feedback to identify areas for improvement. Then, iterate and make changes to improve the overall UI design.

By following these principles, you can create a UI for your Android app that is effective, intuitive, and enjoyable to use.

UI Layouts

- In Android, the user interface of an app is made up of layouts.
- Layouts are containers that hold and organize UI elements such as buttons, text views, and image views.
- Layouts are defined in XML files and are inflated by the app at runtime. There are several types of layouts available in Android, including:
 - `LinearLayout`: arranges its children in a single row or column
 - `RelativeLayout`: arranges its children relative to each other or to the parent layout
 - `FrameLayout`: displays a single child element at a time, with the most recently added child appearing on top
 - `ConstraintLayout`: a flexible layout that allows you to create complex, responsive layouts using constraints to specify how UI elements should be positioned and sized
- Here is an example of a simple `LinearLayout` in Android:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 1" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 2" />
```

```
</LinearLayout>
```

This layout defines a horizontal `LinearLayout` with two `Button` elements as children. The layout has a width of `"match_parent"`, which means it will take up the full width of its parent, and a height of `"wrap_content"`, which means it will be as tall as its children.

Layout properties

In Android, layout properties are attributes that you can set on layout elements to control their appearance and position on the screen.

There are many layout properties available in Android, including:

- **layout_width** and **layout_height**: used to set the width and height of a layout element, respectively
- **layout_margin** and **layout_padding**: used to set the margin and padding of a layout element, respectively
- **layout_gravity**: used to specify how a layout element should be positioned within its parent layout
- **gravity**: used to specify how the content of a layout element should be positioned within the element itself
- **visibility**: used to specify whether a layout element should be visible or hidden
- **alpha**: used to specify the transparency of a layout element

Example of a layout element with some layout properties set-

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:layout_gravity="center_horizontal"
    android:gravity="center"
    android:text="Button" />
```

Creating new Layouts

Here is a general outline of the steps to create a new layout in Android:

1. Start Android Studio and open your project.
2. In the Project pane, navigate to the **res/layout** directory.
3. Right-click on the **layout** directory and select "New > Layout resource file".
4. Enter a name for the layout file and click "OK".

5. Android Studio will open the layout file in the XML editor.
6. Add layout elements to the file using XML tags. For example, you can use a **LinearLayout** element to create a linear layout and add **Button** elements as children.
7. Use layout properties to control the appearance and position of the layout elements. For example, you can use the **layout_width** and **layout_height** properties to set the size of a layout element.
8. Save the layout file.

Here is an example of a simple LinearLayout in Android: