Mobile Application Development

Unit – 1 Chapter - 1

1. Introduction:

1.1 What is Mobile technology?

- The Latin term mobile means "ability to move" or portable. Therefore, mobile technology means, technology that is portable.
- **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. It is used to send data from one system to another.
- Mobile technology is technology that goes where the user goes. It consists of portable two-way communications devices, computing devices and the networking technology that connects them.
- Regardless of the context, mobile technology enables people to complete tasks from any location, adding more flexibility to tech users' everyday lives.

1.2 History of Mobile how it was developed?

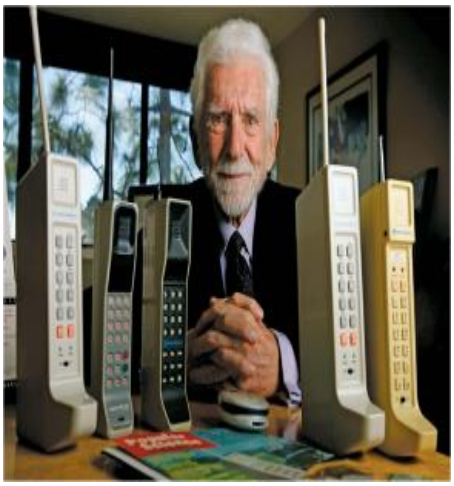
Mobile technology refers to the technology that is specifically designed to be used in mobile (or portable) devices. A mobile phone is a portable telephone that can make and receive calls over a radio frequency carrier while the user is moving within a telephone service area. The radio frequency link establishes a connection to the switching systems of a mobile phone operator, which provides access to the Public Switched Telephone Network (PSTN). Most modern mobile telephone services use cellular network architecture, and therefore mobile telephones are often also called cellular telephones or cell phones.

- **History of Mobile Phones:** Alexander Graham Bell invented telephone and 1878 he made the first phone call. Almost a century later Motorola introduced

some of its first cell phones during the 1980s. No SMS No Touch Screen No GPS No Camera No Music No Bluetooth, Those phones were completely different from the devices we have today and weren't nearly as cost effective and handy like today's phones.

- Early telephones were connected with wires. Every early phone call was put through by hand! The telephone operators spoke to each person who wanted to make a call and then connected them to the person they wanted to speak to.

- The first cell phone the first mobile phone developed by Motorola in 1973. It was Martin Cooper who placed the first call at AT&T Bells Labs from the streets of New York.These mobile phone that you could hold in your hand.
- This phone was very heavy and large. It was nicknamed "the brick" and weighed 1.1 kg and measured 23cm x 13cm x 4.5cm.You could only use the phone for 35 minutes and it took 10 hours to recharge!

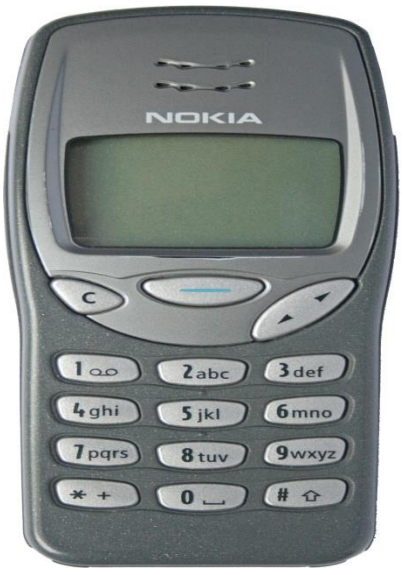


- 1984 Nokia Mobira Talkman the Phone weighed under 5 kgs and it was the world's first transportable phones. People want to be able to talk to each other when they are away from their homes and offices.
- These mobile phone services were built into cars, but they weren't very easy to use. To make a call you first had to speak to a telephone operator who put the call through for you. Only a few people could make calls at the same time.

- In 1989 Motorola Microtac was the smallest and lightest available phone at the time of its release. It was called MicroTac Pocket Cellular Telephone. It was designed in such a way that it could easily fit in any ones shirt pocket!

- First digital-sized mobile phone from Motorola introduced in 1992. This was the first handset that gave the world an idea of Flip Phones.
- The first Smartphone was developed by IBM & BellSouth which was released to the public in 1993.

- **1999 Nokia 8210** The lightest and smallest available Nokia phone at that time. This phone had the feature of speed dial in which the user can assign a name to each key on keypad. The phone uses SMS (Short Message Service) with predictive text input, with support for major European languages.



- The first cellular telephone to feature the new operating system was the T-Mobile G1, released on October 22, 2008. In 2012 Android became the most popular operating system for mobile devices, surpassing Apple's iOS, and, as of 2020, about 75 percent of mobile devices run Android.

- **2007 Apple iPhone** This phone completely changed the definition of a Smartphone. The iPhone is a line of smart phones designed by Apple Inc. This phone runs on Apples iOS mobile operating system.
- Mobile phones enable communication of voice, images, text and video. The important fact is that these could be shared with anyone in any corner of the world at the demand of the user.
- Communication is no longer the only service mobile technology offers. It offers a wide range of services such as access to the World Wide Web, view television and movies, interact with GPS, play games and read and respond to barcode and augmented reality messages.
- The history of mobile technologies originated with the limited use of radio frequencies, where the ability to establish simultaneous two-way communication (full duplex) was considered a technological feat. From the social perspective, mobile technologies began as a rare device used by limited personnel who needed to communicate to others in real time emergencies.
- From the user perspective, the history of mobile technologies began with the use of two way radios and evolved to the current state of prolific smart phones, tablets, and other mobile devices.
- Popularity of the technology sky rocketed with the invention of "smart phones".

2. History of Mobile Technology:

- The history of mobile phone and mobile technology can be said to have begun with the first generation of wireless mobile technologies.

5G (Fifth Generation):

- The latest generation of mobile technology, 5G, promises even faster data speeds, lower latency, and increased network capacity. It facilitates the development of new applications like augmented reality (AR), virtual reality (VR), and the Internet of Things (IoT). 5G is reported to be up to 100 times faster at sending and receiving signals than 4G.

4G (Fourth Generation)

- The first release of 4G was sometimes referred to as 3.9G or LTE (Long-Term Revolution). It was first launched in Oslo in 1998 before being adopted around the world.
- 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.
- 4G networks significantly enhanced data speeds, allowing for high-quality video streaming, online gaming, and improved overall mobile internet performance. These give even faster speeds and make it possible to play complex games and watch films on a mobile phone.

3G (First Generation):

- 3G became a truly global standard and combined the best of competing technologies in a single standard. 3G evolutions were mainly centered around high speed data applications.
- 3G networks (UMTS FDD and TDD, CDMA2000 1x EVDO, CDMA2000 3x, TD-SCDMA, Arib WCDMA, EDGE, IMT-2000 DECT) are newer cellular networks that have data rates of 384kbit/s and more.
- 3G networks improved data transfer rates, enabling mobile internet access, video calling, and faster data services.
- These networks made it possible to download information much faster and surf (Browsing) the web on a mobile phone.

2G (First Generation):

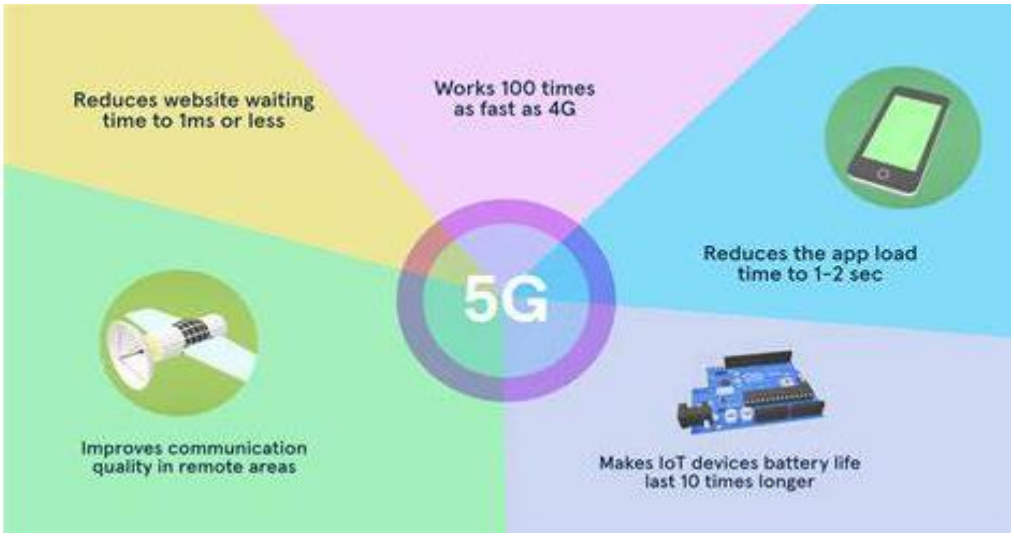
- 2G networks (GSM, CDMAOne, D-AMPS) are the first digital cellular systems launched early 1990s, offering improved sound quality, better security and higher total capacity.
- GSM supports circuit-switched data (CSD), allowing users to place dial-up data calls digitally, so that the network's switching station receives actual ones and zeroes rather than the screech of an analog modem. 2G networks with theoretical data rates up to about 144kbit/s.

- In 2G, roaming and SMS messaging were introduced and were later enhanced with GPRS (General Packet Radio Service) and GSM (Global System for Mobile Communication) for data communication.
- SMS messaging and GPRS became widely used for basic telemetry. Roaming made mobile technology suitable for deployments in multiple countries.
- Telenor was one of the first operators to offer M2M communications with things connected over the 2G network as early as the 1990s.

1G (First Generation):

- 1G network (NMT, C-Nets, AMPS, TACS) are considered to be the first analog cellular systems, which started early 1980s. There were radio telephone systems even before that.
- 1G used the analog system and signals. The drawback with analog signals is that they can't cover a long distance. It was used only for the voice services within one country no roaming initially phone was Large and Bulky.

5 G :



Different Types of Mobile Technology:

- Mobile technology refers to any technology designed to be used on a mobile device, such as smart phones or tablets. It encompasses a wide range of technologies, including mobile apps, mobile websites, and mobile operating systems. Mobile technology has become increasingly popular in recent years as more and more people rely on their mobile devices for everyday tasks such as communication, entertainment, and productivity.

Cellular Technology:

- Cellular technology is one of the most popular types of mobile technologies today. It refers to cellular networks, which allow mobile devices to connect to the internet and make phone calls. With cellular technology, users can access various services and applications on their smart phones or tablets, such as email, social media, streaming videos, and online shopping.
- 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.
- The widespread adoption of cellular technology has revolutionized communication and made it easier for people to stay connected wherever they go.

WiFi :

- The next type of wireless mobile technology is Wi-Fi, which is highly popular. It allows devices to connect to the Internet wirelessly, allowing users to access online resources without needing a physical connection. WiFi technology uses radio waves to transmit data between devices and a wireless router, enabling seamless communication and internet access.
- With the increasing demand for connectivity on the go, WiFi technology has become essential in homes, offices, public spaces, and even vehicles. Its widespread availability and ease of use have made it a preferred choice for connecting smart phones, tablets, laptops, and other mobile devices to the Internet.

5G:

- The latest generation of communication, 5G, has brought with it a plethora of exciting benefits. With faster speeds, reduced latency, and more reliable connectivity, it has revolutionized how we communicate. We can now enjoy lightning-fast data transmission, seamless connectivity for multiple devices simultaneously, and enhanced coverage, ensuring a more reliable connection.

4G :

- 4G networking is the fourth generation of mobile technology, providing faster data speeds and improved network performance compared to its predecessor, 3G. With 4G, users can experience download speeds of up to 100 megabits per second (Mbps), making activities such as streaming high-definition videos and downloading large files much quicker and more efficient.

Bluetooth:

- Rather than connect devices to the internet, Bluetooth networks connect devices to other devices via short-wavelength radio waves. With Bluetooth technology, users can quickly pair devices such as headsets and speakers with desktops, laptops, and phones.

SMS:

- SMS stands for Short Message Service. It is a text messaging service that allows the exchange of short text messages between mobile devices. SMS messages

typically have a maximum length of 160 characters and can be sent and received on various mobile networks. But with the introduction of advanced technologies and updates, the limit has increased to over 700 while maintaining the same concept.

MMS:

- MMS stands for Multimedia Messaging Service. It is a standard way to send multimedia content such as images, videos, audio files, and contact cards between mobile devices using a cellular network. MMS is an extension of the Short Message Service (SMS), which is used for sending text messages.

Key Mobile Application Services :

Certainly! When it comes to **mobile application services**, there are several crucial aspects to consider. Let's explore some of the key services:

1. **User Sign-up/Sign-in and Management**:
    - This service involves creating a seamless experience for users to register, sign in, and manage their accounts within the mobile app. It includes features like email-based registration, social login (such as Face book or Twitter), and password recovery.
2. **Social Login**:
    - Social login allows users to sign in to your app using their existing social media credentials (e.g., Face book, Google, Twitter). It simplifies the authentication process and enhances user convenience.
3. **Analytics and User Engagement**:
    - Analytics services help track user behavior within the app. By analyzing data such as user interactions, session duration, and conversion rates, you can make informed decisions to improve the app's performance and engagement.
4. **Push Notifications**:
    - Push notifications keep users informed and engaged by sending timely updates, reminders, or personalized messages directly to their devices. Effective push notification strategies can enhance user retention and drive app usage.
5. **Real Device Testing**:
    - Ensuring your app works flawlessly across various devices and operating systems is essential. Real device testing involves testing

your app on actual devices (not just simulators) to identify any issues related to performance, compatibility, or usability.

Remember that these services play a critical role in delivering a successful mobile app experience. Whether you're developing for iOS, Android, or cross-platform, thoughtful implementation of these services contributes to user satisfaction and app success.

Mobile application development

- A mobile application (also called a mobile app) is a type of application designed to run on a mobile device, which can be a Smartphone or tablet. Even if apps are usually small software units with limited function, they still manage to provide users with quality services and experiences.
- Apps are divided into two broad categories:

Native apps and web apps

Native App:

- Native apps are developed for specific platforms, such as iOS for iPhones or iPads, Android for Android devices, or Windows for Windows-based devices.
- They are built using platform-specific programming languages and development frameworks (e.g., Swift or Objective-C for iOS, Java or Kotlin for Android).
- Native apps have access to device-specific features and functionalities, such as camera, GPS, accelerometer, and push notifications, allowing for more seamless integration with the device's hardware and software.
- When you want to get a native app for your device, you usually go to the app stores (e.g., Apple App Store, Google Play Store), making them easily find and download the apps.
- Native apps often provide better performance and responsiveness compared to web apps, as they are optimized for the specific platform they are built for.
- **Examples** of popular native apps include Instagram, WhatsApp, and Pokémon GO, Spotify, Google maps,.

Web App:

- Web apps are accessed through a web browser and do not need to be downloaded or installed on a device.
- They are developed using web technologies such as HTML, CSS, and JavaScript, and can be accessed on any device with a web browser, regardless of the operating system.
- Web apps are often platform-independent, making them accessible across various devices and operating systems without the need for separate versions.
- They can be easily updated, as changes made to the web app are immediately reflected for all users accessing it through the web.

There are several types of apps currently available.

- **Gaming apps:** They are essentially the mobile equivalent of computer video games, offering a wide range of experiences from casual puzzle games to intense action-packed adventures. Candy crush Saga, PUBG Mobile.
- **Productivity apps:** These focus on improving business efficiency by easing various tasks such as sending emails, tracking work progress, booking hotels, and much more.
- ✓ **Lifestyle and entertainment apps:** Lifestyle and entertainment apps have become integral parts of our daily lives, offering various forms of entertainment, socialization, and personal expression. Here's a breakdown of what they encompass:
- ✓ **Social Media Apps:** These platforms enable users to connect with friends, family, and communities, sharing updates, photos, and videos. Examples include: Facebook, Instagram, Twitter, Snapchat, and TikTok.
- ✓ **Video Streaming Apps:** These apps provide access to a vast library of movies, TV shows, and original content for on-demand viewing. Examples include: Netflix, Amazon Prime Video, Disney+, and YouTube.
- ✓ **Music Streaming Apps:** These platforms offer a wide range of music content, including songs, playlists, albums, and podcasts, for streaming and offline listening. Examples include: Spotify, Apple Music ,Pandora , YouTube Music
- ✓ **Fitness and Wellness Apps:** These apps help users maintain a healthy lifestyle by providing workout routines, meditation guides, nutrition plans,

and activity tracking features. Examples include: Nike Training Club,MyFitnessPal, Headspace, and Strava.

Use of Mobile technology

- The incorporation of mobile technology into business has aided telecollaboration. Now, people could connect from anywhere using mobile technology, and access the papers and documents they need to complete collaborative work.
- Work is being redefined by mobile technologies. Employees are no longer confined to their desks; they can work from anywhere in the world.
- Mobile technology can help your company save time and money. Employees who work from home save thousands on a regular basis. Mobile phones eliminate the need for costly technology like landline carrier services. Cloud-based services are less expensive than traditional systems. Technology can also help your company become more flexible and productive.
- Mobile technology has the potential to boost productivity significantly. Mobile application integration saves an average of 7.5 hours per week per employee. Workers can also become more productive with the use of smartphones and mobile gadgets.
- The popularity of cloud-based services has skyrocketed in recent years. Cloud-based mobile technology applications have been seen to be more useful than any smartphone, particularly in terms of available storage space.

Advantages of Mobile technology

- **Greater reach and visibility**
  - Mobile apps can be downloaded from app stores like the App Store or Google Play, meaning you can reach a wider audience around the world. Additionally, apps can be easily shared by users, further increasing their reach and visibility.
- **Better user experience**
  - Mobile apps are typically designed specifically for mobile devices and offer an optimized user experience compared to mobile websites. Apps typically have an intuitive user interface and faster navigation, which can improve user satisfaction.
- **Take advantage of the device's features**

- Mobile applications can take advantage of the features of mobile devices such as camera, GPS, microphone, accelerometer, etc. This allows applications to be more interactive and personalized for users.

- **Improve customer loyalty**
  - Mobile apps allow businesses to connect directly with their customers and offer a personalized and relevant experience. Push notifications can be used to send important information and promotions to users, which can help improve customer loyalty.
- **Generate income**
  - Mobile apps can generate revenue in a variety of ways, such as by selling products or services within the app, by advertising, or by subscribing users. Apps can also help reduce marketing and advertising costs by promoting the brand and products effectively.

Disadvantages of Mobile technology

- **Higher development cost**
  - Mobile app development can be more expensive than web app development, as specific versions are required to be created for different operating systems (iOS and Android) and devices.
- **Longer development time**
  - The development process for a mobile app can be longer than that of a web app due to the need to create multiple versions for different operating systems and devices, which can delay time to market.
- **Lower audience reaches**
  - Mobile apps are limited by the number of devices they can be downloaded on, while web apps can be accessible from any device with an Internet connection. Additionally, mobile applications may require users to have a stable Internet connection and a sufficient amount of available storage on their devices to download the application.
- **Updates and maintenance**
  - Mobile apps require regular updates to ensure they work correctly on users' devices, and this can be expensive and time-consuming. Additionally, additional testing may be required to ensure that the app works correctly on different operating systems and devices.

Android:

Introduction to Android:

When we talked about operating systems few years ago, the most common answers were Windows, Linux, and macOS operating system. However, with the undying competition in the mobile phones market, the next big thing entered was ANDROID, which in no time became the heart of smart phones. Android provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment.

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touch screen mobile devices such as smart phones and tablets. The Android Operating System is a Linux-based OS developed by the Open Handset Alliance (OHA). The Android OS was originally created by Android, Inc., which was bought by Google in 2005.

Open Handset Alliance - It's a consortium of 84 companies such as Google, Samsung, AKM, synaptics, KDDI, Garmin, Teleca, eBay, Intel etc. It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Platform.

The android is a powerful operating system and it supports large number of applications in Smart phones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open source operating system means that it’s free and any one can use it.

In addition, Google has further developed Android TV for televisions, Android Auto for cars and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

The android has got millions of apps available that can help you managing your life one or other way and it is available low cost in market at that reasons android is very popular.

The main advantage to adopting Android is that it offers a unified approach to application development. Developers need only develop for Android in general, and their

applications should be able to run on numerous different devices, as long as the devices are powered using Android.

Android Features:

The important features of android are given below:

1) It is open-source.

2) Anyone can customize the Android Platform.

3) There are a lot of mobile applications that can be chosen by the consumer.

4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

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

Android versions:

- Android has gone through quite a number of updates since its first release. Table 1-1 shows the various versions of Android and their codenames.

Table 1: A Brief history of Android Versions

Android Version	Release Date	Code Name
1.1	9 February 2009	**"Astro Boy"** and **"Bender"**
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 October 2009	Éclair
2.2	20 May 2010	Froyo
2.3	6 December 2010	Gingerbread
3.0	22 February 2011	Honeycomb
4.0	18 October 2011	Ice Cream sandwich
4.1 to 4.3	09 July 2012	Jelly Bean
4.4	31 October 2013	Kit Kat
5.0	12 November 2014	Lollipop
6.0	05 October 2015	Marshmallow
7.0	22 August 2016	Nougat
8.0	December 5, 2017	Oreo
9.0	August 6, 2018	Pie
10	September 3, 2019	Queen Cake
11	September 8, 2020	Red Velvet Cake
12	October 4, 2021	Snow Cone

Android version 1.5: Cup Cake

On April 2009, the Android 1.5 update, Cup Cake was released based on linux kernel 2.6/2.7. It has Virtual Keyboards, Video Recording, Auto –Pairing, & Stereo Support for Bluetooth, Use Pictures shown for favorites in Contacts & Few more.

Android version 1.6: Donut

In 2009, Google released the Android operating system version 1.5, which was internally named "Cupcake" and introduced significant updates and features to the platform. Following Cupcake, Google continued its development, and later in 2009, they released Android version 1.6, which was internally nicknamed "Donut. "It has improved Functionality (quick search box, Updated Camera, Gallery and voice search). Battery Usage Indicator, and support for Super-Sharp 480X800 pixel Screens.

Android version 2.0: Éclair

Android version 2.0, codenamed Éclair, was a major update to the Android operating system released in October 2009. It gave us improved typing speed on virtual Keyboard By using Multi touch Data, Auto Brightness, Improved Google Maps 3.1.2 with Navigation, Numerous New Camera features, including flash, Color effects, Macro focus, Picture Size, storage location and much more updates.

Android version 2.2: Froyo

Android 2.2, also known as "**Froyo**" (short for Frozen Yogurt), did indeed introduce significant improvements in voice control and search capabilities, allowing users to control their phones without touching them. You could also now installed apps on memory Card. For instance, they could send text messages, make calls, play music, search the web, or get directions by speaking commands to their device. Prior to this update, Android devices only allowed apps to be installed on the device's internal storage. In Froyo users gained the capability to install applications onto the external storage, typically referred to as the memory card.

Android version 2.3: Gingerbread

In these update, Android updates with user Interface design with increased simplicity and speed, support for multiple cameras on the device, new download manager, new audio effects and Native support for SIP VOIP internet telephony. With native SIP VOIP support, Android users gained the ability to make voice calls over Wi-Fi or mobile data networks, potentially reducing call costs and providing more flexibility in communication.

Android version 3.0: Honeycomb

Honeycomb stands out in Android history as the only version developed specifically for tablets. Interface elements like the virtual keyboard were optimized for bigger screens and you had support for multi-core processors, including the virtual

keyboard, were designed to work better on devices with larger displays, providing a more comfortable and efficient user experience. Multi-core processors allow devices to execute multiple tasks simultaneously, leading to improved performance and multitasking capabilities.

Android version 4.0: Ice Cream Sandwich

An Ice Cream Sandwich hit phones in 2011, bringing an all-new look and feel to Android. You could also now close apps with a quick swipe, shoot 1080p video and providing users with higher-quality video recording capabilities, unlock your phone with your face(facial recognition technology that allowed users to unlock their phones using their faces), where supported. Android introduced the ability to close apps quickly by swiping them away from the multitasking or recent apps menu.

Android version 4.1 to 4.3 Jelly bean

Android Jelly Bean made Google's OS more responsive than ever, improving search functionality and introducing the ability to share files with your friends using Android. Android Jelly Bean, a feature that allowed users to share content such as web pages, contacts, photos, videos, and more by simply tapping their devices together.

Android version 4.4: Kit Kat

It was launched on October 31, 2013. It was designed to require less RAM than before, in order to help its performance on phones with less than 1 Gigabyte of RAM.

Android version 5.0: Lollipop

Android Lollipop hit our phones in 2014 and brought multiple profiles on one device, the 'no interruptions' feature to get some peace and an all-new notifications bar. You could also now unlock your phone with a trusted Bluetooth device.

Ex: Family Sharing: A parent can set up separate user profiles for themselves and their children on a tablet. Each child can have their own profile with age-appropriate apps and settings, while the parent can have a profile with full access to all apps and settings.

Android version 6.0: Marshmallow

Android marshmallow was unveiled by Google in September 2015, improving battery life and adding Cool new features like now on tap and fingerprint sensor support.

Android version 7.0: Nougat

Nougat is the latest update of Android having multiple features like quick switch between apps, Multi – window view, multi locale language settings, over 1500 emoji including 72 new ones, Virtual Reality mode and much more.

Android version 8.0: Oreo

Android Oreo also introduces two major platform features: Android Go – a software distribution of the operating system for low-end devices – and support for implementing a hardware abstraction layer. It supports Notification grouping, Picture-in-picture support for video, Performance improvements and battery usage optimization, Support for auto fillers, Bluetooth 5, System-level integration with VoIP apps.

Android version 9.0: Pie

It was first announced by Google on March 7, 2018, but was released on July 25, 2018. Pie's most transformative change was its hybrid gesture/button navigation system, which traded Android's traditional "Back", "Home", and "Overview" keys for a large, multifunctional Home button and a small Back button that appeared alongside it as needed. It supports the following features:

- A "screenshot" button has been added to the power options.
- Richer messaging notifications, where a full conversation can be had within a notification, full-scale images.
- New user interface for the quick settings menu.
- An Adaptive Battery feature that maximizes battery power by prioritizing the apps you're most likely to use next.
- Improved Adaptive Brightness feature which modifies screen brightness based on your own personal preferences.
- New Back Button Icon in the navigation bar.

Android version 10: Queen Cake

Android 10 is the 10th major release and seventeenth version of Android, released on September 3, 2019. Most noticeably, the first Android version to shed its letter and be

known simply by a number, with no dessert theme. However, there was a code name for Android 10 named "Queen Cake".

It supports the following features:

- Gesture Interface added in this release.
- Introduced swipe-driven approach to system navigation.
- Update the device tracking functions.
- Introduced dark theme.
- Introduces a new Live Caption feature that allows you to generate on-the-fly visual captions for any media playing on your phone

Android version 11: Red Velvet Cake

The version's most significant changes revolve around privacy. The update builds upon the expanded permissions system introduced in Android 10 and adds the ability for users to grant apps certain permissions – those related to location access, camera access, and microphone access – only on a limited, single-use basis. Although this version was marketed as Android 11 without a name for the desert, there was a code name for Android 11 which is named "Red Velvet Cake".

Features of Android 11.0:

- This version introduced the background location permission even deeper into the system and made it more difficult for apps to request.
- Android 11 removes an app's ability to see what other apps are installed on your phone.
- It refines the system notification area to emphasize and simplify conversation-related alerts.
- It introduces a new streamlined media player that contains controls for all audio and video-playing apps in a single space

Android version 12: Snow Cone

Android 12 is the first software version to integrate an updated and completely overhauled take on that standard – something known as Material You. Material you brings a dramatically different look and feel to the entire Android experience, and it isn't limited only to system-level elements, either. Eventually, Android 12's design principles will stretch into both apps on your phone and Google services on the web.

It supports the following features:

- Material you and wallpaper-based theming (Accent colours picked from your wallpaper - optional).
- New Extra Dim Mode (Reduces screen brightness).
- Unified the WIFI and Mobile Data quick settings to bring up a mini-settings panel with both options.
- Scrolling Screenshots in supported apps.
- Privacy Dashboard.
- Hold the power button for the assistant gestures.
- Adaptive Charging added.
- Giving access to an approximate location instead of an accurate location.
- Universal device search

Features of Android:

Because Android is open source and freely available to manufacturers for customization, there are no fixed hardware or software configurations. However, Android itself supports the following features:

1. **Storage** — Uses SQLite, a lightweight relational database, for data storage.

2. **Connectivity** — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), Wi-Fi, LTE, and WiMAX.

3. **Messaging** — Supports both SMS and MMS.

4. **Web browser** — based on the open source WebKit, together with Chrome's V8 JavaScript engine.

5. **Media support** — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.

6. **Hardware support** — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS

7. **Multi-touch** — Supports multi-touch screens

8. **Multi-tasking** — Supports multi-tasking applications

9. **Flash support** — Android 2.3 supports Flash 10.1. (it could play videos, games, and animations on websites that were made using something called Flash. It's like a special tool websites used for adding moving stuff. Flash became less popular because of some problems it

had, so newer versions of Android and most web browsers stopped supporting it. Now, websites use different ways to make things move, so you don't see Flash as much anymore.)

10. **Tethering** — Supports sharing of Internet connections as a wired/wireless hotspot.

Android Architecture:

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

The main components of android architecture are following:-

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

1. Linux kernel — This layer contains all the low level device drivers for the various hardware components of an Android device. At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

2. Libraries — These contain all the code that provides the main features of an Android OS. On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc. The WebKit library provides functionalities for web browsing.

3. Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every

Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU power.

4. Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications. The Application Framework layer provides many higher-level level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android framework includes the following key services.

Activity Manager − Controls all aspects of the application lifecycle and activity stack.

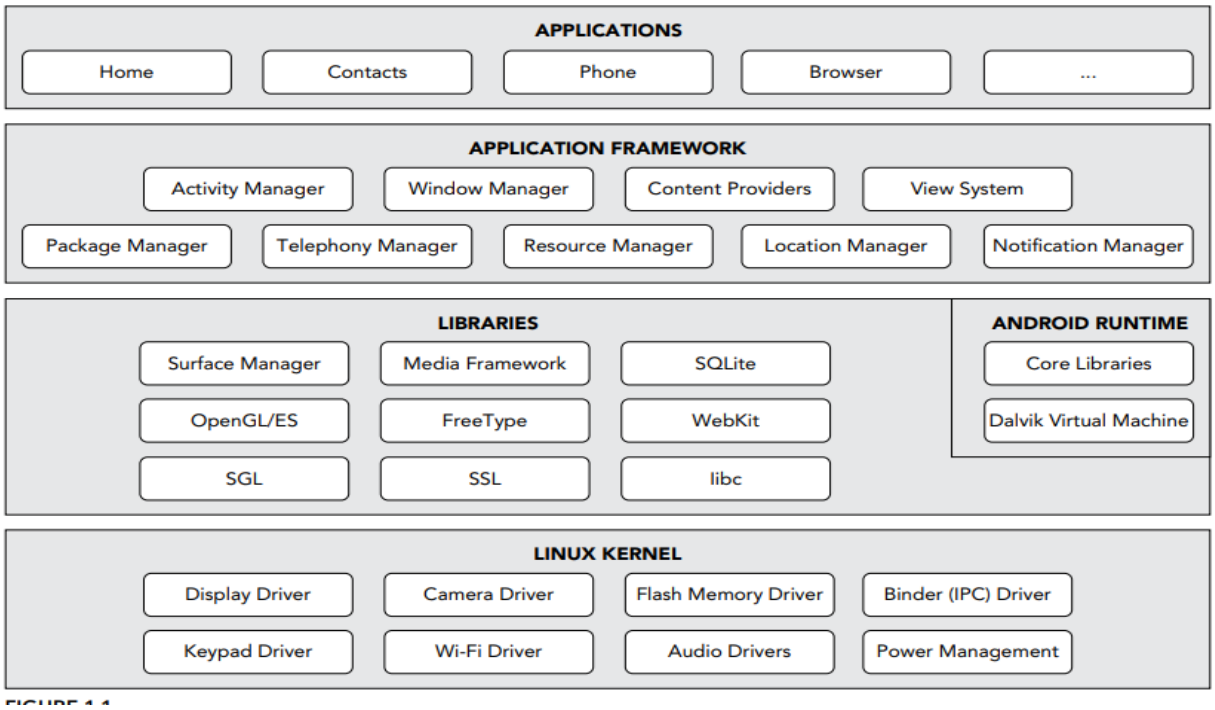
Content Providers − Allows applications to publish and share data with other applications.

Resource Manager − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.

Notifications Manager − Allows applications to display alerts and notifications to the user.

View System − An extensible set of views used to create application user interfaces.

5. Applications - You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.



Android - Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application –

Sl.No	Components & Description
1	**Activities** They dictate the UI and handle the user interaction to the smart phone screen.
2	**Services** They handle background processing associated with an application.
3	**Broadcast Receivers** They handle communication between Android OS and applications.
4	**Content Providers** They handle data and database management issues.

1. **Activities**

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. Activity is the main entry point of an Android. It consist of 2 layers. In ***frontend it uses xml*** and in ***Backend it uses java/kotlin.***

For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

Ex: Gmail Application

- ✓ Open Gmail App
- ✓ Compose email
- ✓ Send Email
- ✓ Display list of emails
- ✓ Open sent items

An activity is implemented as a subclass of Activity class as follows –

```
public class MainActivity extends Activity {
}
```

2. **Services**

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

(Services are like background workers. They do tasks in the background without needing a user interface. For instance, a music player app might have a service that keeps playing music even when you switch to another app.)

- Ex: Alarm, Media Player

A service is implemented as a subclass of Service class as follows –

```
public class MyService extends Service {
}
```

3. Broadcast Receivers

- Broadcast Receivers are one of the major component of the android. Broadcast Receivers simply respond to broadcast messages from other applications or from the system.
- A broadcast receiver (receiver) is an Android component which allows you to register for system or application events.
- For example, to deliver other apps know that some data downloaded to the device and available for use. Even though broadcast receivers do not show user interface, they might create status bar notification to user for events.
- Broadcast receiver is generally implemented to delegate the tasks to services depending on the type of intent data that's received.

Example:

1. Charger Connected/Disconnected
2. Screen ON/OFF
3. Headset
4. Battery Low
5. Wi-fi Network available

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
public class MyReceiver extends BroadcastReceiver {
        public void onReceive (context, intent){}
}
```

4. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

OR

- Content Provider component is providing data from one application to others on request.
- These requests are handled by the "ContentResolver" class methods.
- A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.
- Content providers let you centralize content in one place and have many different application access it as needed
- A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete() and query() methods.
- In most cases this data is stored in an SQlite database.
- A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider {
        public void onCreate(){}
}
```

5. Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

S.No	Components & Description
1	**Fragments** Represents a portion of user interface in an Activity.
2	**Views** UI elements that are drawn on-screen including buttons, lists forms etc.
3	**Layouts** View hierarchies that control screen format and appearance of the views.
4	**Intents** Messages wiring components together.
5	**Resources** External elements, such as strings, constants and drawable pictures.
6	**Manifest** Configuration file for the application.

Exploring the Development Environment:

For developing application for android platform, you will require Integrated Development Environment (IDE). Android Studio is the official IDE for Android application development. Android Studio provides everything you need to start developing apps for Android, including the Android Studio IDE and the Android SDK tool.

1. **Android studio**
2. **Eclipse IDE**

Exploring the development environment in Android is crucial for anyone looking to build Android applications. Here's a breakdown of the essential components and steps involved:

Java or Kotlin: Android apps are primarily developed using either Java or Kotlin programming languages. Kotlin is the newer, preferred language for Android development due to its conciseness, safety features, and interoperability with Java.

Android Studio: This is the official Integrated Development Environment (IDE) for Android development, provided by Google. Android Studio offers a comprehensive suite of tools for designing, coding, testing, and debugging Android applications. It includes features like a visual layout editor, APK analyzer, and built-in emulator.

SDK Manager: The Software Development Kit (SDK) Manager is a tool within Android Studio that allows developers to download and manage the various SDK components

necessary for Android development. This includes platform tools, system images for emulators, and additional libraries.

Emulator: Android Studio includes a built-in emulator that allows developers to test their apps on virtual devices with different configurations (e.g., screen size, Android version). This is useful for debugging and ensuring compatibility across various devices.

Device for Testing: While emulators are handy, testing on real devices is essential to ensure your app performs correctly across different hardware configurations. You can connect physical devices to your development machine via USB for testing.

Debugger: Android Studio provides a powerful debugger that allows developers to step through their code, set breakpoints, inspect variables, and analyze the runtime behavior of their applications. This is invaluable for identifying and fixing bugs.

Version Control: Version control systems like Git are commonly used in Android development to track changes to the codebase, collaborate with other developers, and manage different versions of the app.

Documentation and Community Resources: The Android developer website provides extensive documentation, tutorials, and API references to help developers learn and understand various aspects of Android development. Additionally, online forums like Stack Overflow and Reddit's r/androiddev community are valuable resources for seeking help and sharing knowledge with fellow developers.

Obtaining the Required Tools:

The first and most important piece of software you need to download is Android Studio After you have downloaded and installed Android Studio you can use the SDK Manager to download and install multiple versions of the Android SDK. Having multiple versions of the SDK available enables you to write programs that target different devices.

Android Studio:

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug. Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the

IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. To support application development within the Android operating system, Android Studio uses a Gradlebased build system, emulator, code templates, and GitHub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Android SDK:

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. SDK provides a selection of tools required to build Android apps or to ensure the process goes as smoothly as possible. The Android SDK comprises all the tools necessary to code programs from scratch and even test them. These tools provide a smooth flow of the development process from developing and debugging, through to packaging. The Android SDK is compatible with Windows, macOS, and Linux, so you can develop on any of those platforms.

1. SDK tools

SDK tools are generally platform independent and are required no matter which android platform you are working on. When you install the Android SDK into your system, these tools get automatically installed. The list of SDK tools has been given below –

SL.NO	Tools & Description
1	**android** This tool lets you manage AVDs, projects, and the installed components of the SDK
2	**ddms** (Dalvik debug monitor server) This tool lets you debug Android applications
3	**Draw 9-Patch** This tool allows you to easily create a Nine Patch graphic using a WYSIWYG editor
4	**emulator** This tools let you test your applications without using a physical device
5	**mksdcard** Helps you create a disk image (external sdcard storage) that you can use with the emulator
6	**proguard** Shrinks, optimizes, and obfuscates your code by removing unused code
7	**sqlite3** Lets you access the SQLite data files created and used by Android applications
8	**Traceview**

	Provides a graphical viewer for execution logs saved by your application
9	**Adb** Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device.

DDMS

- DDMS stands for Dalvik debug monitor server that provides many services on the device. The service could include message formation, call spoofing, capturing screenshot, exploring internal threads and file systems etc.
- Running DDMS from Android studio click on Tools>Android>Android device Monitor.

Sqlite3

- Sqlite3 is a command line program which is used to manage the SQLite databases created by Android applications. The tool also allows us to execute the SQL statements on the fly.
- There are two ways through which you can use SQLite, either from remote shell or you can use locally.

Android Emulator: -

- The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.
- The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
- Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.
- The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

- In short, An Android emulator is an Android Virtual Device (AVD) that represents a specific Android device. You can use an Android emulator as a target platform to run and test your Android applications on your PC. Using Android emulators is optional.

Android Virtual Device Manager: - An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The AVD Manager is interfaces you can launch from Android Studio that helps you create and manage AVDs. An AVD contains a hardware profile, system image, storage area, skin, and other properties.

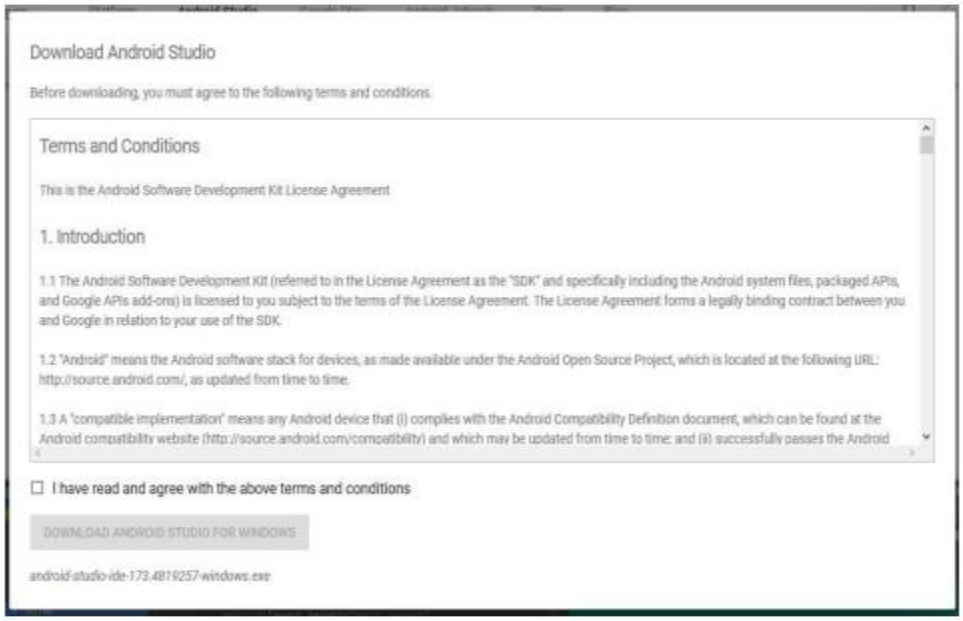
Use the following steps to go through the installation process of Android Studio:

Step – 1: Head over to bellow link to get the Android Studio executable or zip file . https://developer.android.com/studio/#downloads

Step – 2: Click on the download android studio button

.

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



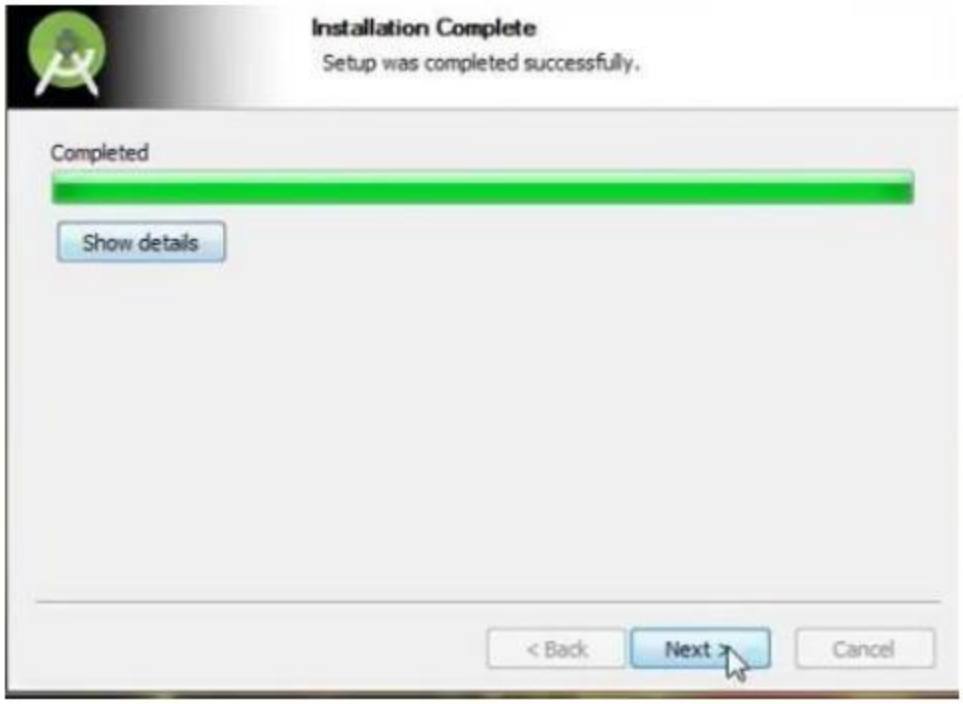
Click on save file button in the appeared prompt box and the file will start downloading.

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

Click on next.

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

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



Click Next

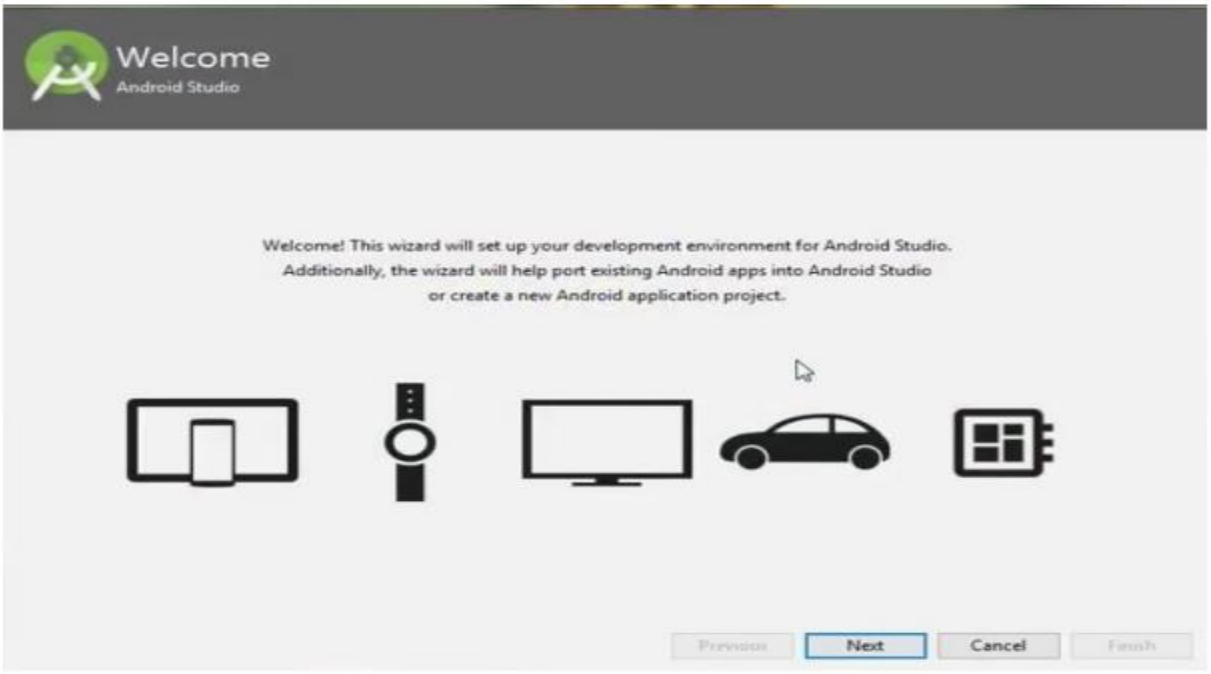
Step – 5: Once "Finish" is clicked, it will ask whether the previous settings needs to be imported [if android studio had been installed earlier], or not.

It is better to choose the 'Don't import Settings option'.

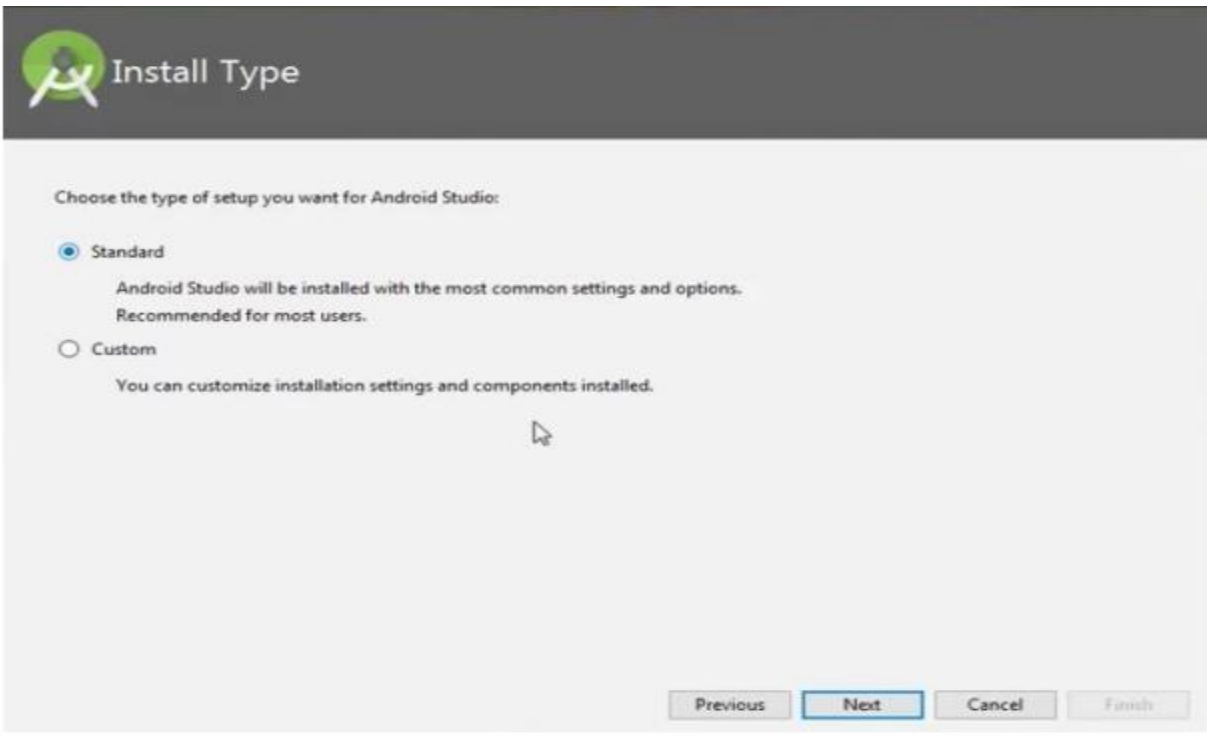
Step – 6 : This will start the Android Studio.

Meanwhile it will be finding the available SDK components.

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



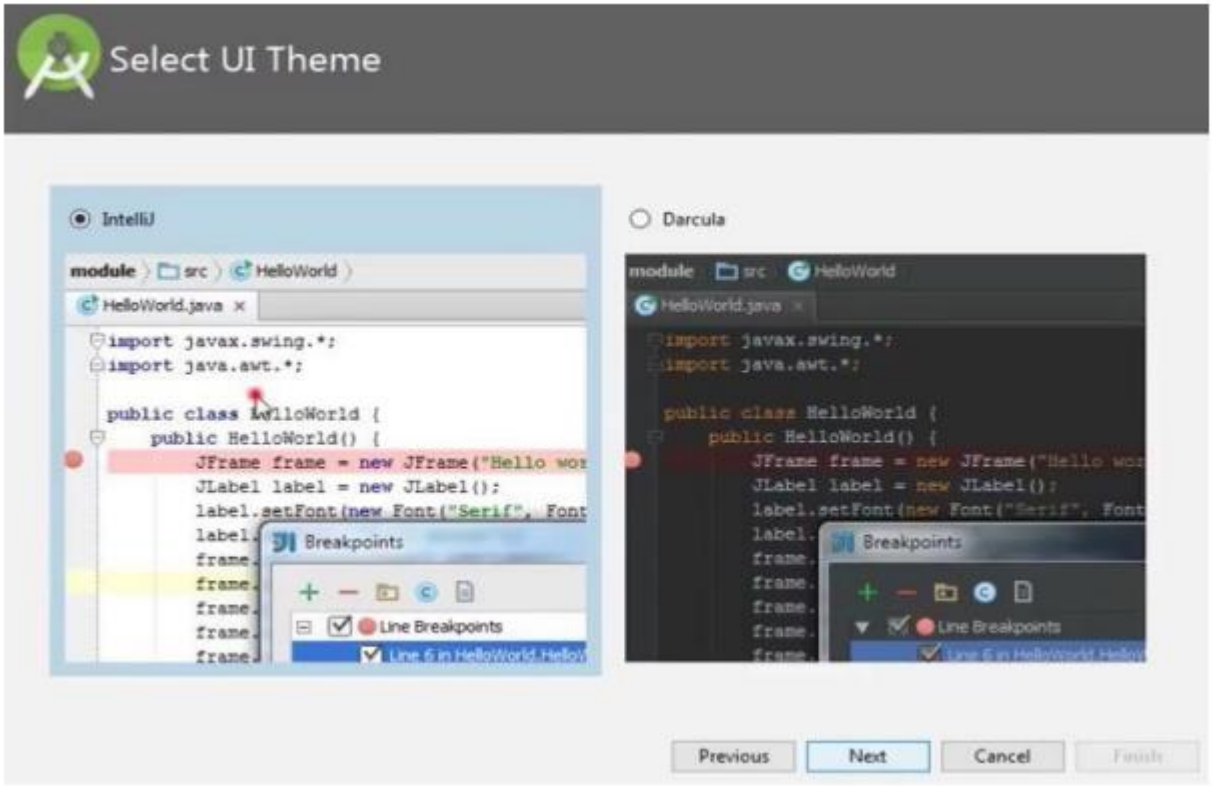
Click Next



Choose Standard and click on Next.

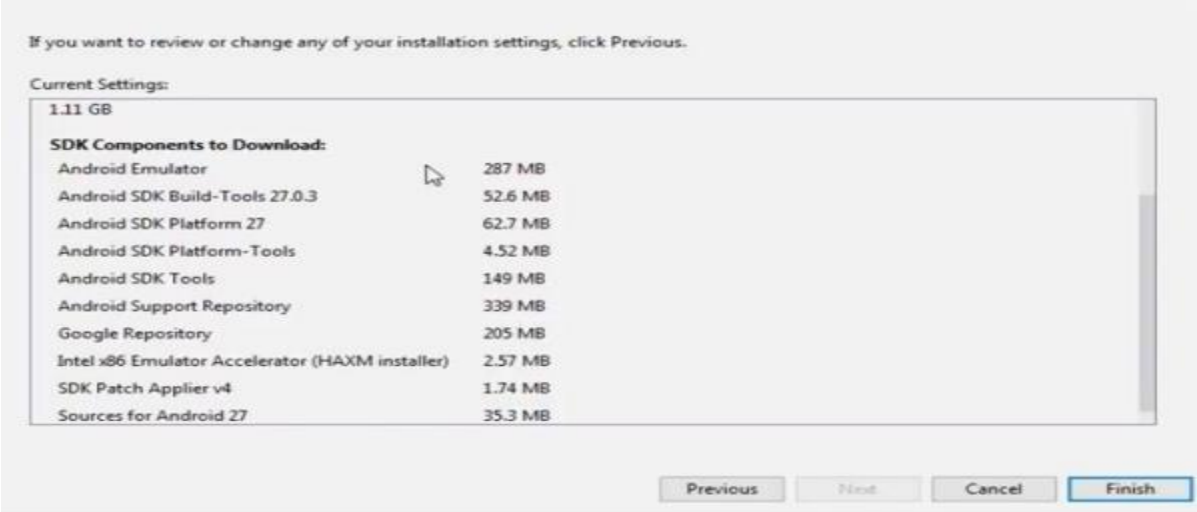
Now choose the theme, whether Light theme or the Dark one .

The light one is called the IntelliJ theme whereas the dark theme is called Darcula . Choose as required.



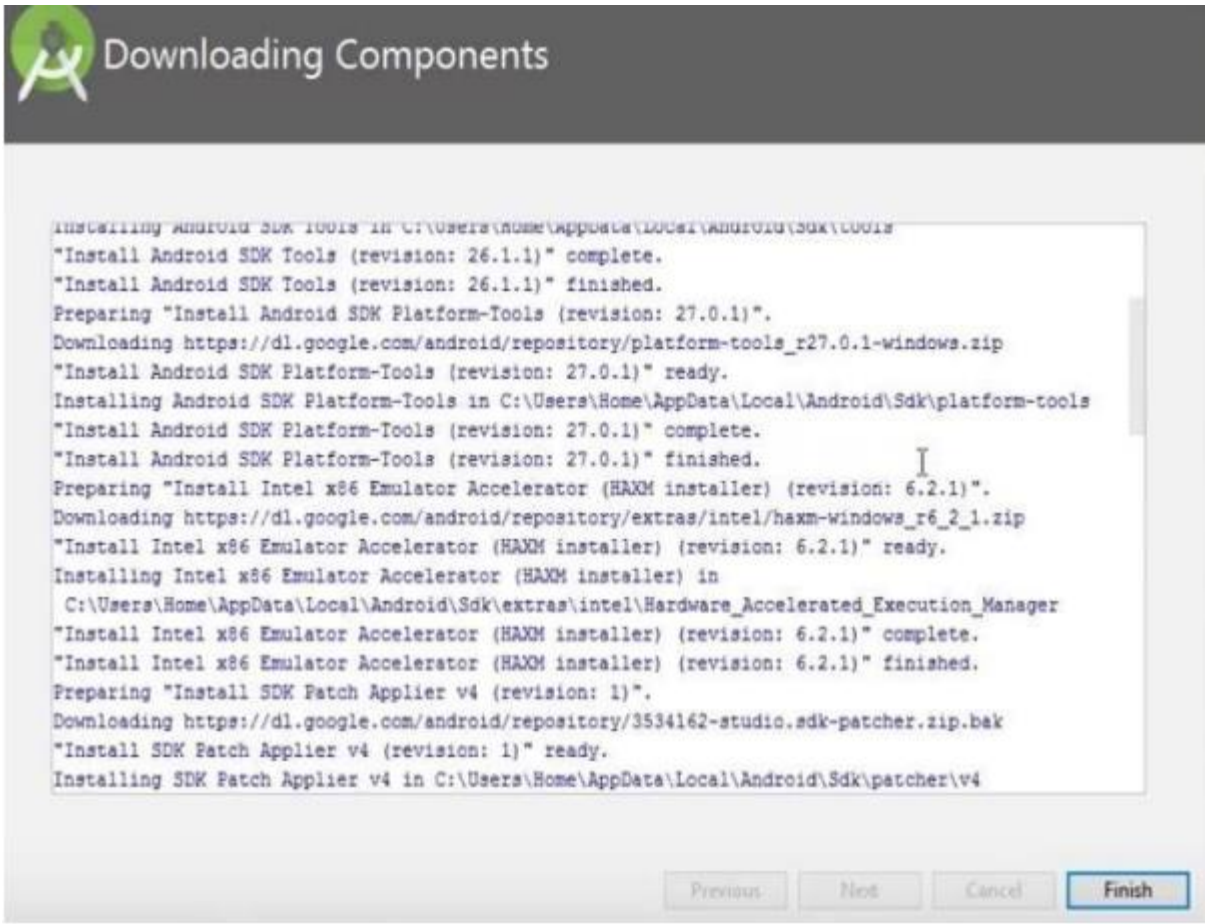
Click on the Next button

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



Click on Finish.

It has started downloading the components



The Android Studio has been successfully configured. Now it's time to launch and build app.

To create your new Android project, follow these steps:

• Install the latest version of Android Studio.

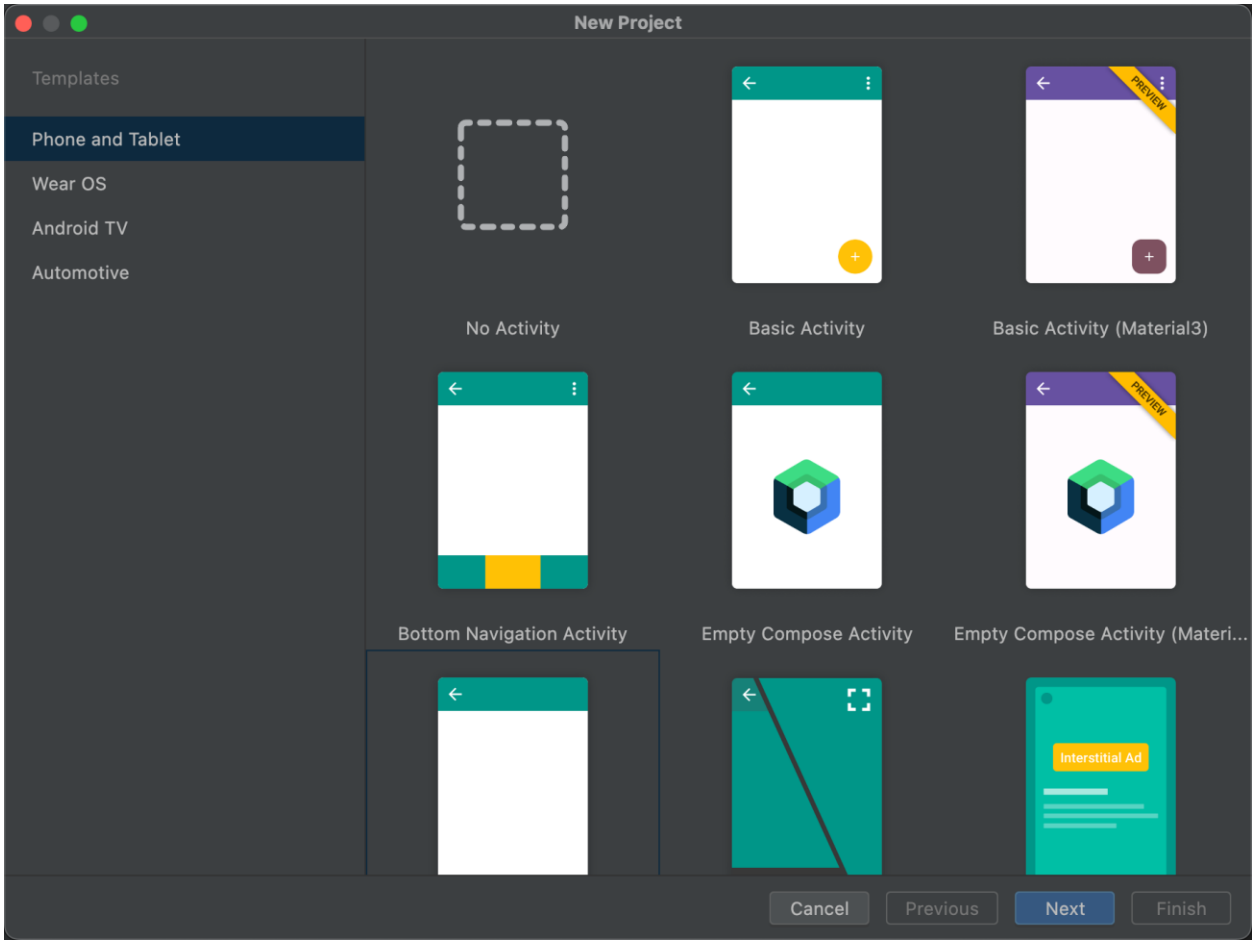
• In the Welcome to Android Studio window, click Start a new Android Studio project.

Creating a Project in Android and Launching Your First Android Application:

1. Create a new project by clicking **Start a new Android Studio project** on the Android Studio Welcome screen.
2. If you do have a project opened, create a new project by selecting **File > New > New Project** from the main menu.

Choose your project type

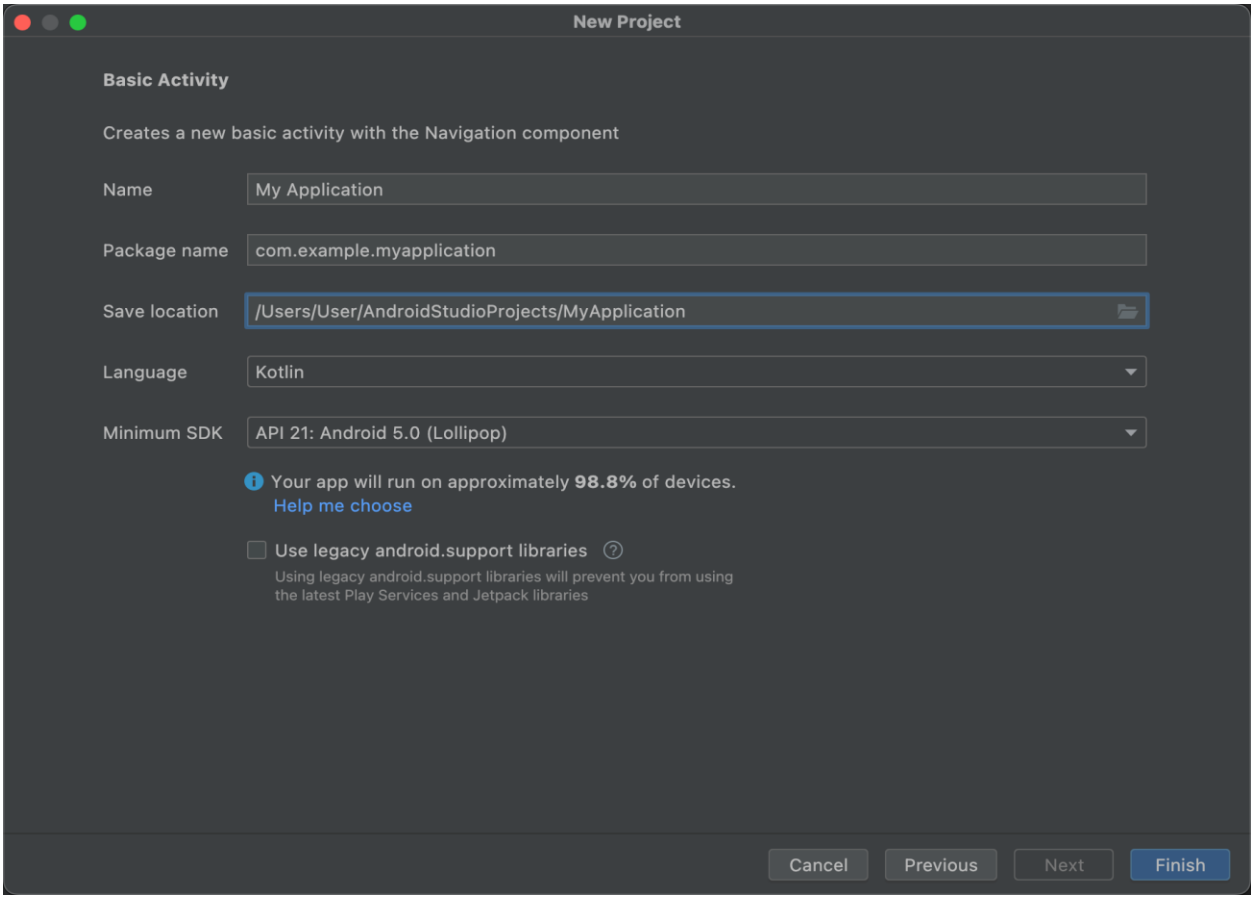
In the **New Project** screen that appears, you can select the type of project you want to create from categories of device form factors, shown in the **Templates** pane. For example, figure 1 shows the project templates for phone and tablet.



3. Selecting the type of project, you want to create lets Android Studio include sample code and resources in your project to help you get started. Once you select your project type, click **Next**.

Configure your project

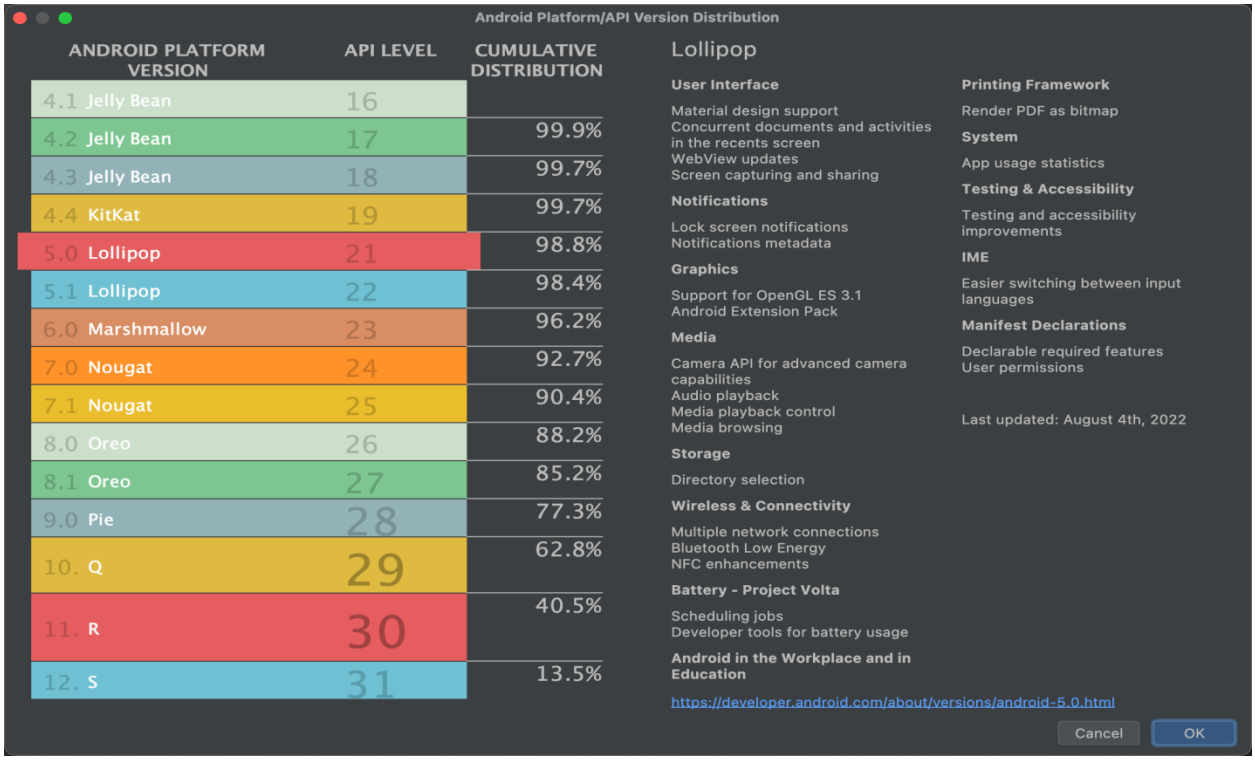
The next step in creating your project is to configure some settings, as shown in figure 2. If you're creating a **Native C++** project, read Create a new project with C/C++ support to learn more about the options you need to configure.



4. Specify the **Name** of your project. The **Name** field is used to enter the name of your project, for this codelab type "Greeting Card".
5. Specify the **Package name**. By default, this package name becomes your project's namespace (used to access your project resources) and your project's application ID (used as the ID for publishing). Leave the **Package name** field as is. This is how your files will be organized in the file structure. In this case, the package name will be `com.example.greetingcard`.
6. Specify the **Save location** where you want to locally store your project.
7. Select the **Language**, Kotlin or Java, you want Android Studio to use when creating sample code for your new project. Keep in mind that you aren't limited to using only that language in the project.
8. Select the **Minimum API level** you want your app to support. When you select a lower API level, your app can't use as many modern Android APIs. However, a larger percentage of Android devices can run your app. The opposite is true when

selecting a higher API level. Select API 24: Android 7.0 (Nougat) from the menu in the Minimum SDK field. Minimum SDK indicates the minimum version of Android that your app can run on.

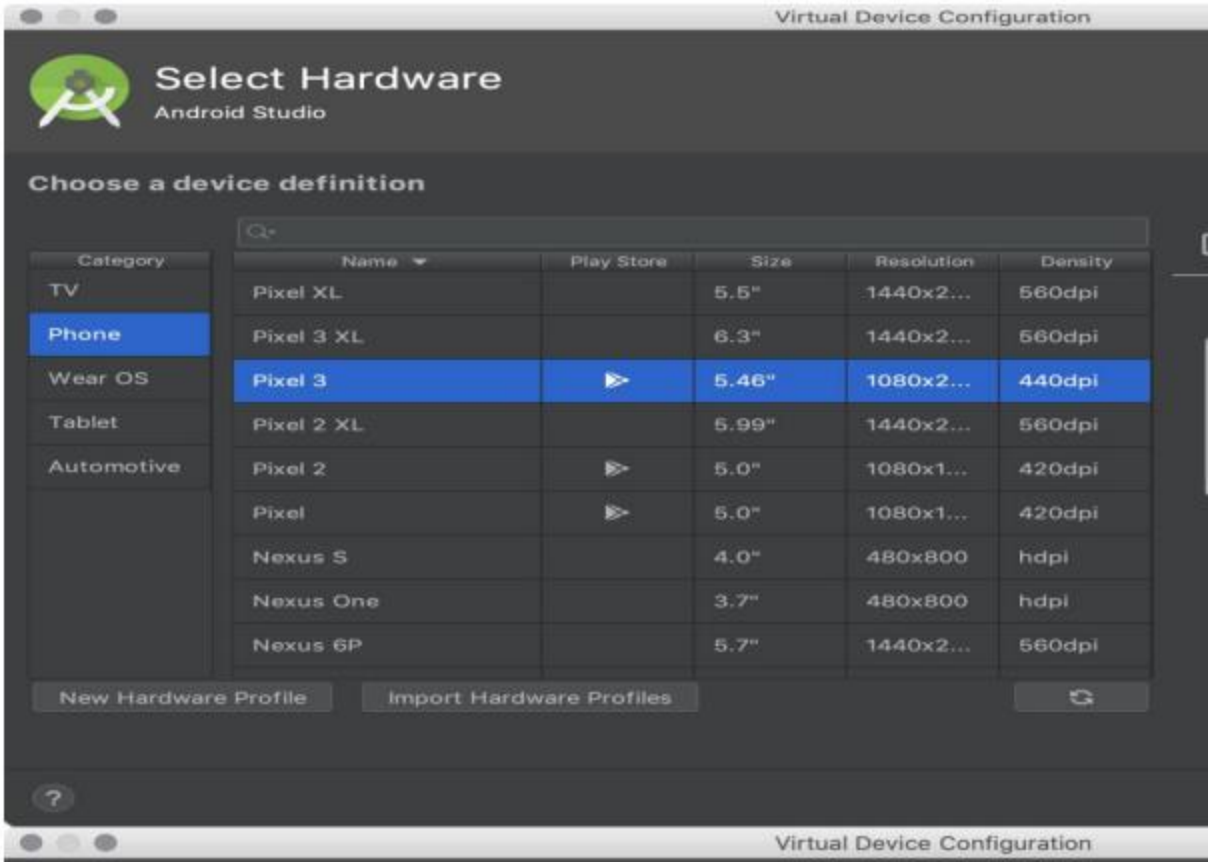
9. If you want to see more data to help you decide, click **Help me choose**. This displays a dialog showing the cumulative distribution for the API level you have selected and lets you see the impact of using different minimum API levels.

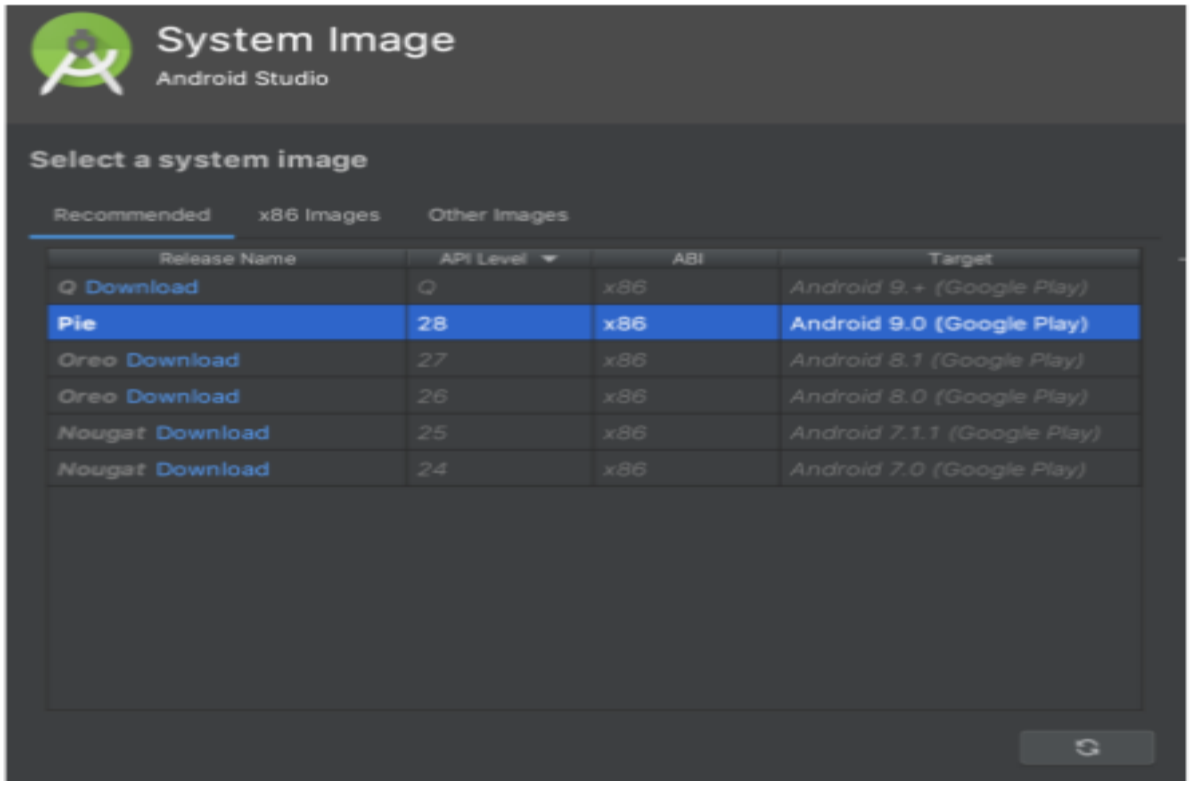


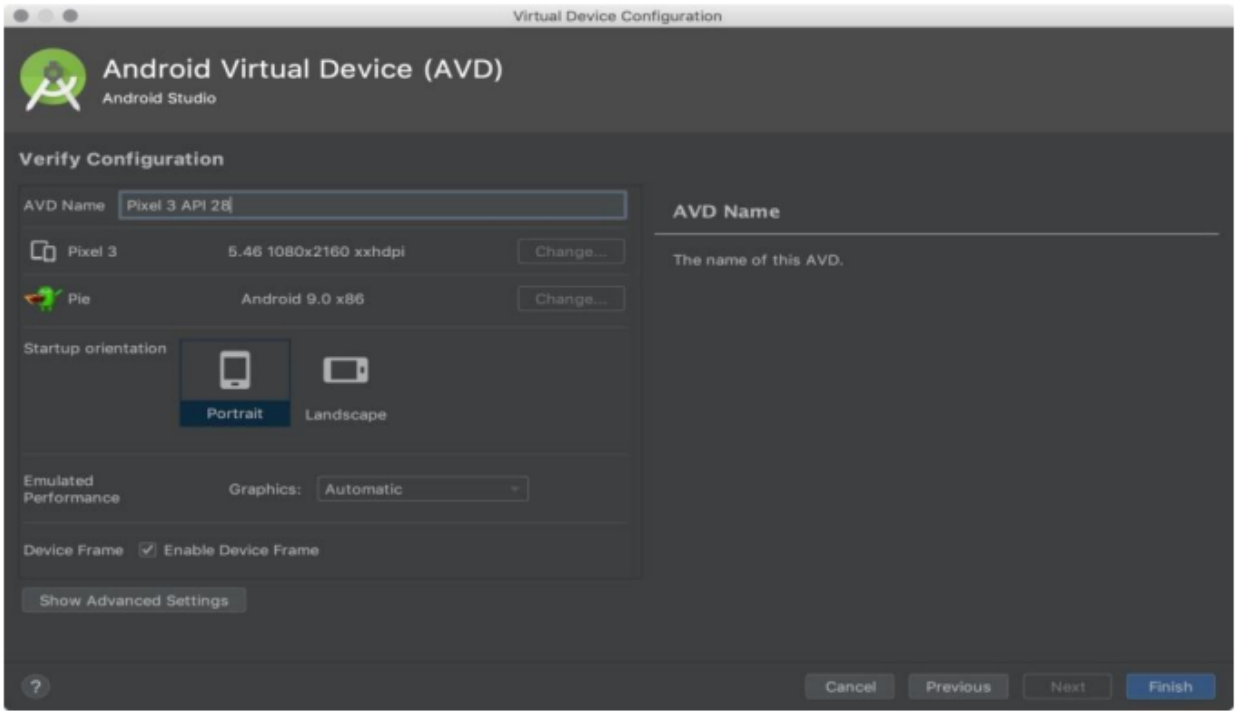
10. Your project is configured to use AndroidX libraries by default, which replace the Android Support libraries. To use the legacy support libraries instead, select **Use legacy android.support libraries**. However, this is not recommended, as the legacy support libraries are no longer supported. To learn more, read the AndroidX overview.
11. When you're ready to create your project, click **Finish**.
12. Android Studio creates your new project with some basic code and resources to get you started.
13. After some processing time, the Android Studio main window appears.

Creating AVD in Android Studio:

1. Open the AVD Manager by clicking Tools > AVD Manager.
2. Click Create Virtual Device, at the bottom of the AVD Manager dialog....
3. Select a hardware profile, and then click Next. Select the Nexus 5X API N (feel free to select the Nexus 5x API 18, which is the Jelly Bean emulator that you created in the Try It Out for the last section), and click Next.
4. Select the system image for a particular API level, and then click Next.
5. Change AVD properties as needed, and then click Finish.

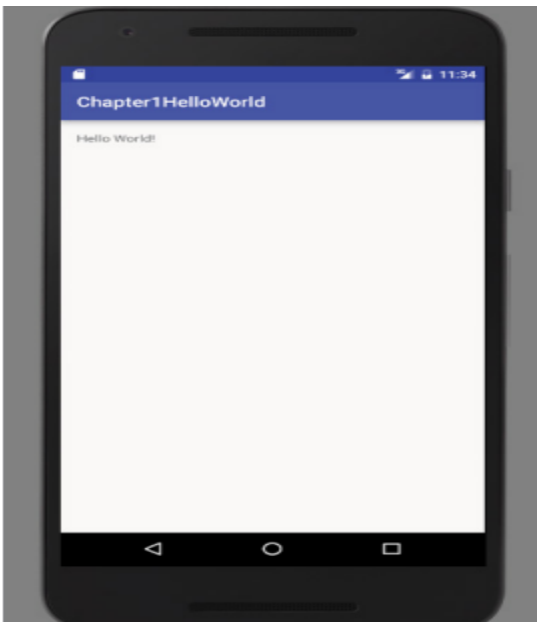

Virtual Device Configuration
Select Hardware
Android Studio
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Store
Size
Resolution
Density
Pixel XL 5.5" 1440x2... 560dpi
Pixel 3 XL 6.3" 1440x2... 560dpi
Pixel 3 5.46" 1080x2... 440dpi
Pixel 2 XL 5.99" 1440x2... 560dpi
Pixel 2 5.0" 1080x1... 420dpi
Pixel 5.0" 1080x1... 420dpi
Nexus S 4.0" 480x800 hdpi
Nexus One 3.7" 480x800 hdpi
Nexus 6P 5.7" 1440x2... 560dpi
New Hardware Profile
Import Hardware Profiles
Virtual Device Configuration


System Image
Android Studio
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
Q Download Q x86 Android 9.+ (Google Play)
Pie 28 x86 Android 9.0 (Google Play)
Oreo Download 27 x86 Android 8.1 (Google Play)
Oreo Download 26 x86 Android 8.0 (Google Play)
Nougat Download 25 x86 Android 7.1.1 (Google Play)
Nougat Download 24 x86 Android 7.0 (Google Play)



How to Run the Project:

1. After the project is created, there are 2 files, MainActivity.java and activity_main.xml
2. Go to activity_main.xml and select Design View
3. In Design View, change the layout to Linear Layout (Vertical) select Add Text View, and change the text to “Hello World!”
4. Click on Run and select the AVD already created (if not created, first create the AVD)
5. Output screen should show “Hello World”.



Debugging Your Application

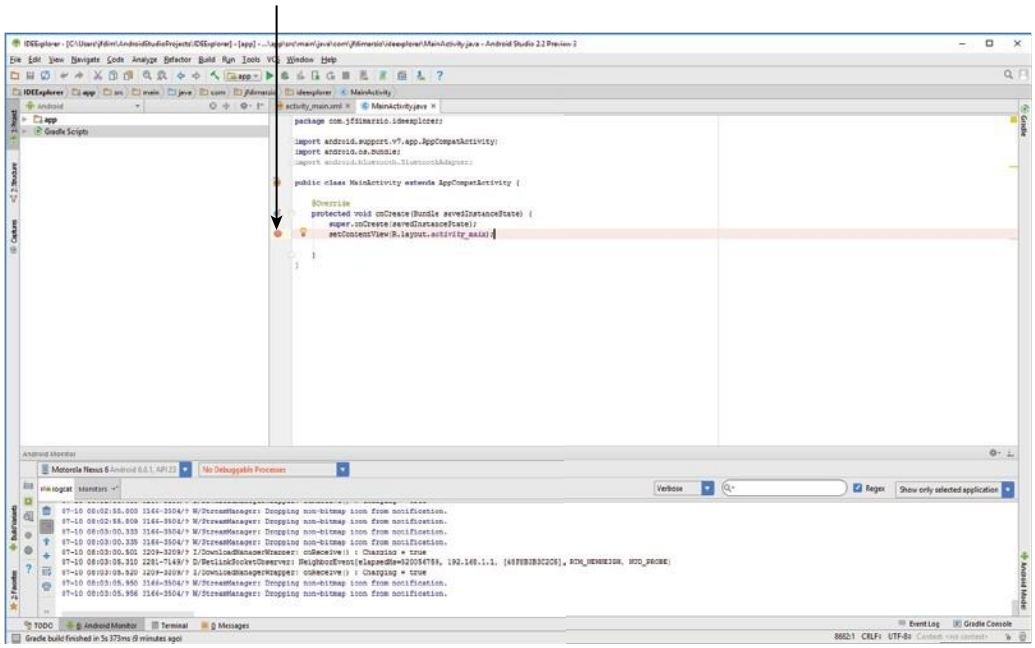
Debugging is the process of finding and fixing errors or bugs in the source code of any software.

- After you have built an application, you need to be able to debug it and see what is going on insideyour code.
- One of the handiest ways to be able to see inside your code it through the use of breakpoints.
- Breakpoints allow you to pause the execution of your code at specific locations and see whatis going on (or what is going wrong).

Setting Breakpoints

- Breakpoints are a mechanism by which you can tell Android Studio to temporarily pause execution of your code, which allows you to examine the condition of your application.
- This means that you can check on the values of variables in your application while you are debugging it. Also, you can check whether certain lines of code are being executed as expected—or at all.

- To tell Android Studio that you want to examine a specific line of code during debugging, you must set a breakpoint at that line.
- Click the margin of the editor tab next to line of code you want to break at, to set a breakpoint. A red circle is placed in the margin, and the corresponding line is highlighted in red.



- You can also set a breakpoint by placing your cursor in the line of code where you want it to break and clicking Run ⇨ Toggle Line Breakpoint. Notice that the term used is *toggle*, which means that any breakpoints you set can be turned off the same way you turn them on.

- Simply click an existing breakpoint to remove it from your code.

- *Android Studio only pauses execution at breakpoints when you debug* ***your application—not when you run it.***

Line breakpoint

The most common type is a line breakpoint that pauses the execution of your app at a specified line of code. The debugger suspends program execution once the execution reaches this line. While paused, you can examine variables, evaluate expressions, and then continue execution line by line to determine the causes of runtime errors.

To add a line breakpoint, proceed as follows:

1. Locate the line of code where you want to pause execution.

2. Click the left gutter along that line of code or place the caret on the line and press Control+F8 (on macOS, Command+F8).

3. If your app is already running, click **Attach debugger to Android process**  .

   Otherwise, to start debugging, click **Debug**  .

   A red dot appears next to the line when you set a breakpoint, as shown in figure 5.

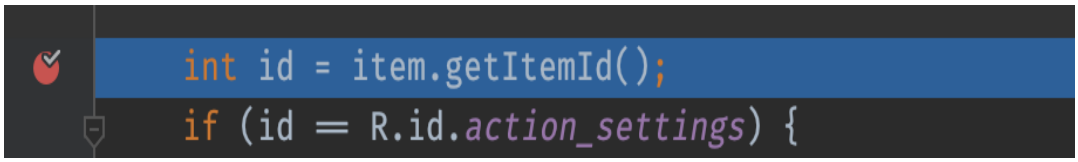


Figure 5. A red dot appears next to the line when you set a breakpoint.

When your code execution reaches the breakpoint, Android Studio pauses execution of your app.

METHOD Breakpoint

- Method breakpoints suspend the program each time it calls the specified method. A method breakpoint pauses the execution of your app when it enters or exits a specific method. While paused, you can examine variables, evaluate expressions, and then continue execution line by line to determine the causes of runtime errors.
- When dealing with composable functions, breakpoints become even more valuable as they provide detailed information about the function's parameters and their state to help identify what changes might have caused the recomposition.
- You can set a method breakpoint by selecting Run ⇨ Toggle Method Breakpoint. A method breakpoint is represented by a red circle containing four dots placed at the method signature.

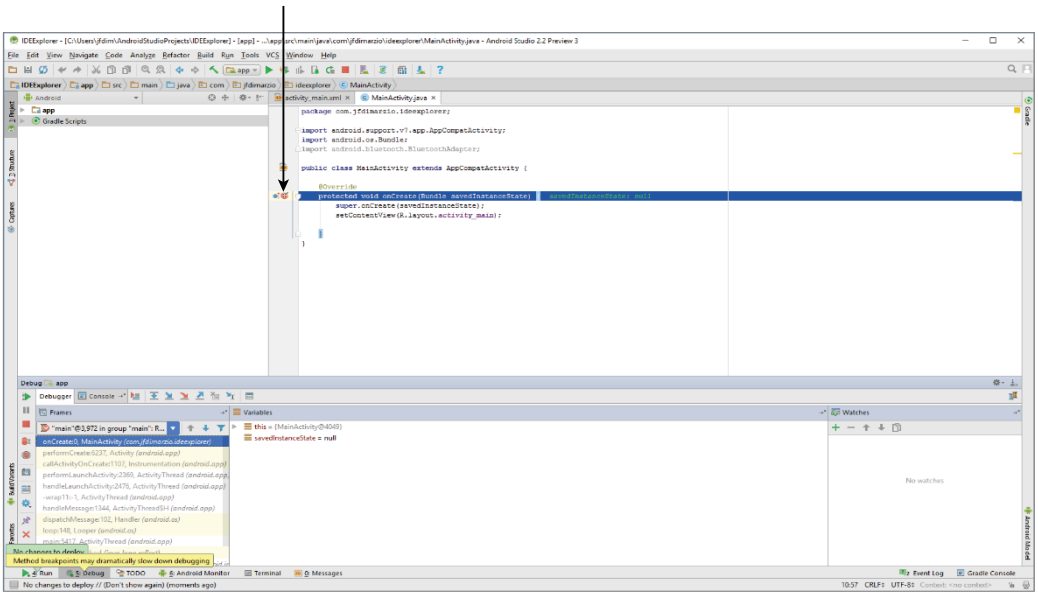


Figure 2-12

Temporary Breakpoints

- A temporary breakpoint is useful when you are trying to debug a large loop, or you just want to make sure a line of code is being hit during execution.
- To set a temporary breakpoint, place your cursor at the location in the code where you want it to break and select Run ⇨ Toggle Temporary Line Breakpoint.
- Notice that a red circle containing a 1 is now placed in the margin
- The 1 in the red circle represents the fact that Android Studio only stops at this breakpoint the first time your code enters it.
- After that, the line is executed as though there is no breakpoint set. This can be very useful if you want to ensure a line within a loop is being hit, but you don't want to stopat the line every time it is executed.

Conditional Breakpoints

- A condition breakpoint is a breakpoint at which Android Studio only pauses when specific conditionsare met.
- To set a conditional breakpoint, first set a simple breakpoint at the line of code youwant to examine, then right-click the simple breakpoint to bring up the condition context menu.
- From here you can set conditions that tell Android Studio when to pause at a breakpoint.
- For example, you can tell Android Studio to only pause at a line of code when your variable named foo equals true. You would then set the condition in the breakpoint to

foo == true

- Conditional breakpoints are extremely useful in diagnosing intermittent issues in complex codeblocks.

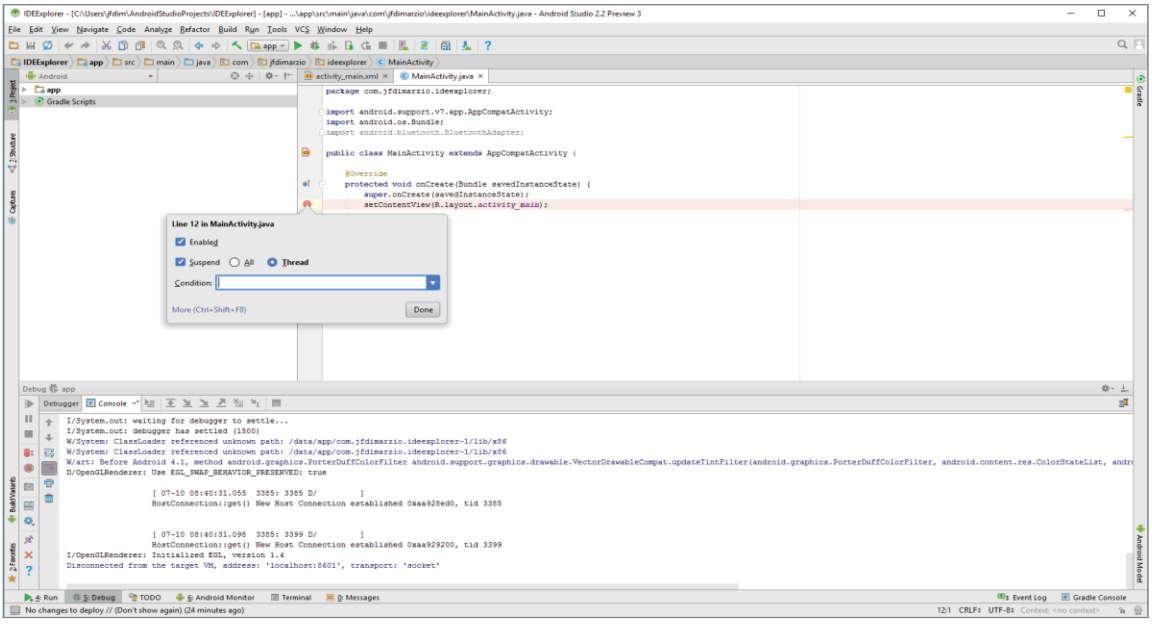


Figure 2-14

Field breakpoint

A field breakpoint pauses the execution of your app when it reads from or writes to a specific field.

Exception breakpoint

An exception breakpoint pauses the execution of your app when an exception is thrown.

Navigating Paused Code

- While in debug mode, Android Studio pauses at any breakpoint that you have set. That is, as long as a breakpoint has been set on a reachable line of code Android Studio halts execution at that line until you tell it to continue.
  - After your app's execution has stopped because a breakpoint has been reached, you can execute your code from that point one line at a time with the Step Over, Step Into, and Step Out functions.
  - To use any of the step functions:
- Begin debugging your app. Pause the execution of your app with a breakpoint.
  - Your app's execution stops, and the debugger shows the current state of the app. The current line is highlighted in your code.
- Click the **Step Over**  icon, select **Run > Step Over,** or type F8.
  - Step Over executes the next line of the code in the current class and method, executing all of the method calls on that line and remaining in the same file.
- Click the **Step Into**  icon, select **Run > Step Into,** or type F7.
  - Step Into jumps into the execution of a method call on the current line .If the method call is contained in another class, the file for that class is opened and the current line in that file is highlighted.
- Click the **Step Out**  icon, select **Run > Step Out,** or type Shift-F8.
  - Step Out finishes executing the current method and returns to the point where that method was called.

Publishing Your Application

After you have created, and fully debugged, your application, you might want to deploy it to the Google Store for others to enjoy. The following sections outline the steps for publishing your applications.

Generating a Signed APK

To publish your finished application on the Google Play Store, you must generate a signed APK(the Android application package).

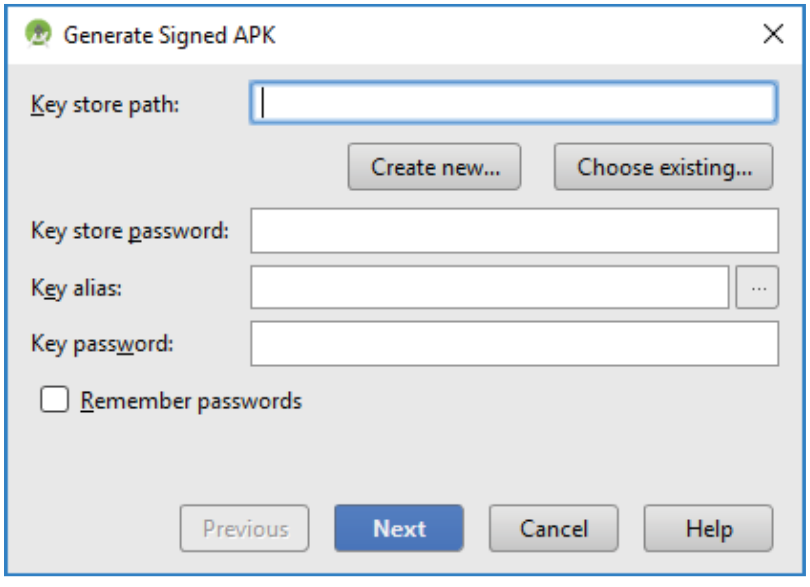
The APK is the compiled, executable version of your application.

Signing it is much like signing your name to a document. The signature identifies the app's developer to Google and the users who install your application.

More importantly, unless your Android Studio is in developer mode, unsigned applications will not run. Use the following steps to generate

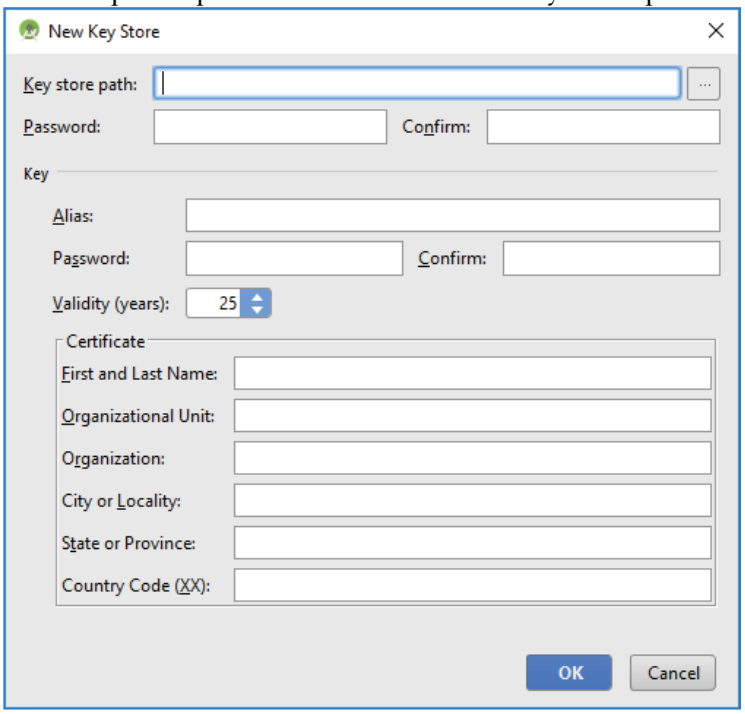
a signed APK:

1. Generate a signed APK from your code by selecting Build ➪ Generate Signed APK from the Menu bar to bring up the Generate Signed APK window



2. Assuming you have never published an application from Android Studio, you need to create a new key store. Click the Create New button to display the New Key Store window.

3. Fill out all of the information on this form because it pertains to your entity and application. Notice that there are two places for a password. These are the passwords for your key store and your key, respectively. Because a key store can hold multiple keys, it requires a separate password than that of the key for a specific app.



4. Click OK to return to the Generate Signed APK window.

5. In the Generate Signed APK windows, click Next to review and finish the process. Now that you have a signed APK, you can upload it to the Google Play Store using the developer console at **https://play.google.com/apps/publish/.**

Chapter-2 :

Using Activities - Fragments and Intents in Android: Working with activities, Using Intents, Fragments, Using the Intent Object to Invoke Built–in Application

Introduction to Activities in Android:

- An activity is a class that represents a single screen in android. It is like window or frame of Java.
- By the help of activity, you can place all your UI components (button, label, text field etc.) or widgets in a single screen.
- Activity is one of the most important components for any android app.
- It is similar to the main () function in different programming languages.
- It is the main entry point for user interaction.
- Any application, don't matter how small it is (in terms of code and scalability), has at least one Activity class. You can have multiple activities in your app.
- All your activities must be declared in the manifest file, with their attributes.
- Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

Lifecycle of an android activity:

Every activity has different functions throughout its life, onCreate (), onStart (), onResume (), onPause (), onStop (), onRestart (), onDestroy ().

Figure 3-2 shows the life cycle of an activity and the various stages it goes through—from when the activity is started until it ends.

- **The Activity class defines the following Methods:**

- **OnCreate()** — Called when the activity is first created
- **OnStart()** — Called when the activity becomes visible to the user
- **OnResume()** — Called when the activity starts interacting with the user **or** activity is becoming visible to the user and is ready to start receiving user interactions.
- **OnPause()** — Called when the current activity is being paused and the previous activity is being resumed. This is where you typically pause ongoing processes or save transient data.
- **OnStop()** — Called when the activity is no longer visible to the user.
- **OnDestroy()** — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- **OnRestart()** — Called when the activity has been stopped and is restarting again.

➢ By default, the activity created for you contains the onCreate() event. Within this event handler is the code that helps to display the UI elements of your screen.

Demo Android App to Demonstrate Activity Lifecycle in Android

```
package com.example.activity101;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity
{
String tag = "Lifecycle Step";
@Override
protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Log.d(tag, "In the onCreate() event");
}
public void onStart()
```

```
{
super.onStart();
Log.d(tag, "In the onStart() event");
}
public void onRestart()
{
super.onRestart();
Log.d(tag, "In the onRestart() event");
}
public void onResume()
{
super.onResume();
Log.d(tag, "In the onResume() event");
}
public void onPause()
{
super.onPause();
Log.d(tag, "In the onPause() event");
}
public void onStop()
{
super.onStop();
Log.d(tag, "In the onStop() event");
}
public void onDestroy()
{
super.onDestroy();
Log.d(tag, "In the onDestroy() event");
}
}
```

Explanation:

1. package com. example.activity101;

This declares the package name for the Java file (activity101.java).

2. import android.support.v7.app.AppCompatActivity;
   import android.os.Bundle;
   import android.util.Log;

These import statements bring in classes from the Android framework that is necessary for this activity. AppCompatActivity is the base class for activities that use the Support Library action bar features. Bundle is used for passing data between activities. Log is used for logging messages.

3. public class MainActivity extends AppCompatActivity
{

When a class extends another class in Java, it means that the subclass (in this case, MainActivity) inherits all the fields and methods from the super class (AppCompatActivity).

4. String tag = "Lifecycle Step";

This declares a string variable tag and initializes it with the value "Lifecycle Step". This tag will be used in logging messages to identify them.

5. @Override
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Log.d(tag, "In the onCreate() event");
}

- This method is called when the activity is first created. It sets the content view to a layout defined in activity_main.xml file. Then it logs a debug message indicating that onCreate() event has occurred.
- When an activity is destroyed and recreated due to configuration changes (such as screen rotation), Android preserves certain data from the activity's previous state when it's recreated. This preserved data is stored in the savedInstanceState bundle.

6.
```
@Override
public void onStart() { /* onStart code */ }

@Override
public void onRestart() { /* onRestart code */ }

@Override
public void onResume() { /* onResume code */ }

@Override
public void onPause() { /* onPause code */ }

@Override
public void onStop() { /* onStop code */ }

@Override
public void onDestroy() { /* onDestroy code */ }
```

These methods are overrides of the lifecycle callback methods provided by the AppCompatActivity class. Each of these methods is called by the Android system at specific points in the activity's lifecycle.

Logging Messages:

In each of the overridden methods, there's a call to Log.d() to log a debug message indicating the current lifecycle event.

Intents or Linking Activities Using Intents:

- An Android application can contain zero or more activities. When your application has more than one activity, you often need to navigate from one to another.
- In Android, you navigate between activities through what is known as intent.
- Intent is a messaging object used to request any action from another app component. Intent's most common use is to launch a new activity from the current activity.
- Intent facilitates communication between different components. Intent object is used to call other activities.
- The intent is used to launch an activity, start the services, broadcast receivers, display a web page, dial a phone call, send messages from one activity to another activity, and so on.

Common Use cases for Intents include:

1. **Starting Activities:** Use intents to launch new activities within your application or to launch activities from other applications.

For example, imagine you have a button in your app that says "Open Camera". When a user taps that button, you can use intent to ask the Android system to open the camera app.

2. **Broadcasting Messages:** Use intents to send broadcast messages within your application or to other applications, allowing them to receive and respond to events.
   Ex: let's say you have a music player app and you want to notify other apps whenever a new song starts playing.

- You would use a broadcast intent to send a message saying "Hey, a new song is playing!
- Other apps, like maybe a notification app or a social media app, can listen for this message. When they receive it, they can do things like show a notification saying what song is playing or share the song information on social media.
- Sent when the device finishes booting up, allowing apps to start up services or perform initialization tasks.

3. **Invoking Services:** Use intents to start services that perform background tasks or handle long-running operations.
    - Services: Background workers in your app that perform tasks without needing to be in the foreground.
    - Intents: Messages used to start services.

    Example
    - Create a service to download a file.
    - Register the service in the manifest.
    - Start the service using intent from an activity.

    Using services and intents allows your app to perform tasks like downloading files or playing music in the background, ensuring the main thread remains responsive and providing a smoother user experience.

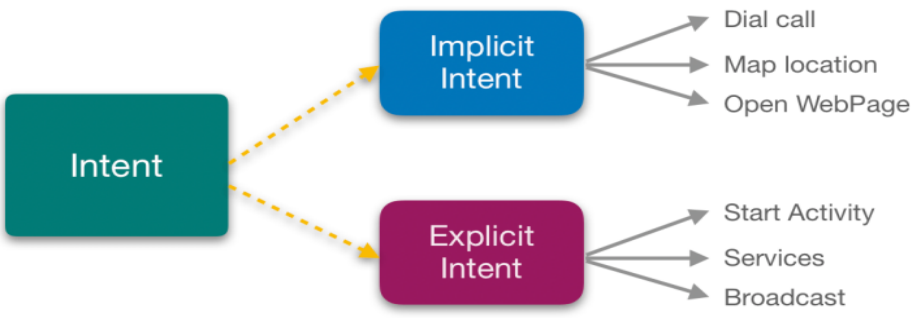
4. **Passing Data:** Intents can carry data (extra information) as key-value pairs, allowing components to exchange information, such as passing data between activities or between different parts of your application.
    - For example, let's say you have an app with two screens, and you want to send a message from the first screen to the second screen. You would use intent to carry that message.
    - **First Screen (Calling Screen):** You create intent and put the information about the call, like the caller's name, as extras.
    - **Second Screen (Receiving Screen):** You get the information about the call from the intent.

Methods and their Description:

Methods	Description
Context.startActivity()	This is to launch a new activity or get an existing activity to be action.
Context.startService()	This is to start a new service or deliver instructions for an existing service.
Context.sendBroadcast()	This is to deliver the message to broadcast receivers.

TYPES OF INTENT

- Intents are of two types:



- Explicit Intent is used to invoke a specific target component. It is used to switch from one activity to another activity in the same application. It is also used to pass data by invoking the external class.

- Explicit Intents specify the target component by providing the exact class name of the component to be invoked.

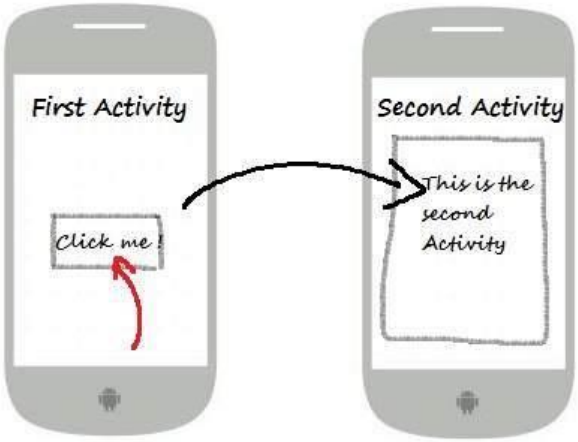
Example:-

1. We can use explicit intent to start a new activity when the user invokes an action or plays music in the background or on click button go to another activity.

   Intent intent = new Intent(MainActivity.this, SecondActivity.class);
   startActivity(intent);

   - MainActivity.this specifies the current context, which is the MainActivity.
   - SecondActivity.class specifies the target activity to which you want to navigate.
   - In Explicit we use the name of component which will be affected by Intent. **For Example:** If we know class name then we can navigate the app from One Activity to another activity using Intent.

2. In amazon app if you go to Home page you can see prime, fresh, mobiles, electronics etc. If you click prime it transit to prime page activity likewise other tab also works.



IMPLICIT INTENT:

- An implicit intent is used when you want to perform an action, but you don't care which component performs it.

- The system will determine the appropriate component to handle the intent based on the available components that can respond to it.
- Example: Suppose you want to open a website URL in a web browser. You don't know which browser the user prefers, so you'll use an implicit intent:

```
//Intent object and open the webpage
Intent intent = new
Intent(Intent.ACTION_VIEW,Uri.parse("https://example.com"));
startActivity(intent); //call a webpage
```

- Intent.ACTION_VIEW is the action you want to perform, which is viewing content.
- Uri.parse("https://example.com") specifies the data you want to view, which is a website URL.
  - Explicit intents are used for navigating within your own by specifying the target component, while implicit intents are used for performing actions where the system determines the appropriate component to handle the request.

Using the Intent Object to Invoke Built–in Application:

- One of the key aspects of Android programming is using the intent to call activities from other Applications. An Application can call many built-in Applications, which are included with an Android device.
- Suppose, you want to load a Web page, which is not part of your Application. You can use the Intent object to invoke the built-in Web Browser to display the Web page, instead of building your own Web Browser for this purpose.
- Add three buttons are in activity_main.xml file as Web Browser. Call here and display Map.

Code:

```
<Button
android:id="@+id/buttonwebbrowser"
android:text="Web Browser"
```

```
android:onClick="onClickBrowseWeb"/>
<Button
android:id="@+id/buttondocall"
android:text="Call Here"
android:onClick="onClickCallHere"/>
<Button
android:id="@+id/buttondisplaymap"
android:text="Display Map"
android:onClick="onClickDisplayMap"/>
```

Add three methods are in MainActivity.java class to display the Website in the Web Browser. Call on any mobile number and display the map by Google maps.

Code:

```
public void onClickBrowseWeb(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.co.in"));
    startActivity(intent);
  }
  public void onClickCallHere(View view) {
    Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:+91xxxxxxxxxx"));
    startActivity(intent);
  }

                    Or
setContentView(R.layout.activity_main);
String[] phoneNumbers = {
      "+91xxxxxxxxxx", // India
      "+1xxxxxxxxxx",  // USA/Canada
      "+44xxxxxxxxxx", // UK
      "+61xxxxxxxxxx", // Australia
      "+81xxxxxxxxxx"  // Japan
    };

private void startCallIntent(String phoneNumber) {
    Intent intent = new Intent(Intent.ACTION_CALL);
    intent.setData(Uri.parse("tel:" + phoneNumber));
```

```
    startActivity(intent);
}
```

To display the Map based upon lat,long Co ordinates

```
public void onClickDisplayMap(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:28.7041,77.1025"));
    startActivity(intent);
}
```

Or

To display the Map based upon Location name

```
public void onClickDisplayMap(View view) {
    String location = editTextLocation.getText().toString();
    if (!location.isEmpty()) {
        Uri geoLocation = Uri.parse("geo:0,0?q=" + Uri.encode(location));
        showMap(geoLocation);
    }
}
```

In the first button, create an object of Intent and then pass the two arguments to its constructor. The action and the data will be,

```
1. Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.co.in"));


2. startActivity(intent);
```

The action here is represented by the Intent.ACION_VIEW constant. We used the parse() method of the URL class to convert a URL string into a URL object.

For the second button, we can dial a specific number by passing in the telephone number in the portion.

```
1.  Intent intent=new Intent(Intent.ACTION_DIAL, URL.parse("tel:+91xxxxxxxxxx"));
```

```
2. startActivity(intent);
```

In this case, the dialler will display the number to be called. The user must still click the dial button to dial the number. If you want to directly call the number without the user intervention, change the action, given below,

```
1. Intent intent=new Intent(Intent.ACTION_CALL, URL.parse("tel:+91xxxxxxxxxx"));
2. startActivity(intent);
```

For this, you need to assign the Android.permission.CALL_PHONE permission to your Application, defined in the manifest file.

You can simply omit the data portion to display the dialer without specifying any number.

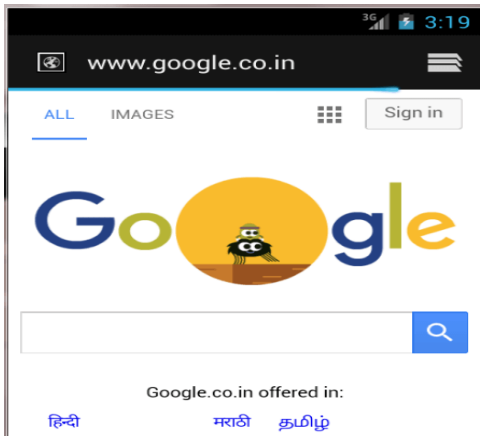
```
1. Intent intent=new Intent(Intent.ACTION_DIAL);
2. startActivity(intent);
```

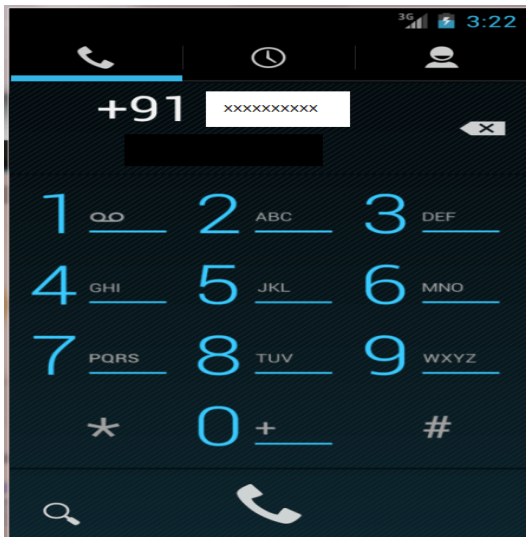
The third button displays a map using the ACTION_VIEW constant. Here we use “geo” in place of “http”.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW,Uri.parse("geo:28.7041,77.1025"));
2. startActivity(intent);
```

In this app, we have used the Intent class to invoke some of the built-in Applications in Android like Browser, Phone, and Maps).

- Now, click on the first option. Google page will open in the Browser.
- Click the second option. It will open a dial-up option to make a call to the specified mobile number.
- To load the Maps Application, click the display map button. To display the Maps Application, you need to run the Application on an AVD, which supports the Google APIs or you have to run this app on your device.





Intent filter

- Implicit intent uses the intent filter to serve the user request.
- The intent filter specifies the types of intents that an activity, service, or broadcast receiver can respond to.
- Intent filters are declared in the Android manifest file.
- Intent filter must contain <action>

1. **<actions>** In this, you can keep all the actions you wish your intent to accept.

Constant	Target Component	Action
ACTION_CALL	Activity	Initiate a phone call
ACTION_EDIT	Activity	Display data for the user to edit
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.

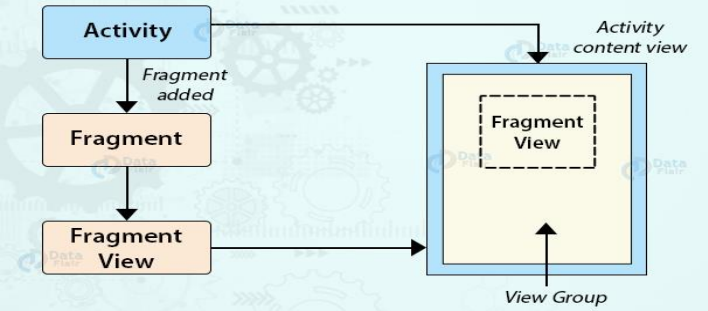
2. **<data>** It defines the type of data that the intent will take.

- Example: I f the action field is ACTION_EDIT, the data field would contain the URI of the document to be displayed for editing.

3. **<Category>** It represents the name of the category that the intent will accept.A string containing additional information about the kind of component that should handle the intent.

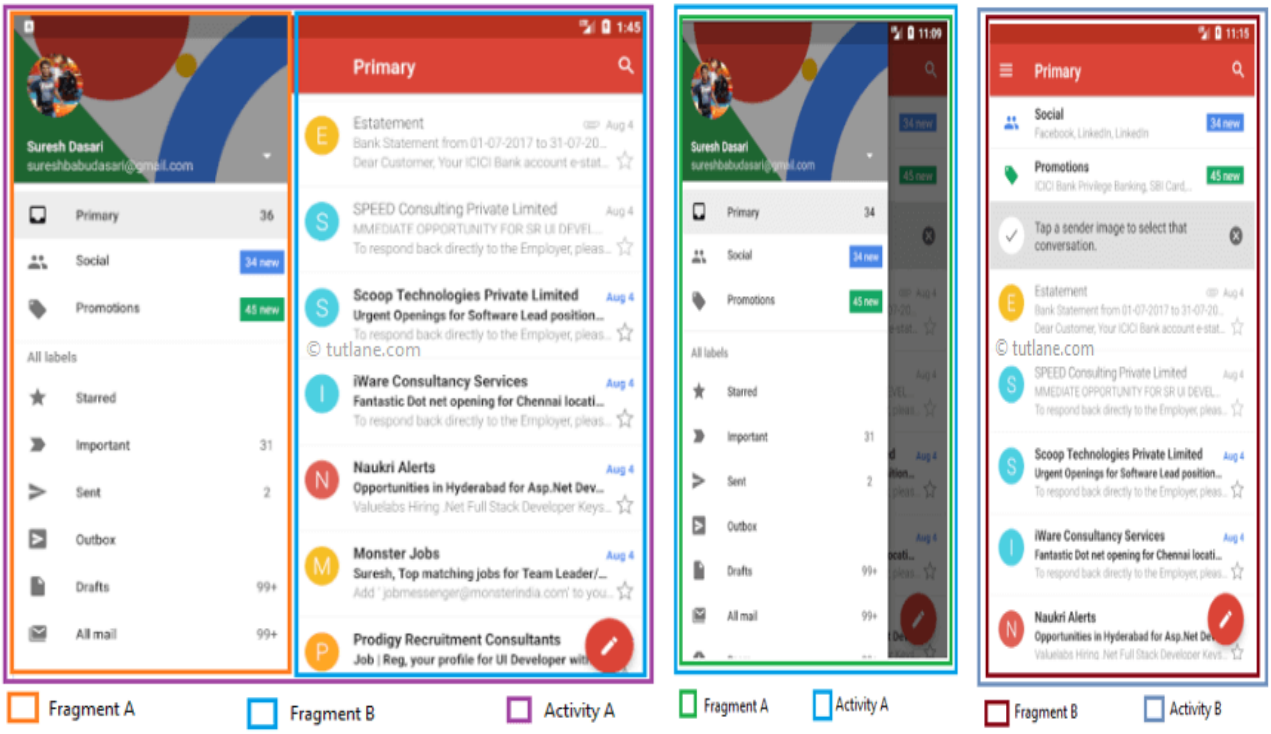
Android Fragments:

- Android Fragment is a Graphical User Interface component of Android. It resides within the Activities of an Android application. It represents a portion of UI that the user sees on the screen.
- Android Fragments cannot exist outside an activity. Another name for Fragment can be **Sub-Activity** as they are part of Activities.
- Fragments are always embedded in Activities; Multiple Fragments can be added to single activity.

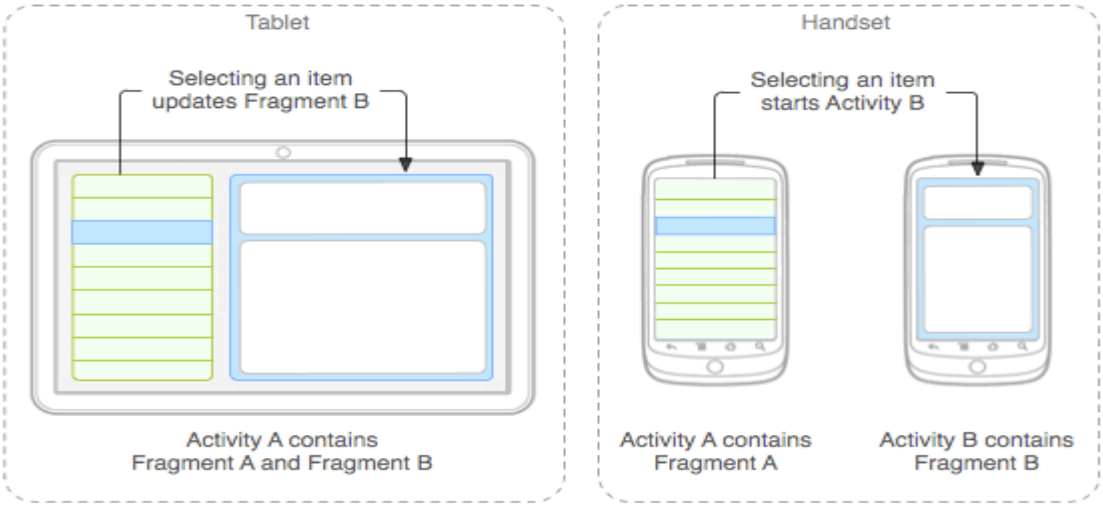


How Fragment Interacts with Activity in Different Devices:

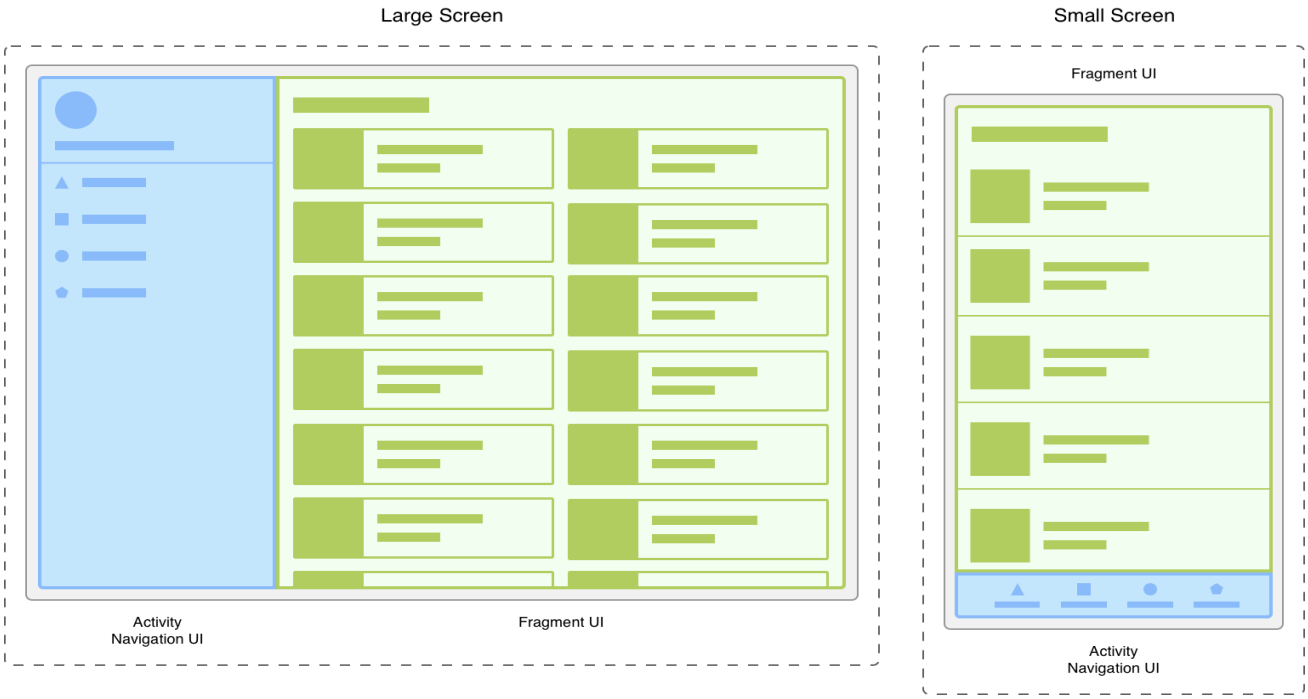
- If you observe above example for **Tablet** we defined an **Activity A** with two fragments such as one is to show the list of items and second one is to show the details of item which we selected in first fragment.
- For **Handset** device, there is no enough space to show both the fragments in single activity, so the **Activity A** includes first fragment to show the list of items and the **Activity B** which includes another fragment to display the details of an item which is selected in **Activity A**.
- For example, **GMAIL app** is designed with multiple fragments, so the design of GMAIL app will be varied based on the size of device such as tablet or mobile device.



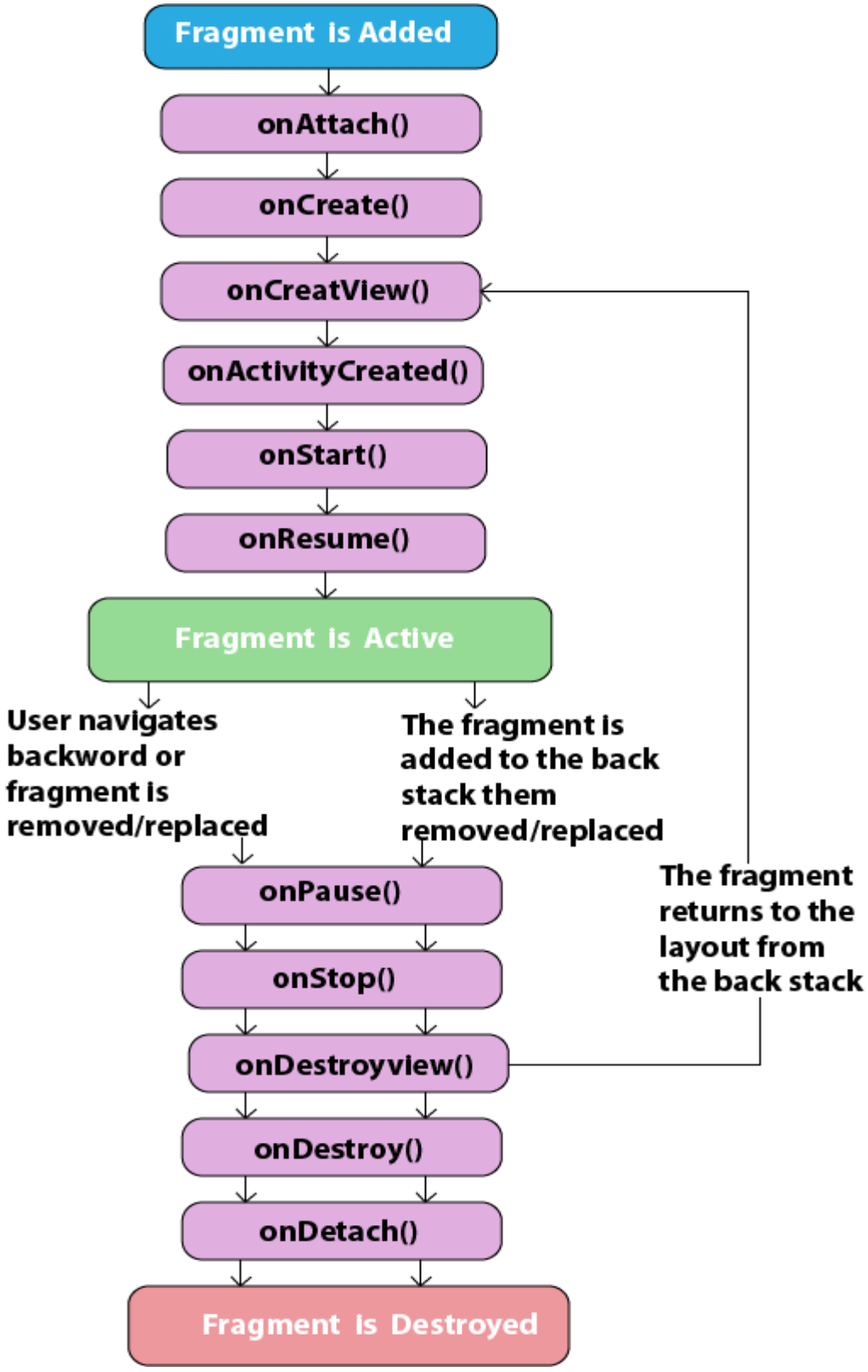
Tablet View **Mobile View**



- **In the case of mobiles, there are two activities that are:**
  - Activity 1 with Fragment A and Activity 2 with Fragment B. When we select an item from Fragment A, it gets open in the Fragment B of Activity 2.
- **In tablets, there is only one activity that is Activity 1.**
  - In **Activity 1**, there are two fragments, **Fragment A** and **Fragment B**. When we select an item from Fragment A, it gets open in Fragment B of the same activity.
- On larger screens, you might want the app to display a static navigation drawer and a list in a grid layout. On smaller screens, you might want the app to display a bottom navigation bar and a list in a linear layout.



Android Fragment Lifecycle:



Method	Description
onAttach()	It is called when the fragment has been associated with an activity.
onCreate()	It is used to initialize the fragment.
onCreteView()	It is used to create a view hierarchy associated with the fragment.
onActivityCreated()	It is called when the fragment activity has been created and the fragment view hierarchy instantiated.
onStart()	It is used to make the fragment visible.
onResume()	It is used to make the fragment visible in an activity.
onPause()	It is called when fragment is no longer visible and it indicates that the user is leaving the fragment.
onStop()	It is called to stop the fragment using the onStop() method.
onDestoryView()	The view hierarchy associated with the fragment is being removed after executing this method.
onDestroy()	It is called to perform a final clean up of the fragments state.
onDetach()	It is called immediately after the fragment disassociated from the activity.

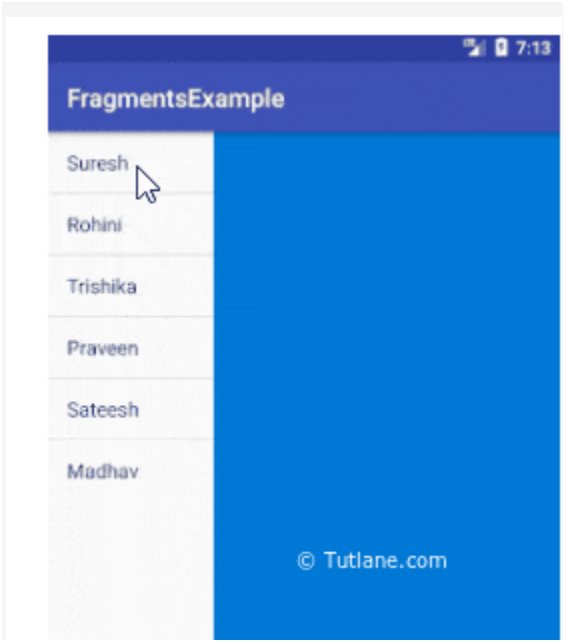
ListMenuFragment.java

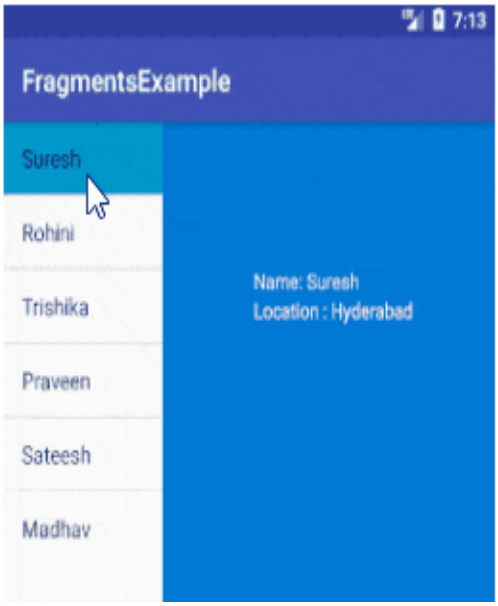
```
public class ListMenuFragment extends ListFragment {
    String[] users = new String[]
{ "Suresh","Rohini","Trishika","Praveen","Sateesh","Madhav" };

String[] location = new String[]{"Hyderabad","Guntur","Hyderabad","Bangalore","Vizag","Nag
pur"};
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view =inflater.inflate(R.layout.listitems_info, container, false);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
                android.R.layout.simple_list_item_1, users);
```

activity_main.xml:

```
<fragment
        android:layout_height="match_parent"
        android:layout_width="350px"
        class="com.tutlane.fragmentsexample.ListMenuFragment"
        android:id="@+id/fragment"/>
    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.tutlane.fragmentsexample.DetailsFragment"
        android:id="@+id/fragment2"/>
```





Adding Fragments with Activities:

Embedding Fragments with Activities means adding the Fragments to the respective Activity Layout. Now, there are two ways for adding multiple Fragments in one Activity:

1. Statically

To add the fragment statically, we need to mention it ourselves in the these fragments can't be replaced during the execution as they are static. Defined in XML, fixed during compile-time, and cannot be changed at runtime.

2. Dynamically

In this, we embed our Fragment in Activities dynamically using **Fragment Manager**.

Unlike Static Fragment, in this, we can add, remove or replace the Fragments at the runtime itself. Managed in code, flexible, can be added, removed, or replaced at runtime.

Types of Android Fragments

1. Single Fragments

Single fragments show only a single view for the user on the screen. These are for handheld devices such as mobile phones.

2. List Fragments

List fragments are those that have a special list view feature. In this, there's a list and the user can choose to see a Sub-Activity.

3. Fragment Transactions:

Fragment transactions are for the transition from one fragment to another. It supports switching between two fragments.