# MOVIE-DB (ANDROID)

Summer Internship Report

submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

Information Technology

by

**SHASHANK TEWARI – 42696303117**

Maharaja Surajmal Institute of Technology

(Affiliated to Guru Gobind Singh Indraprastha University)

Janakpuri, New Delhi-58

September 2019

# STUDENT CERTIFICATE

I, SHASHANK TEWARI, Roll No 42696303117, B.tech (Semester – 5th) of the Maharaja Surajmal Institute of Technology, New Delhi hereby declare that the Training Report entitled "MOVIE-DB (ANDROID DEVELOPMENT)" is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any degree.

SHASHANK TEWARI

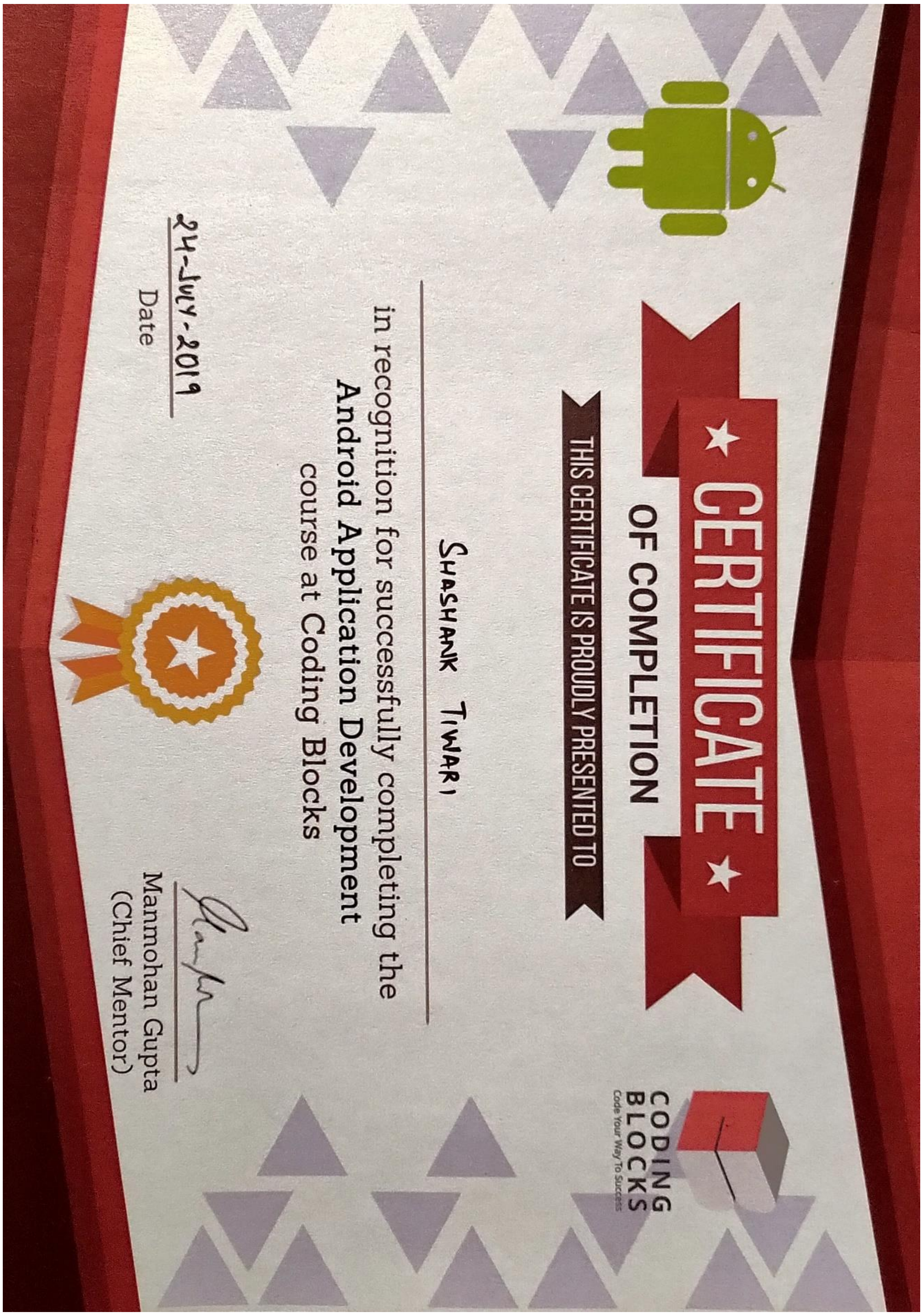
(Roll No 42696303117)

Place: Maharaja Surajmal

Institute Of Technology

Date:

## ORGANIZATION CERTIFICATE



CERTIFICATE

OF COMPLETION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

SHASHANK TIWARI

in recognition for successfully completing the **Android Application Development** course at Coding Blocks

24-JULY-2019

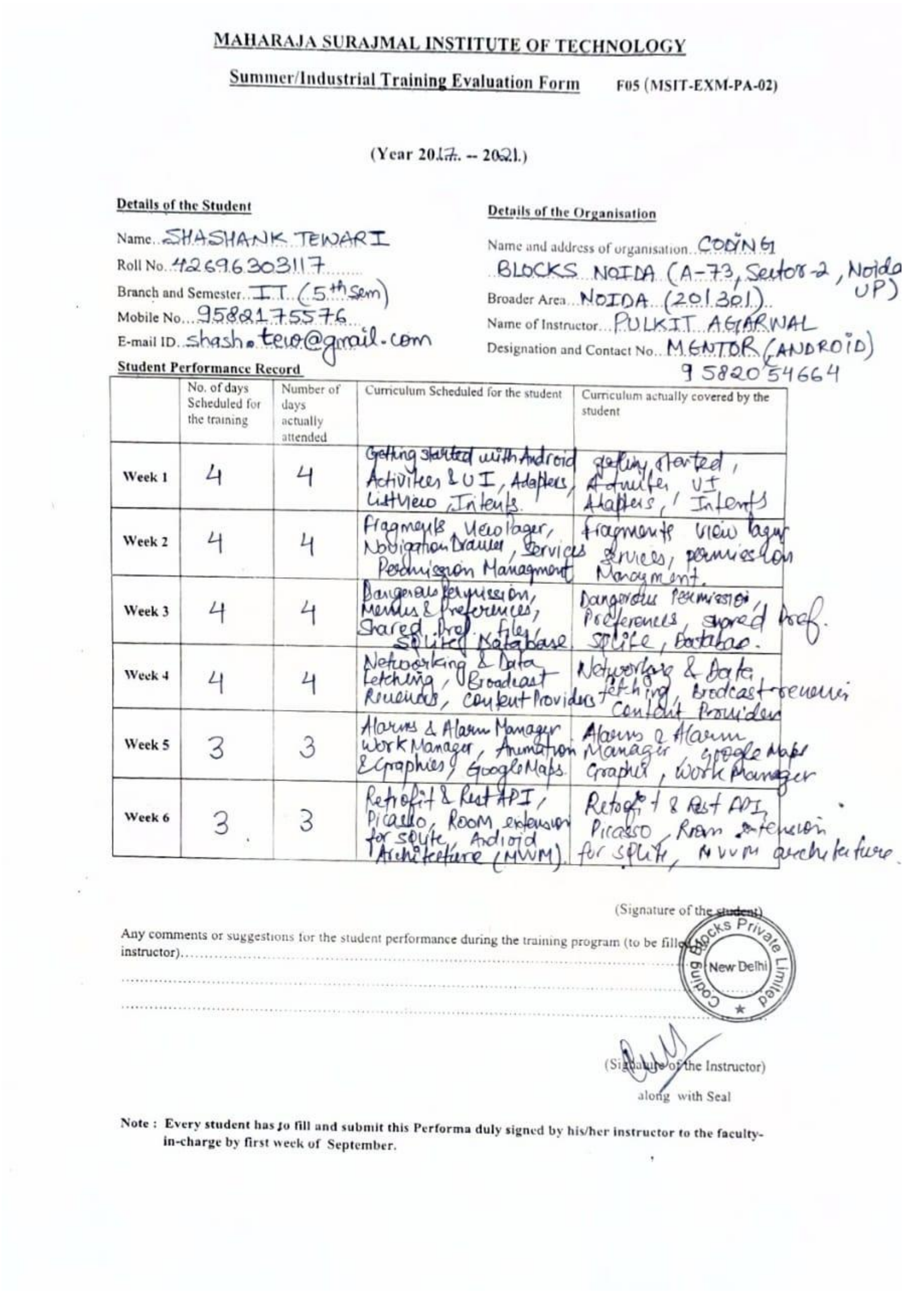
Date

[illegible]

Manmohan Gupta
(Chief Mentor)

CODING BLOCKS
Code Your Way To Success

# EVALUATION FORM



**MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY**

**Summer/Industrial Training Evaluation Form** F05 (MSIT-EXM-PA-02)

(Year 2017 -- 2021)

**Details of the Student**

Name: SHASHANK TEWARI
Roll No.: 42696303117
Branch and Semester: I.T. (5th sem)
Mobile No.: 9582175576
E-mail ID: shash.tew@gmail.com

**Details of the Organisation**

Name and address of organisation: CODING BLOCKS NOIDA (A-73, Sector-2, Noida UP)
Broader Area: NOIDA (201301)
Name of Instructor: PULKIT AGARWAL
Designation and Contact No.: MENTOR (ANDROID) 9582054664

**Student Performance Record**

| | No. of days Scheduled for the training | Number of days actually attended | Curriculum Scheduled for the student | Curriculum actually covered by the student |
|---|---|---|---|---|
| **Week 1** | 4 | 4 | Getting started with Android Activities & UI, Adapters, ListView, Intents. | getting started, Activities, UI, Adapters, Intents |
| **Week 2** | 4 | 4 | Fragments, ViewPager, Navigation Drawer, Services Permission Management | fragments view pager services, permission Management. |
| **Week 3** | 4 | 4 | Dangerous Permission, Menus & Preferences, Shared Pref. files SQLite Database | Dangerous Permission, Preferences, shared pref. SQLite, Database. |
| **Week 4** | 4 | 4 | Networking & Data Fetching, Broadcast Receivers, Content Providers | Networking & Data fetching, Brodcast receivers Content Providers |
| **Week 5** | 3 | 3 | Alarms & Alarm Manager Work Manager, Animation & Graphics, GoogleMaps. | Alarms & Alarm Manager google Maps Graphics, Work Manager |
| **Week 6** | 3 | 3 | Retrofit & Rest API, Picasso, ROOM extension for SQLite, Android Architecture (MVVM). | Retrofit & Rest API, Picasso, Room extension for SQLite, MVVM architecture. |

(Signature of the student)

Any comments or suggestions for the student performance during the training program (to be filled by instructor)........................................

Coding Blocks Private Limited ★ New Delhi

[illegible]
(Signature of the Instructor)
along with Seal

**Note : Every student has to fill and submit this Performa duly signed by his/her instructor to the faculty-in-charge by first week of September.**

# ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. PULKIT AGGARWAL for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Pulkit Sir constantly supported me towards the course of my project development and helped me in every possible way. He is a complete source of knowledge and patience who guided me through this path.

I would like to express my gratitude towards my parents and faculty of the Coding Blocks for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

# LIST OF FIGURES

# LIST OF TABLES

# TABLE OF CONTENTS

## Contents

# ABSTRACT

I have done a summer training in Android Development from Coding Blocks. This Project report is the report of the end project that I have done during the training. I have learned about the Android and how applications can be developed. In my training I used Android Studio as a software for developing the project. The project contains code that is written in the Kotlin language, a language which works on bytecode and like Java. In the project I made an app called Movie-Db which is movie and tv show app. This is app is intended towards the user as a platform like IMDB. The app fetches all its contents from the TMDB api which is the same Api Imdb uses in its application and website. My main purpose of making this app is learning and applying my knowledge of android that I gained in the training. This app is basically a clone of another app name as Popcorn, which is present in the Playstore. I tried making my app similar to it, but added some extra features to it. In the development I used several libraries and frameworks like Retrofit and OkHttp, they are used for making the network call to the Tmdb APi. Other than this I have stored the details of the movie/show in the form of favourites. This favourite feature of the app is done using Room library, which is the basically based on SQLite. Other than this I have implemented several other features like added voice search in the search widget and added YouTube sdk in the app itself so that trailer can be played in the app itself. This application is product ready and I can publish this to playstore, but as playstore requires membership, so I instead made it open sourced.

# CHAPTER 1

## 1. INTRODUCTION

**Android** is a mobile operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has 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.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 9 "Pie", released in August 2018. Google released the first beta of the next release, Android Q, on Pixel phones in March 2019. The core Android source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License.

Android is also associated with a suite of proprietary software developed by Google, called Google Mobile Services (GMS), that frequently comes pre-installed on devices. This includes core apps such as Gmail, the application store/digital distribution platform Google Play and associated Google Play Services development platform, and usually includes the Google Chrome web browser and Google Search app. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems such as Amazon.com's Fire OS, which use their own equivalents to Google Mobile Services.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of December 2018, the Google Play store features over 2.6 million apps.

### 1.1. HISTORY OF ANDROID

Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. Rubin described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences". The early intentions of the company were to develop an advanced operating system for digital cameras, and this was the basis of its pitch to investors in April 2004. The company then decided that the market for cameras was not large enough for its goals, and by five months later it had diverted its efforts and was pitching Android as a handset operating system that would rival Symbian and Microsoft Windows Mobile.

Rubin had difficulty attracting investors early on, and Android was facing eviction from its office space. Steve Perlman, a close friend of Rubin, brought him $10,000 in cash in an envelope, and shortly thereafter wired an undisclosed amount as seed funding. Perlman refused a stake in the company and has stated "I did it because I believed in the thing, and I wanted to help Andy."

In July 2005, Google acquired Android Inc. for at least $50 million. Its key employees, including Rubin, Miner and White, joined Google as part of the acquisition. Not much was known about the secretive Android at the time, with the company having provided few details other than that it was making software for mobile phones. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradeable system. Google had "lined up a series of hardware components and software partners and signaled to carriers that it was open to various degrees of cooperation".

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. An early prototype had a close resemblance to a BlackBerry phone, with no touchscreen and a physical QWERTY keyboard, but the arrival of 2007's Apple iPhone meant that Android "had to go back to the drawing board". Google later changed its Android specification documents to state that "Touchscreens will be supported", although "the Product was designed with the presence of discrete physical buttons as an assumption, therefore a touchscreen cannot completely replace physical buttons". By 2008, both Nokia and BlackBerry announced touch-based smartphones to rival the iPhone 3G, and Android's focus eventually switched to just touchscreens. The first commercially available smartphone running Android was the HTC Dream, also known as T-Mobile G1, announced on September 23, 2008.

## 1.2. VERSION HITORY OF ANDROID

The version history of the Android mobile operating system began with the public release of the Android beta on November 5, 2007. The first commercial version, Android 1.0, was released on September 23, 2008. Android is continually developed by Google and the Open Handset Alliance, and it has seen several updates to its base operating system since the initial release.

Table -1 Android Versions

| **CODE NAME** | **VERSION NAME** | **LINUX KERNEL VERSION** | **RELEASE DATE** | **API LEVEL** |
|---|---|---|---|---|
| NO CODENAME | 1.0 | 2.1 | Sep 23, 2008 | 1 |
| PETIT FOUR | 1.1 | 2.6 | Feb 9, 2009 | 2 |
| CUPCAKE | 1.5 | 2.6.27 | April 27, 2009 | 3 |
| DONUT | 1.6 | 2.6.29 | Sep 15, 2009 | 4 |
| ÉCLAIR | 2.0 – 2.1 | 2.6.29 | October 26, 2009 | 5 – 7 |
| FROYO | 2.2 – 2.2.3 | 2.6.32 | May 20, 2010 | 8 |
| GINGERBREAD | 2.3 – 2.3.7 | 2.6.35 | Dec 6, 2010 | 9 – 10 |
| HONEYCOMB | 3.0 – 3.2.6 | 2.6.36 | Feb 22, 2011 | 11 – 13 |
| ICE CREAM | 4.0 – 4.0.4 | 3.0.1 | October 18, 2011 | 14 – 15 |
| JELLYBEAN | 4.1 – 4.3.1 | 3.0.39 – 3.4.39 | July 9, 2012 | 16 – 18 |

| | | | | |
|---|---|---|---|---|
| KITKAT | 4.4 – 4.4.4 | 3.10 | October 31, 2013 | 19 – 20 |
| LOLLIPOP | 5.0 – 5.1.1 | 3.16 | Nov 12, 2014 | 21 – 22 |
| MARSHMELLOW | 6.0 – 6.0.1 | 3.18 | October 5, 2015 | 23 |
| NOUGOT | 7.0 – 7.1.2 | 4.4 | August 22, 2016 | 24 – 25 |
| OREO | 8.0 – 8.1 | 4.10 | August 21, 2017 | 26 – 27 |
| PIE | 9.0 | 4.4.107 | August 6, 2018 | 28 |
| ANDROID 10 | 10.0 | | | 29 |

## 1.3. INTERFACE

Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard. Game controllers and full-size physical keyboards are supported via Bluetooth or USB. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware, such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

Android devices boot to the homescreen, the primary navigation and information "hub" on Android devices, analogous to the desktop found on personal computers. Android homescreens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content, such as a weather forecast, the user's email inbox, or a news ticker directly on the homescreen. A homescreen may be made up of several pages, between which the user can swipe back and forth. Third-party apps available on Google Play and other app stores can extensively re-theme the homescreen, and even mimic the look of other operating systems, such as Windows Phone. Most manufacturers customize the look and features of their Android devices to differentiate themselves from their competitors.

Along the top of the screen is a status bar, showing information about the device and its connectivity. This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates. Notifications are "short, timely, and relevant information about your app when it's not in use", and when tapped, users are directed to a screen inside the app relating to the notification. Beginning with Android 4.1 "Jelly Bean", "expandable notifications" allow the user to tap an icon on the notification for it to expand and display more information and possible app actions right from the notification.

An All Apps screen lists all installed applications, with the ability for users to drag an app from the list onto the home screen. A Recents screen lets users switch between recently used apps.

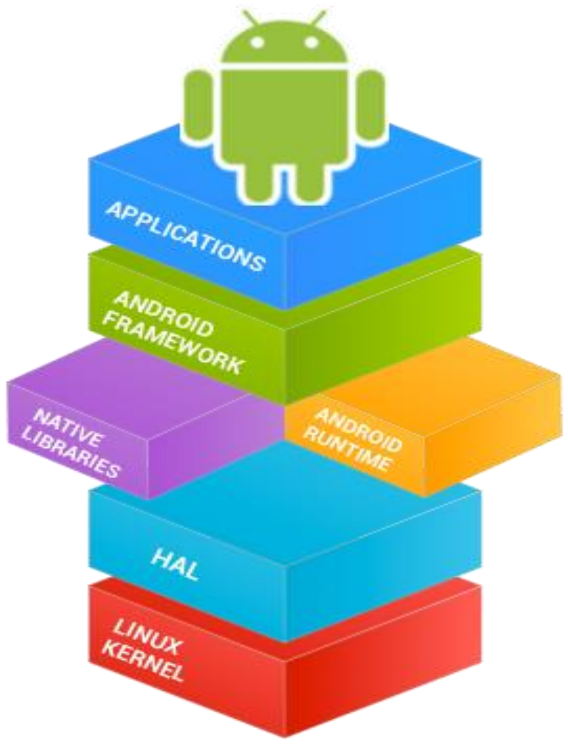


Figure 1 Android Stack

# 2. ABOUT ORGANIZATION

## 2.1. CODING BLOCKS

Coding is the most in-demand skill of the twenty first century. Coding Blocks was started as a learning centre where we teach fundamentals of programming to college students. The courses here are designed to help students with their curriculum and give them a real feel of the IT industry. Coding Blocks was founded in 2014 by Manmohan Gupta (programmer, mathematician and a learner) with a mission to create skilled software engineers for our country and the world. Coding Blocks aim to bridge the gap between the quality of skills demanded by industry and the quality of skills imparted by conventional institutes. Coding Blocks strongly believe that with the right guidance and perfect determination, any student willing to learn programming can become a master of coding. All team members at Coding Blocks are aces of their respective field and the share the highest level of commitment towards quality teaching and student success and satisfaction. The present structure of computer education in colleges and universities is not aligned to the needs of the IT Industry. Students have no place to go and bridge this huge gap. Coding Blocks was started as a learning centre where we teach fundamentals of programming to college students. The courses here are designed to help students with their curriculum and give them a real feel of the IT industry. In the last three years, we have helped more than 7200 students achieve their goal and made them the darling of the industry. At Coding Blocks, all our instructors are themselves great coders and highly employable in the Industry. They decided to join Coding Blocks instead of lucrative jobs in the industry because they are passionate about teaching and strongly believe in the company's vision.

CODING
BLOCKS
Code Your Way To Success

Figure 2 Coding blocks Logo

### 2.2. COURSES / TRAININGS

Coding Blocks offers many different courses and trainings to the students. The course material is divided into four different levels namely-

1) Beginner level
   a) C++ for beginners/Launchpad
   b) Java for beginners/Crux
   c) Python app development
2) Development Level
   a) Android
   b) Full Stack Web Development with Node.js
   c) Web Development with Django-Python
   d) Unity Game Development
   e) Web Backend using Java
3) Advance Level
   a) Algo++
   b) Algo.java
   c) Machine Learning
   d) Data Science
   e) Competitive Programming
4) Bootcamp
   a) React Js
   b) Interview Preparation
   c) Block chain
   d) Chatbots
   e) Angular

All these courses are of around 6-8 weeks and have 22-24 classes.

## 3. ANDROID DEVELOPMENT TRAINING

### 3.1. OVERVIEW

There are close to 2 billion Android powered devices as of 2016, and for more than 200 million Indians Android is the primary platform for consumption of content. From social and communication apps like WhatsApp, Instagram and Snapchat to utilities like Uber, Swiggy and Paytm, Android enables companies to cater to their users 24x7.

Our Android App Development course takes you on a comprehensive journey with the Android Studio that starts from building the UI to connecting to network services and launching your product on the Google Play Store. Since most products look for both web and mobile presence these days, you'll be an in-demand Android Developer the day you graduate our Course.

**Figure - 3 Android**

- **Ui Design**

  An app with great functionality must be complemented by top notch User Experience and a neat User Interface. This course covers building both common and upcoming UI/UX pattern.

- **Firebase & Push Notifications**

  Firebase is Google’s service that provides a real time database and push notifications, so that we can send information to the user even when he is not using the app.

- **Hardware Sensors**

Sensors help us build context-aware apps. We will learn how to use GPS location and perform actions on gesture such as flips and shakes

- **Database**

Most apps are connected in nature. We will learn how to connect the app to a backend to create an ecosystem of users and their data.

## 3.2 COURSE CONTENTS

### 1. ACTIVITIES AND UI

The basic building blocks of an Android app – 'the activity'. Learn to make beautifully designed apps, with all kind of screen sizes and all type of layouts. Learn how to build 'List views' for showing the data in beautiful cards, common Navigation drawer, Action bar, Floating action button and all Material Design concepts.

- **Object oriented programming concepts in Java.**
- **Getting started with Android app development tools.**
- **Activities and basic UI building blocks.**

### 2. BACKGROUND AND SERVICES

Not everything happens on the screen, learn to create services that run in Background. AsyncTasks use multi-threading to run parallel operations on separate threads so the user doesn't notice any work being done, and the app remains ever responsive.

- **Adapters and List views**
- **Intents**
- **Fragments, View Pager and Navigation Drawer**
- **Services**

- **Permission Management**
- **Menus and Preferences**

3. **DATA STORAGE**

Three ways of Data Storage available in Android are –

- **Shared Preferences**
- **Flies**
- **SQLite Database**

4. **NETWORK**

A mobile app is nothing without connectivity. Learn to add network connectivity and integrate your app with remote servers. the course will also cover how to use backend services like Firebase to create real-time databases and push notifications to create chat-based applications.

- **Network and data fetching**
- **Broadcast Receivers**
- **Hardware Sensors**
- **Animations and Graphics**
- **Alarms and Alarm Manager**
- **Content Providers**

5. **3rd PARTY LIBRARIES**

Learn to Integrate most commonly used Libraries that are used to make powerful and interactive apps like

- **Google Maps**
- **Retrofit and Rest API**
- **Picasso and Glide for showing image**

# CHAPTER -2

## 1. ANDROID SOFTWARE DEVELOPMENT

**Android software development** is the process by which new applications are created for devices running the Android operating system. Google states that "Android apps can be written using Kotlin, Java, and C++ languages" using the Android software development kit (SDK), while using other languages is also possible. All non-JVM languages, such as Go, JavaScript, C, C++ or assembly, need the help of JVM language code, that may be supplied by tools, likely with restricted API support. Some programming languages and tools allow cross-platform app support (i.e. for both Android and iOS). Third party tools, development environments, and language support have also continued to evolve and expand since the initial SDK was released in 2008. In addition, with major business entities like Walmart, Amazon, and America eyeing to engage and sell through mobiles, mobile application development is witnessing a transformation.

### 1.1. ANDROID SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially-supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go together with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains. dex files (compiled byte code files called Dalvik executables), resource files, etc.

## 1.2. ANDROID SDK PLATFORM TOOLS

The Android SDK Platform Tools are a separately downloadable subset of the full SDK, consisting of command-line tools such as adb and fast boot.

## 1.3. LINUX KERNEL

Android's kernel is based on the Linux kernel's long-term support (LTS) branches. As of 2018, Android targets versions 4.4, 4.9 or 4.14 of the Linux kernels. The actual kernel depends on the individual device.

Android's variant of the Linux kernel has further architectural changes that are implemented by Google outside the typical Linux kernel development cycle, such as the inclusion of components like device trees, ashmem, ION, and different out of memory (OOM) handling. Certain features that Google contributed back to the Linux kernel, notably a power management feature called "wakelocks", were initially rejected by mainline kernel developers partly because they felt that Google did not show any intent to maintain its own code. Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, but Greg Kroah-Hartman, the current Linux kernel maintainer for the stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux. Google engineer Patrick Brady once stated in the company's developer conference that "Android is not Linux", with Computerworld adding that "Let me make it simple for you, without

Linux, there is no Android". *Ars Technica* wrote that "Although Android is built on top of the Linux kernel, the platform has very little in common with the conventional desktop Linux stack".

In August 2011, Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years". In December 2011, Greg Kroah-Hartman announced the start of Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. Linux included the autosleep and wakelocks capabilities in the 3.5 kernel, after many previous attempts at merger. The interfaces are the same, but the upstream Linux implementation allows for two different suspend modes: to memory (the traditional suspend that Android uses), and to disk (hibernate, as it is known on the desktop). Google maintains a public code repository that contains their experimental work to re-base Android off the latest stable Linux versions.

The flash storage on Android devices is split into several partitions, such as *`/system`* for the operating system itself, and `/data` for user data and application installations. In contrast to desktop Linux distributions, Android device owners are not given root access to the operating system and sensitive partitions such as /system are read-only. However, root access can be obtained by exploiting security flaws in Android, which is used frequently by the open-source community to enhance the capabilities of their devices, but also by malicious parties to install viruses and malware.

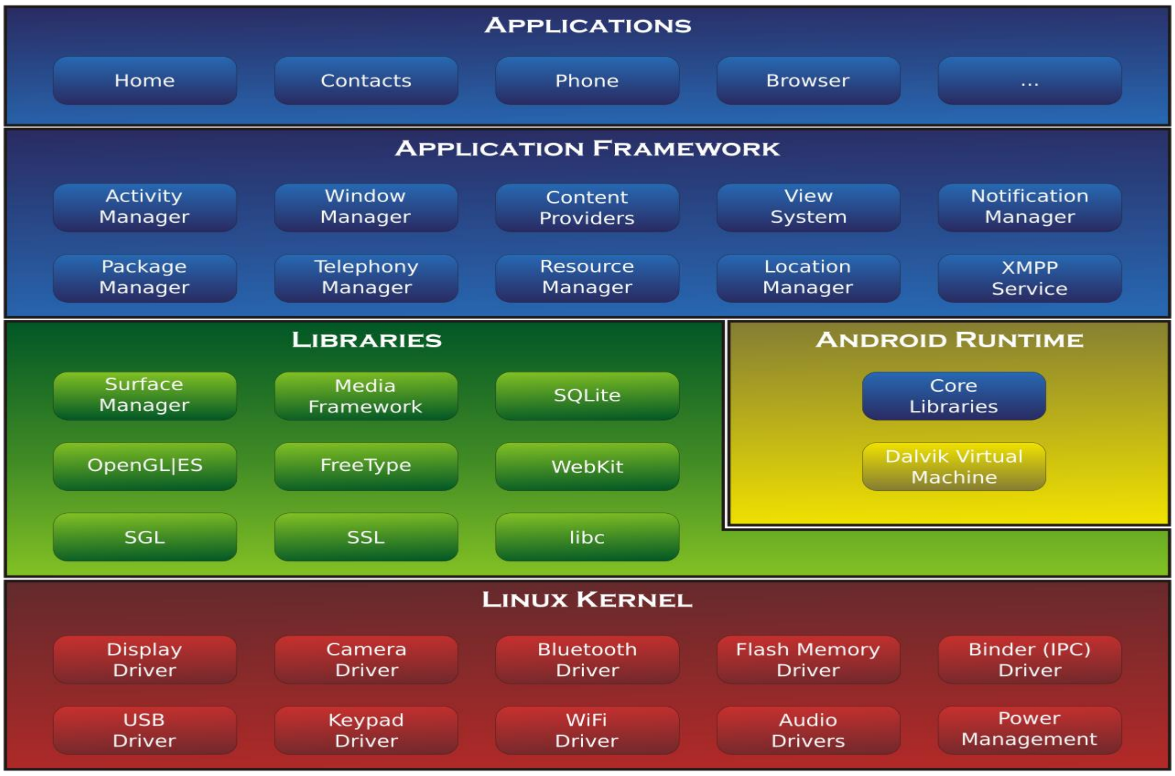


Figure 4 Android Architecture

## 1.4. SOFTWARE STACK

On top of the Linux kernel, there are the middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries. Development of the Linux kernel continues independently of Android's other source code projects.

Android uses Android Runtime (ART) as its runtime environment (introduced in version 4.4), which uses ahead-of-time (AOT) compilation to entirely compile the application bytecode into machine code upon the installation of an application. In Android 4.4, ART was an experimental feature and not enabled by default; it became the only runtime option in the next major version of Android, 5.0. In versions no longer supported, until version 5.0 when ART took over, Android previously used Dalvik as a process virtual machine with trace-based just-in-time (JIT) compilation to run Dalvik "dex-code" (Dalvik Executable), which is usually translated from the Java bytecode. Following the trace-based JIT principle, in addition to interpreting most of the application code, Dalvik performs the compilation and native execution of select frequently executed code segments ("traces") each time an application is launched. For its Java library, the Android platform uses a subset of the now discontinued Apache Harmony project. In December 2015, Google announced that the next version of Android would switch to a Java implementation based on the OpenJDK project.

## 1.5. OPEN SOURCE COMMUNITY

Android's source code is released by Google under an open source license, and its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, with a comparable level of quality; provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the HP TouchPad. Community releases often come pre-rooted and contain modifications not provided by the original vendor, such as the ability to overclock or over/undervolted the device's processor. Cyanogen MoD was the most widely used community firmware, now discontinued and succeeded by LineageOS.

## 2. JAVA

**Java** is a general-purpose programming language that is class-based, object-oriented (although not a pure object-oriented language, as it contains primitive types]), and designed to have as few implementation dependencies as possible. It is intended to let application developers *write once, run anywhere* (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Figure 5 Java logo

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Meanwhile, others have developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and Iced Tea-Web (browser plugin for applets).

# 3. JAVA JVM AND BYTECODE

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate run time support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.

Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions made interpreted programs almost always run more slowly than native executables. Just-in-time (JIT) compilers that compile byte-codes to machine code during runtime were introduced from an early stage. Java itself is platform-independent and is adapted to the particular platform it is to run on by a Java virtual machine for it, which translates the Java bytecode into the platform's machine language.

## 3.1. PERFORMANCE

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++. However, Java programs' execution speed improved significantly with the introduction of just-in-time compilation in 1997/1998 for Java 1.1, the addition of language features supporting better code analysis (such as inner classes, the StringBuilder class, optional assertions, etc.), and optimizations in the Java virtual machine, such as Hotspot becoming the default for Sun's JVM in 2000. With Java 1.5, the performance was improved with the addition of the ‘java.util.concurrent’ package, including lock free implementations of the Concurrent Maps and other multi-core collections, and it was improved further with Java 1.6.

## 3.2. AUTOMATIC MEMORY MANAGEMENT

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a non-existent object are called, a null pointer exception is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack or explicitly allocated and deallocated from the heap. In the latter case, the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable or crash. This can be partially remedied using smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent logical memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

# 4. KOTLIN

**Kotlin** is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of its standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript or native code (via LLVM). Kotlin is sponsored by JetBrains and Google through the Kotlin Foundation.

Kotlin is officially supported by Google for mobile development on Android. Since the release of Android Studio 3.0 in October 2017, Kotlin is included as an alternative to the standard Java compiler. The Android Kotlin compiler lets the user choose between targeting Java 6 or Java 8 compatible bytecode.

Kotlin has been Google's preferred language for Android app development since 7 May 2019.

Figure - 6 Kotlin logo

### 4.1. HISTORY OF KOTLIN

In July 2011, JetBrains unveiled Project Kotlin, a new language for the JVM, which had been under development for a year. JetBrains lead 'Dmitry Jemerov' said that most languages did not have the features they were looking for, with the exception of Scala. However, he cited the slow compilation time of Scala as a deficiency. One of the stated goals of Kotlin is to compile as quickly as Java. In February 2012, JetBrains open sourced the project under the Apache 2 license. The name comes from Kotlin Island, near St. Petersburg. 'Andrey Breslav' mentioned that the team decided to name it after an island just like Java was named after the Indonesian island of Java (though the programming language Java was perhaps named after the coffee).

JetBrains hopes that the new language will drive IntelliJ IDEA sales.

Kotlin v1.0 was released on 15 February 2016. This is considered to be the first officially stable release and JetBrains has committed to long-term backwards compatibility starting with this version.

At Google I/O 2017, Google announced first-class support for Kotlin on Android.

Kotlin v1.2 was released on 28 November 2017. Sharing code between JVM and JavaScript platforms feature was newly added to this release.

Kotlin v1.3 was released on 29 October 2018, bringing coroutines for asynchronous programming.

On 7 May 2019, Google announced that the Kotlin programming language is now its preferred language for Android app developers.

## 4.2. DESIGN

Development lead Andrey Breslav has said that Kotlin is designed to be an industrial-strength object-oriented language, and a "better language" than Java, but still be fully interoperable with Java code, allowing companies to make a gradual migration from Java to Kotlin. Semicolons are optional as a statement terminator; in most cases a newline is enough for the compiler to deduce that the statement has ended. Kotlin variable declarations and parameter lists have the data type come after the variable name (and with a colon separator), like Pascal. Variables in Kotlin can be immutable, declared with the 'Val' keyword, or mutable, declared with the var keyword. Class members are public by default, and classes themselves are final by default, meaning that creating a derived class is disabled unless the base class is declared with the open keyword.

In addition to the classes and methods (called member functions in Kotlin) of object-oriented programming, Kotlin also supports procedural programming with the use of functions.

Kotlin functions (and constructors) support default arguments, variable-length argument lists, named arguments and overloading by unique signature. Class member functions are virtual, i.e. dispatched based on the runtime type of the object they are called on.

# 5. INTELLIJ IDEA

**IntelliJ IDEA** is a Java integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development. The first version of IntelliJ IDEA was released in January 2001, and was one of the first available Java IDEs with advanced code navigation and code refactoring capabilities integrated. In a 2010 *InfoWorld* report, IntelliJ received the highest test centre score out of the four top Java programming tools: Eclipse, IntelliJ IDEA, NetBeans and JDeveloper. In December 2014, Google announced version 1.0 of Android Studio, an open-source IDE for Android apps, based on the open source community edition of IntelliJ IDEA. Other development environments based on IntelliJ's framework include AppCode, CLion, DataGrip, GoLand, PhpStorm, PyCharm, Rider, RubyMine, WebStorm, and MPS.

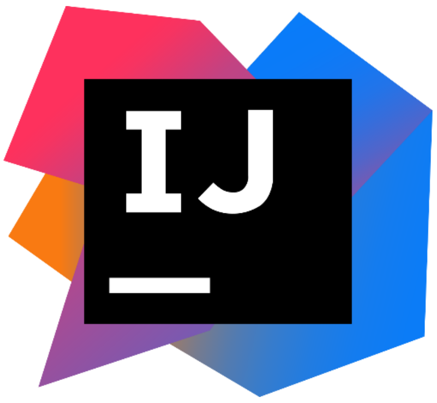


Figure - 7 IntelliJ logo

## 5.1. SYSTEM REQUIREMENT

| | **Windows** | **macOS** | **Linux** |
|---|---|---|---|
| **OS Version** | Microsoft Windows 7 SP1 or later | macOS 10.11 or later | Any 64-bit distribution that supports Gnome or KDE. Latest version of Debian, Ubuntu, or RHEL are recommended. |

| | |
|---|---|
| **RAM** | 2 GB minimum; 8 GB or more recommended for Android development and commercial production. |
| **Disk space** | 1.5 GB hard disk space + at least 1 GB for caches |
| **JDK Version** | JDK 1.8 since 2016. |
| **JRE Version** | JRE 1.8 is bundled. |
| **Screen resolution** | 1024×768 minimum screen resolution. 1920×1080 is a recommended screen resolution. |

Table 2 System Requirements IntelliJ

## 5.2. FEATURES

### 1. Coding assistance

The IDE provides certain features like code completion by analysing the context, code navigation which allows jumping to a class or declaration in the code directly, code refactoring, linting and options to fix inconsistencies via suggestions.

### 2. Built in tools and integration

The IDE provides integration with build/packaging tools like grunt, bower, gradle, and SBT. It supports version control systems like Git, Mercurial, Perforce, and SVN. Databases like Microsoft SQL Server, Oracle, PostgreSQL, SQLite and MySQL can be accessed directly from the IDE in the Ultimate edition, through an embedded version of Data Grip.

### 3. Plugin ecosystem

IntelliJ supports plugins through which one can add additional functionality to the IDE. Plugins can be downloaded and installed either from IntelliJ's plugin repository website or through the IDE's inbuilt plugin search and install feature. Each edition has separate plugin repositories, with both the Community and Ultimate editions totalling over 3000 plugins each as of 2019.

4. **Supported languages**

The Community and Ultimate editions differ in their support for various programming languages as shown in the following table. Supported in both Community and Ultimate Edition:

- Clojure (via a plugin)
- CloudSlang (via a plugin)
- Dart (via a plugin)
- Go (via a plugin)
- Groovy
- Java
- Julia (via a plugin)
- Kotlin
- Python (via a plugin)
- Rust (via a plugin)
- Scala (via a plugin)
- XML/XSL

5. **Technologies and frameworks**

Supported in both Community and Ultimate Edition:

- Android
- Ant
- Gradle
- JUnit
- JavaFX
- Maven
- Python
- SBT
- TestNG

### 6. Software versioning and revision control

The two editions also differ in their support for software versioning and revision control systems.

Supported in both Community and Ultimate Edition:

- CVS
- Git
- GitHub
- Mercurial
- Subversion

## 6. ANDROID STUDIO

**Android Studio** is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

Since 7 May 2019, Kotlin is Google's preferred language for Android app development. Still, other languages are supported, including by Android Studio.

### 6.1. FEATURES

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components

- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

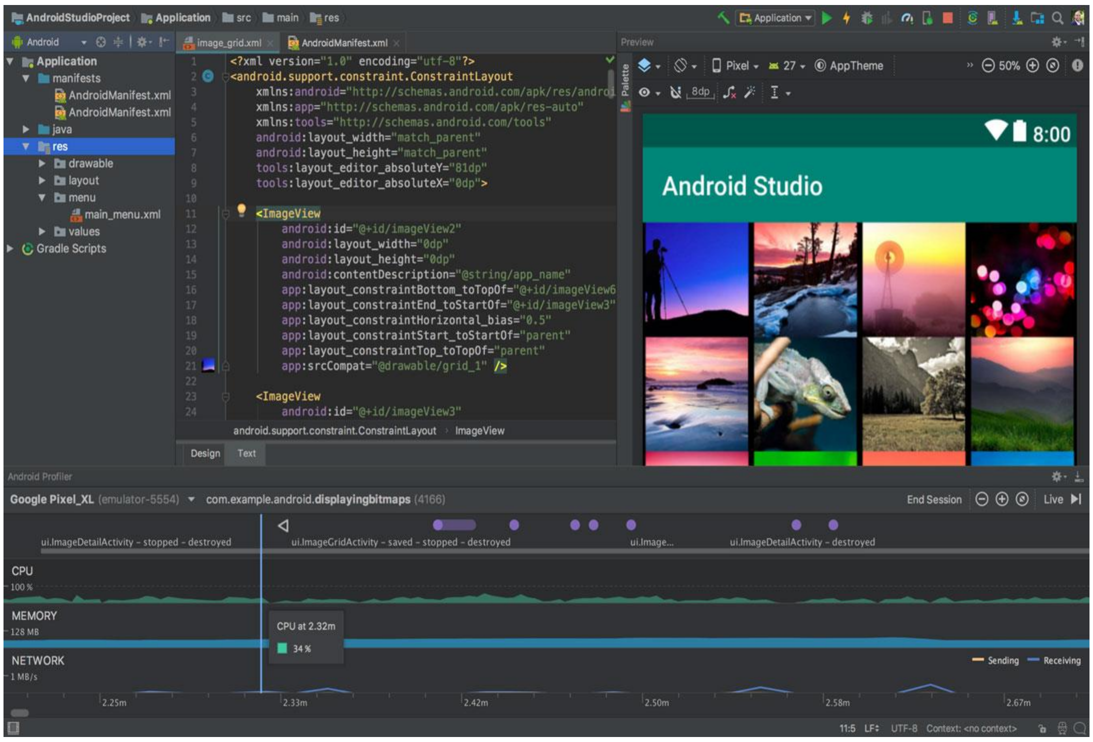
Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

# CHAPTER – 3

## 1) ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on **IntelliJ IDEA** . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for **Google Cloud Platform**, making it easy to integrate Google Cloud Messaging and App Engine



```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout_editor_absoluteY="81dp"
    tools:layout_editor_absoluteX="0dp">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:contentDescription="@string/app_name"
        app:layout_constraintBottom_toTopOf="@+id/imageView6
        app:layout_constraintEnd_toStartOf="@+id/imageView3"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/grid_1" />

    <ImageView
        android:id="@+id/imageView3"
```

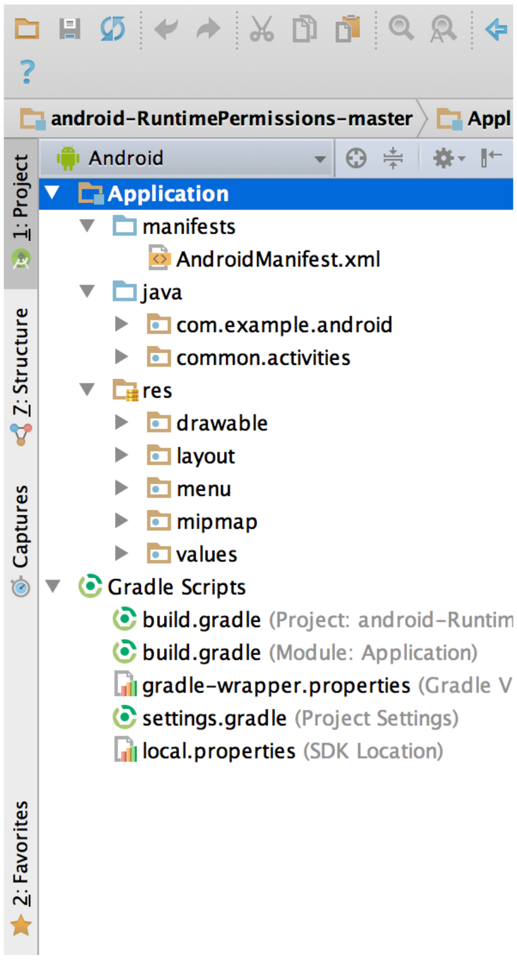
Figure - 8 Android Studio IDE Homepage

## 1.1. PROJECT STRUCTURE

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

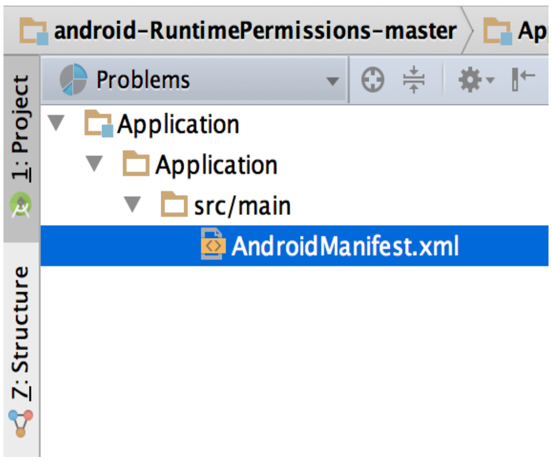
- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.



```
android-RuntimePermissions-master 〉 Appl
Android
▼ Application
    ▼ manifests
        AndroidManifest.xml
    ▼ java
        ▶ com.example.android
        ▶ common.activities
    ▼ res
        ▶ drawable
        ▶ layout
        ▶ menu
        ▶ mipmap
        ▶ values
▼ Gradle Scripts
    build.gradle (Project: android-Runtim
    build.gradle (Module: Application)
    gradle-wrapper.properties (Gradle V
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Figure - 9 The project files in Android view.



```
android-RuntimePermissions-master 〉 Ap
Problems
▼ Application
    ▼ Application
        ▼ src/main
            AndroidManifest.xml
```

Figure 10  Android Manifest

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

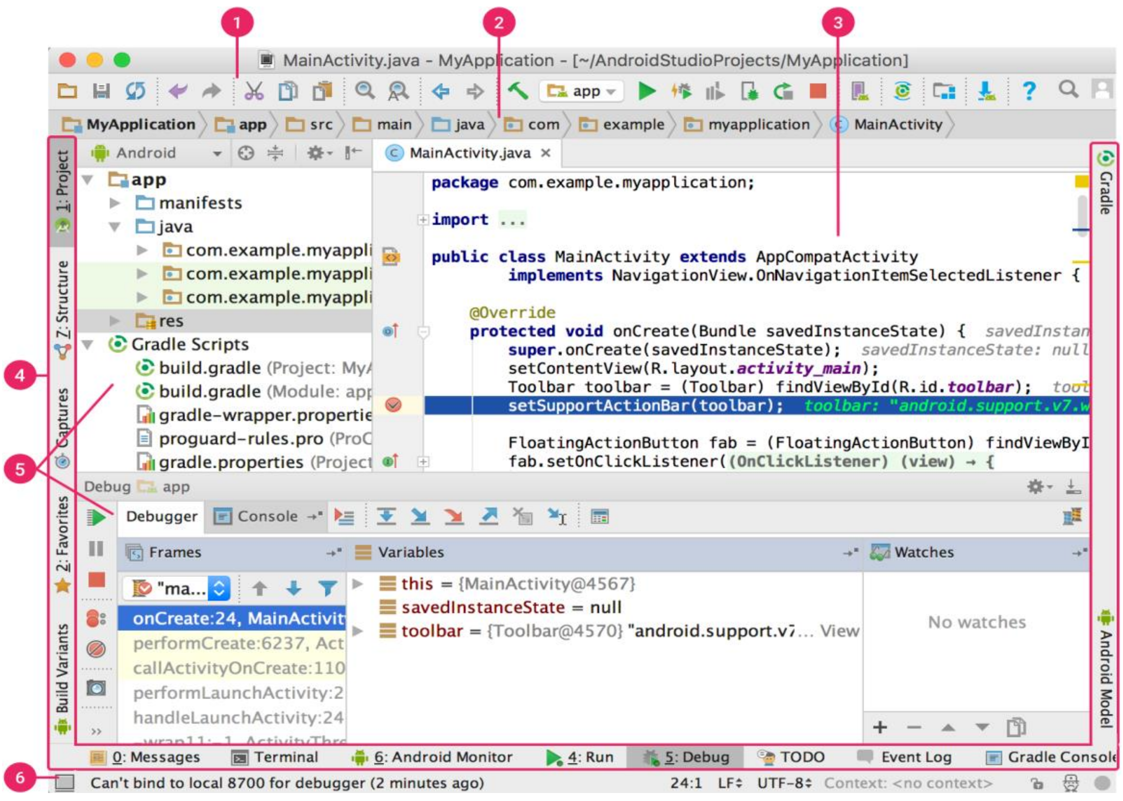
- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Java source code files, including JUnit test code.

- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**). You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

## 1.2. USER INTERFACE

The Android Studio main window is made up of several logical areas identified in figure 11.



```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {  savedInstan
        super.onCreate(savedInstanceState);  savedInstanceState: null
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  tool
        setSupportActionBar(toolbar);  toolbar: "android.support.v7.w

        FloatingActionButton fab = (FloatingActionButton) findViewByI
        fab.setOnClickListener((OnClickListener) (view) → {
```

```
onCreate:24, MainActivit
performCreate:6237, Act
callActivityOnCreate:110
performLaunchActivity:2
handleLaunchActivity:24
```

```
this = {MainActivity@4567}
savedInstanceState = null
toolbar = {Toolbar@4570} "android.support.v7... View
```

```
Can't bind to local 8700 for debugger (2 minutes ago)    24:1  LF÷  UTF-8÷  Context: <no context>
```

Figure - 11 Android User Interface

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.

3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate an IDE action that you have forgotten how to trigger.

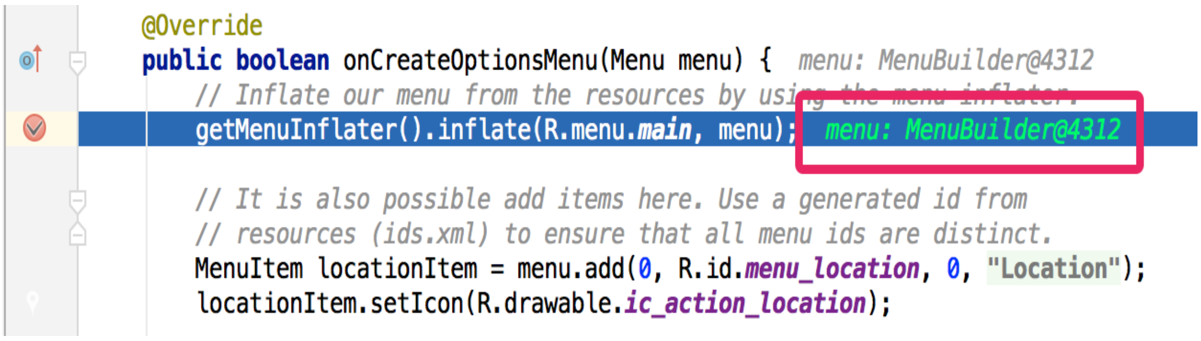
## 1.3. DEBUG AND PROFILE TOOLS

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

### Inline debugging

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

- Inline variable values
- Referring objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values



```
@Override
public boolean onCreateOptionsMenu(Menu menu) {  menu: MenuBuilder@4312
    // Inflate our menu from the resources by using the menu inflater.
    getMenuInflater().inflate(R.menu.main, menu);  menu: MenuBuilder@4312

    // It is also possible add items here. Use a generated id from
    // resources (ids.xml) to ensure that all menu ids are distinct.
    MenuItem locationItem = menu.add(0, R.id.menu_location, 0, "Location");
    locationItem.setIcon(R.drawable.ic_action_location);
```

Figure - 12 Inline Variable value

## 2) TECHNOLOGY USED IN ANDROID STUDIO

### 2.1.1. ACTIVITY AND FRAGMENTS [1]

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle. The mobile-app experience differs from its desktop counterpart in that a user's interaction with the app doesn't always begin in the same place. Instead, the user journey often begins non-deterministically. For instance, if you open an email app from your home screen, you might see a list of emails. By contrast, if you are using a social media app that then launches your email app, you might go directly to the email app's screen for composing an email.

### 2.1.1. Declare activities

To declare your activity, open your manifest file and add an <activity> element as a child of the <application> element. For example:

```
<manifest ... >
  <application ... >
     <activity android:name= “. ExampleActivity" />
     ...
  </application ... >
  ...
</manifest >
```

### 2.1.2. Declare intent filters

Intent filters are a very powerful feature of the Android platform. They provide the ability to launch an activity based not only on an *explicit* request, but also an *implicit* one. For example, an explicit request might tell the system to “Start the Send Email activity in the Gmail app". By contrast, an implicit request tells the system to “Start a Send Email screen in any activity that can do the job." When the system UI asks a user which app to use in performing a task, that’s an intent filter at work.

```
<activity android:name=".ExampleActivity" android:icon="@drawable/app_icon">
   <intent-filter>
      <action android:name="android.intent.action.SEND" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:mimeType="text/plain" />
   </intent-filter>
</activity>
```

In this example, the <action> element specifies that this activity sends data. Declaring the <category> element as `DEFAULT` enables the activity to receive launch requests. The <data> element specifies the type of data that this activity can send. The following code snippet shows how to call the activity described above:

```
val sendIntent = Intent().apply {
   action = Intent.ACTION_SEND
   type = "text/plain"
   putExtra (Intent.EXTRA_TEXT, textMessage)
}
startActivity(sendIntent)
```

### 2.1.3. Managing the activity lifecycle

Over the course of its lifetime, an activity goes through several states. You use a series of callbacks to handle transitions between states. The following sections introduce these callbacks.

- **onCreate()**

You must implement this callback, which fires when the system creates your activity. Your implementation should initialize the essential components of your activity: For example, your app should create views and bind data to lists here. Most importantly, this is where you must call setContentView() to define the layout for the activity's user interface.

When onCreate() finishes, the next callback is always onStart().

- **onStart()**

As onCreate() exits, the activity enters the Started state, and the activity becomes visible to the user. This callback contains what amounts to the activity's final preparations for coming to the foreground and becoming interactive.

- **onResume()**

The system invokes this callback just before the activity starts interacting with the user. At this point, the activity is at the top of the activity stack and captures all user input. Most of an app's core functionality is implemented in the onResume() method.

The onPause() callback always follows onResume().

- **onPause ()**

The system calls onPause() when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Recents button. When the system calls onPause() for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.

An activity in the Paused state may continue to update the UI if the user is expecting the UI to update. Examples of such an activity include one showing a navigation map screen or a media player playing. Even if such activities lose focus, the user expects their UI to continue updating.

### 2.1.4. FRAGMENTS

A Fragment represents a behaviour or a portion of user interface in a Fragment Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle. For example, when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all fragments. However, while an activity is running (it is in the *resumed* lifecycle state), you can manipulate each fragment independently, such as add or remove them. When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred. The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the *Back* button.

To create a fragment, we must create a subclass of Fragment (or an existing subclass of it). The Fragment class has code that looks a lot like an Activity. It contains callback methods similar to an activity, such as onCreate(), onStart(), onPause(), and onStop(). In fact, if you're converting an existing Android application to use fragments, you might simply move code from your activity's callback methods into the respective callback methods of your fragment.

Usually, we should implement at least the following lifecycle methods:

- **onCreate()**

  The system calls this when creating the fragment. Within your implementation, you should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.

- **onCreateView()**

  The system calls this when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

- **onPause()**

  The system calls this method as the first indication that the user is leaving the fragment (though it doesn't always mean the fragment is being destroyed). This is usually where you should commit any changes that should be persisted beyond the current user session

## NETWORK REQUEST IN ANDROID

### OKHTTP

HTTP is the way modern applications network. It's how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth.

OkHttp is an HTTP client that's efficient by default:

- HTTP/2 support allows all requests to the same host to share a socket.
- Connection pooling reduces request latency (if HTTP/2 isn't available).
- Transparent GZIP shrinks download sizes.
- Response caching avoids the network completely for repeat requests.

OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems. If your service has multiple IP addresses OkHttp will attempt alternate addresses if the first connect fails. This is necessary for IPv4+IPv6 and for services hosted in redundant data centers. OkHttp supports modern TLS features (TLS 1.3, ALPN, certificate pinning). It can be configured to fall back for broad connectivity.

Using OkHttp is easy. Its request/response API is designed with fluent builders and immutability. It supports both synchronous blocking calls and async calls with callbacks.

### Requirements

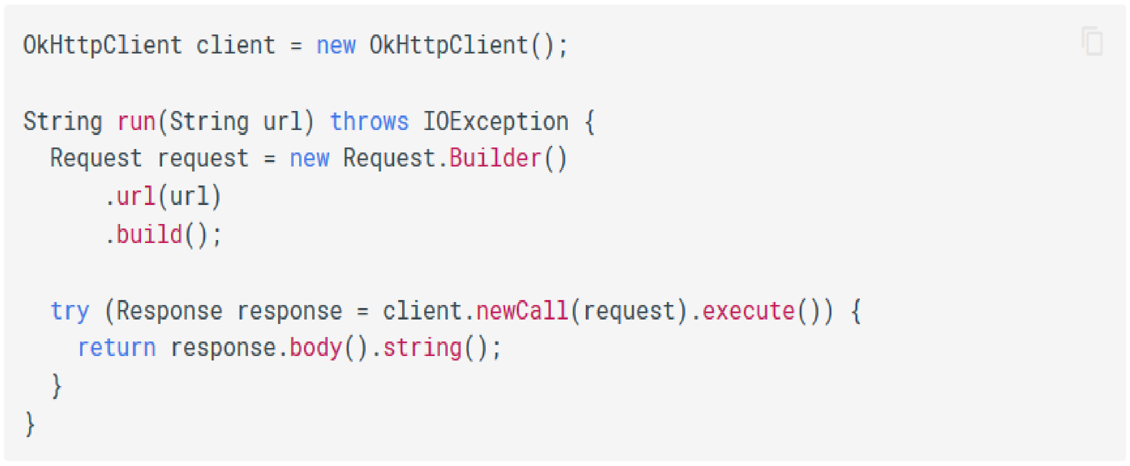
OkHttp works on Android 5.0+ (API level 21+) and on Java 8+.

OkHttp depends on Okio for high-performance I/O and the Kotlin standard library. Both are small libraries with strong backwards-compatibility.

We highly recommend you keep OkHttp up-to-date. As with auto-updating web browsers, staying current with HTTPS clients is an important defense against potential security problems. We track the dynamic TLS ecosystem and adjust OkHttp to improve connectivity and security.

The OkHttp 3.12.x branch supports Android 2.3+ (API level 9+) and Java 7+. These platforms lack support for TLS 1.2 and should not be used. But because upgrading is difficult we will backport critical fixes to the 3.12.x branch through December 31, 2020.

This program downloads a URL and prints its contents as a string.



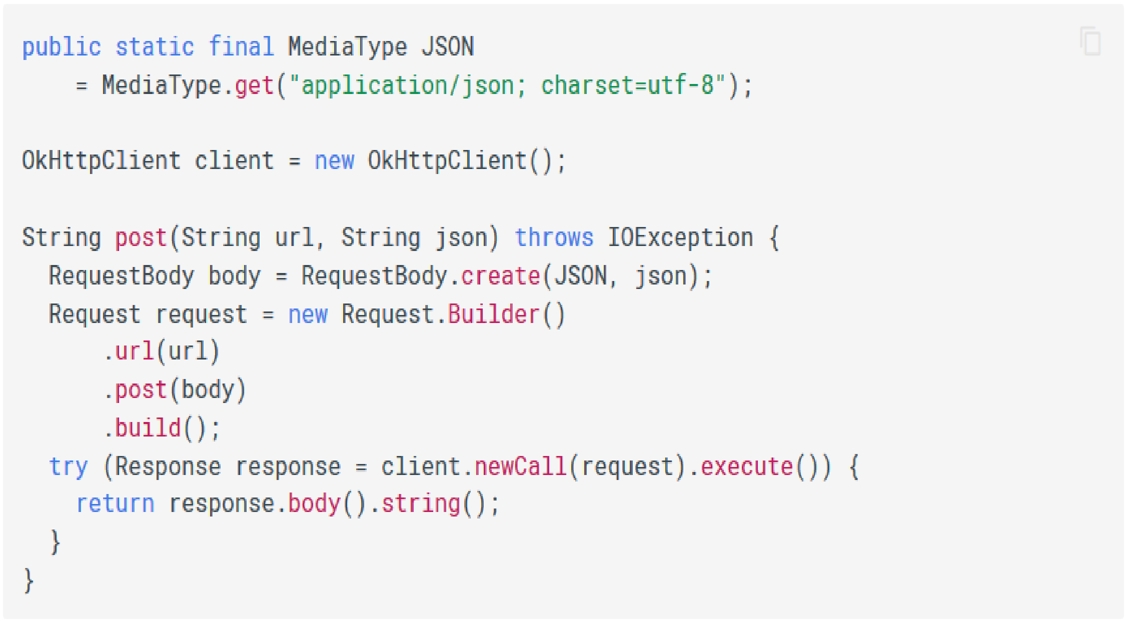
```java
OkHttpClient client = new OkHttpClient();

String run(String url) throws IOException {
  Request request = new Request.Builder()
      .url(url)
      .build();

  try (Response response = client.newCall(request).execute()) {
    return response.body().string();
  }
}
```

**Figure - 13 Get Url request in OkHttp**

This program posts data to a service.



```java
public static final MediaType JSON
    = MediaType.get("application/json; charset=utf-8");

OkHttpClient client = new OkHttpClient();

String post(String url, String json) throws IOException {
  RequestBody body = RequestBody.create(JSON, json);
  Request request = new Request.Builder()
      .url(url)
      .post(body)
      .build();
  try (Response response = client.newCall(request).execute()) {
    return response.body().string();
  }
}
```

Figure 14 Post a request to server using OkHttp

## RETROFIT

**Retrofit** is a REST Client library (Helper Library) used in Android and Java to create an HTTP request and also to process the HTTP response from a REST API. It was created by Square, you can also use retrofit to receive data structures other than JSON, for example SimpleXML and Jackson. Before we continue, let's briefly define REST Client and REST API in our context.

**REST Client** in our case is the Retrofit library that is used on the client side (Android) to make HTTP request to REST API, in our case, The Movie DB API and process the response.

**A REST API** defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST. in our case, The Movie DB (TMDB) API is the REST API.

We can also simply say that a RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

**To use Retrofit in your Android Application, you'll need 3 major classes.**

1. **An Interface which defines the HTTP operations (Functions or methods)**

According to Square, creators of Retrofit documentation, Retrofit turns your HTTP API into a Java interface. Sample codes for the interface and the method declared in it are as below:

Every method inside an interface represents one possible API call. It must have a HTTP annotation (GET, POST, etc.) to specify the request type and the relative URL. The return value wraps the response in a Call object with the type of the expected result.

Query parameters can also be added to a method.

2. **A Retrofit class** which generates an implementation of the GitHubService interface. The below sample code would be inside the Retrofit class and this is how it creates an instance of Retrofit and implements the listRepos() method that's in the GitHubService Interface.
3. **The last of the 3 needed class is a simple POJO** that matches each field in the JSON response object gotten from querying an API. It's a simple class with getter and setter methods for each fields.

**Retrofit Converters**

Retrofit Converters are like an agreement between and Android client and the Server on the format on which data will be represented. Both parties can agree that for our communication, the format for data transfer will be JSON, as in our case in this tutorial. Remember i said apart from the JSON structure converter, we have others and here are some supported by Retrofit.

1. **Gson:**

Gson is for JSON mapping and can be added with the following dependency:

compile 'com.squareup.retrofit2:converter-gson:2.2.0'

2. **SimpleXML**

SimpleXML is for XML mapping. You'll need the following line for your build.gradle:

compile 'com.squareup.retrofit2:converter-simplexml:2.2.0'

3. **Jackson**

Jackson is an alternative to Gson and claims to be faster in mapping JSON data. The setup offers you a lot more customization and might be worth a look. You can add it with:

compile ‘com.squareup.retrofit2:converter-jackson:2.2.0’

4. **Moshi**

Moshi is another alternative to Gson. It’s created by the developers of Retrofit. Moshi is based on Gson but differentiates itself with some simplifications. If you want to give this young new player on the market a try, add it with:

compile ‘com.squareup.retrofit2:converter-moshi:2.2.0’

# CHAPTER – 4

## INSIGHTS OF THE PROJECT

### SCREENSHOTS

This section shows the Popular movies. In the Movies Activity we have 4 fragments namely Now Showing, Upcoming , Top Rated and Popular. I have added a view pager with a Tabbed Layout to achieve this.

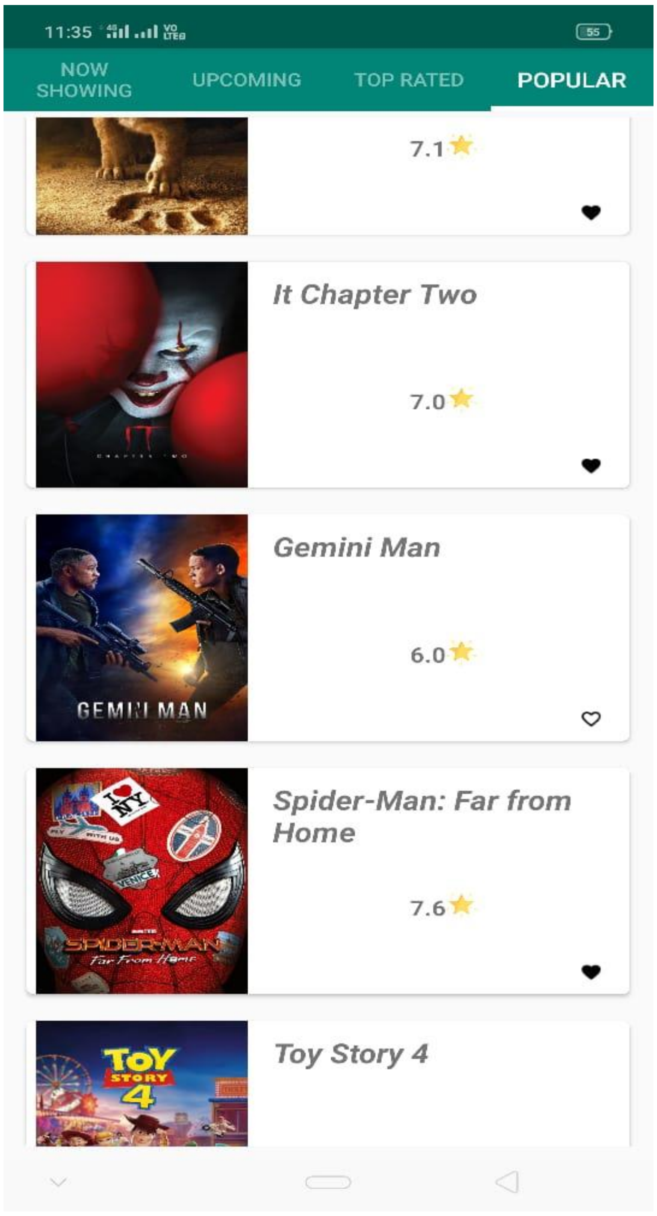


*Figure 15 Popular movies*

This is the main home page of the Movie section as displayed in the figure 16

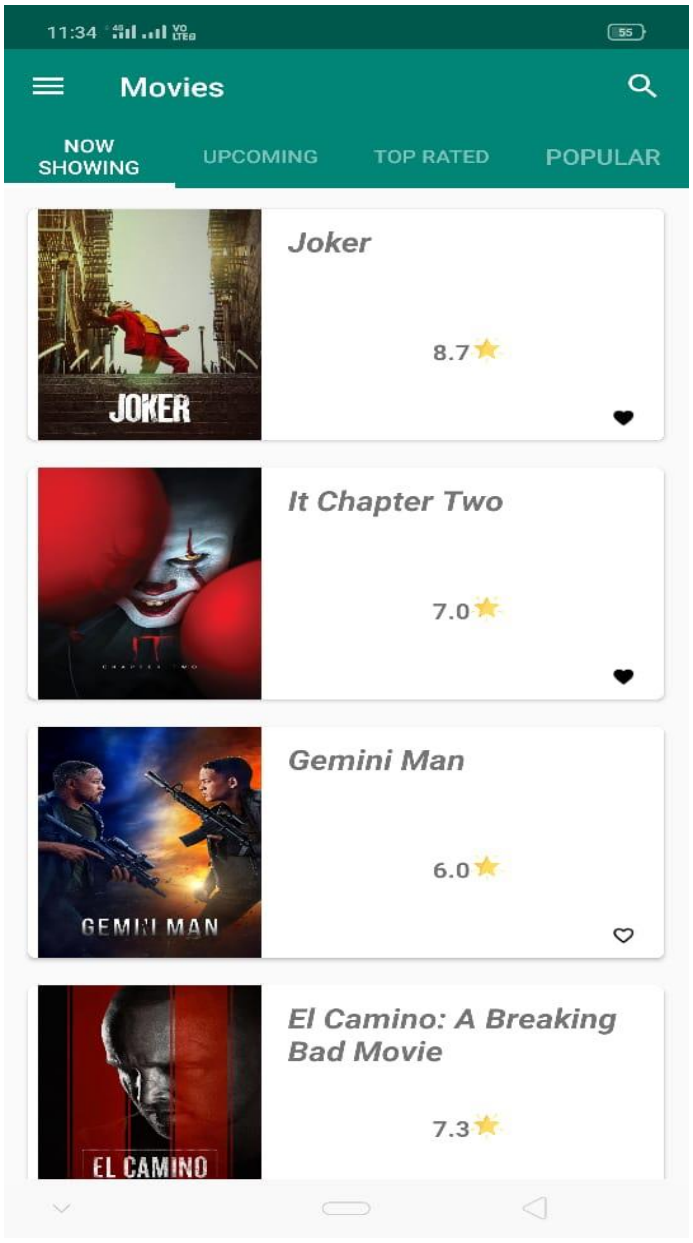
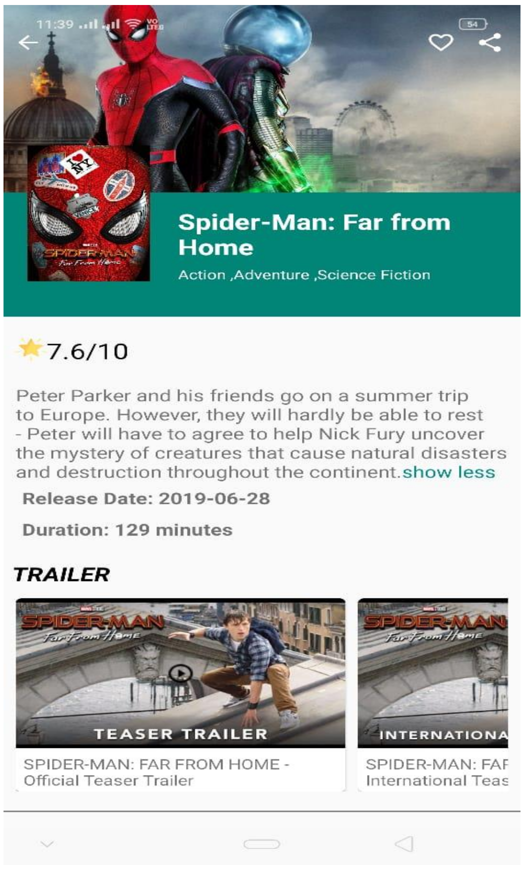


*Figure 16 Movie Section*

The figure 17 displays the Movie Detail Activity. It shows the datils of the movie like the description, running time, release date etc. Also, it shows the Trailers and the casts of the movie.

The Figure 18 shows the cast the given movie and the list of similar movies.


11:39

Spider-Man: Far from Home

Action ,Adventure ,Science Fiction

7.6/10

Peter Parker and his friends go on a summer trip to Europe. However, they will hardly be able to rest - Peter will have to agree to help Nick Fury uncover the mystery of creatures that cause natural disasters and destruction throughout the continent.show less

Release Date: 2019-06-28

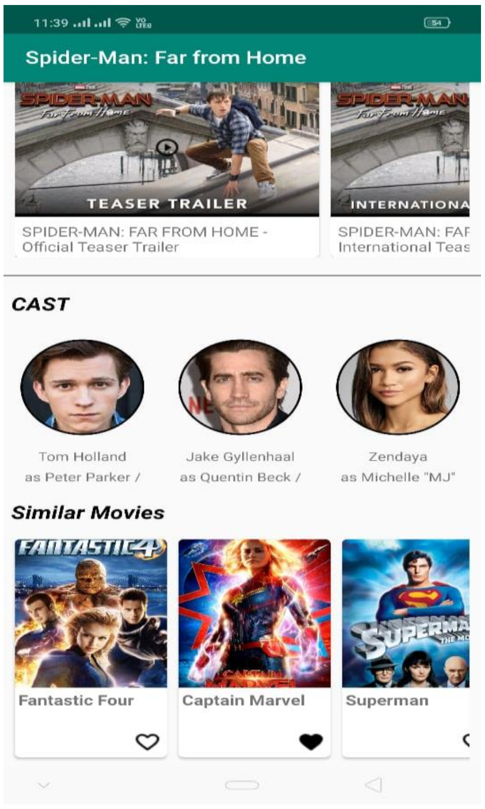
Duration: 129 minutes

TRAILER

SPIDER-MAN: FAR FROM HOME - Official Teaser Trailer

SPIDER-MAN: FAR International Teas

Figure 17 Movie Detail


11:39

Spider-Man: Far from Home

SPIDER-MAN: FAR FROM HOME - Official Teaser Trailer

SPIDER-MAN: FAR International Teas

CAST

| | | |
|---|---|---|
| Tom Holland | Jake Gyllenhaal | Zendaya |
| as Peter Parker / | as Quentin Beck / | as Michelle "MJ" |

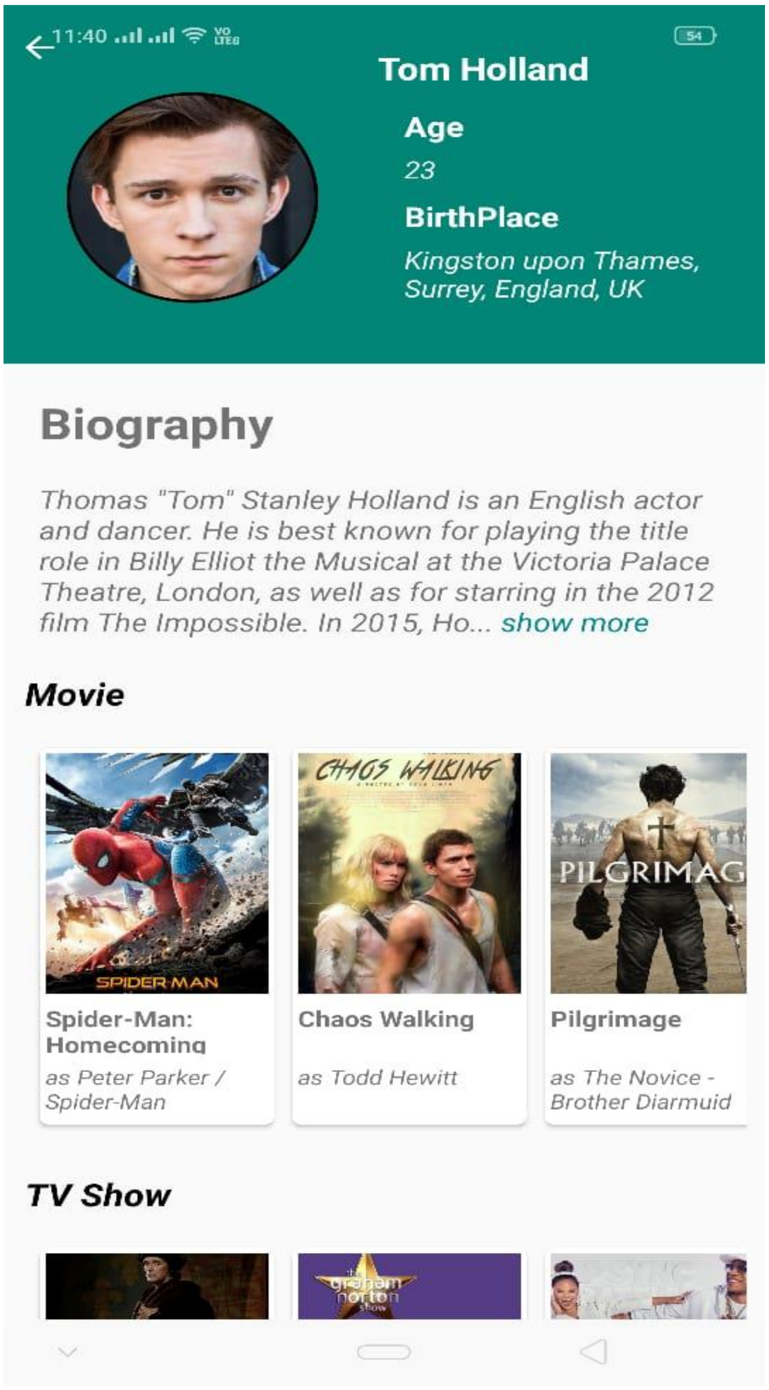
Similar Movies

Fantastic Four

Captain Marvel

Superman

Figure 18 Movie detail cast


11:40

Tom Holland

Age

23

BirthPlace

Kingston upon Thames, Surrey, England, UK

Biography

Thomas "Tom" Stanley Holland is an English actor and dancer. He is best known for playing the title role in Billy Elliot the Musical at the Victoria Palace Theatre, London, as well as for starring in the 2012 film The Impossible. In 2015, Ho... show more

Movie

Spider-Man: Homecoming
as Peter Parker / Spider-Man

Chaos Walking
as Todd Hewitt

Pilgrimage
as The Novice - Brother Diarmuid

TV Show

Figure 19 Cast

Figure 19 shows the Cast Details. This activity tells about the history, career, and all the details about the cast. There is also a list of movies and Tv shows showing all the movies and tv shows the cast has performed.

This figure shows the search functionality in the app. Added a search button which on clicking opens the search dialog. This search dialog also shows search suggestion based on the previous searches. I have also implemented a voice search functionality which I shown in figure 21.

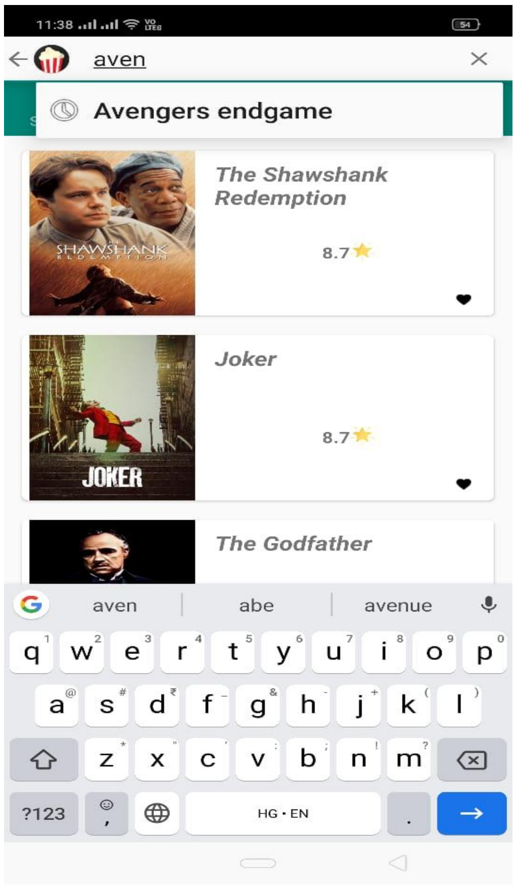
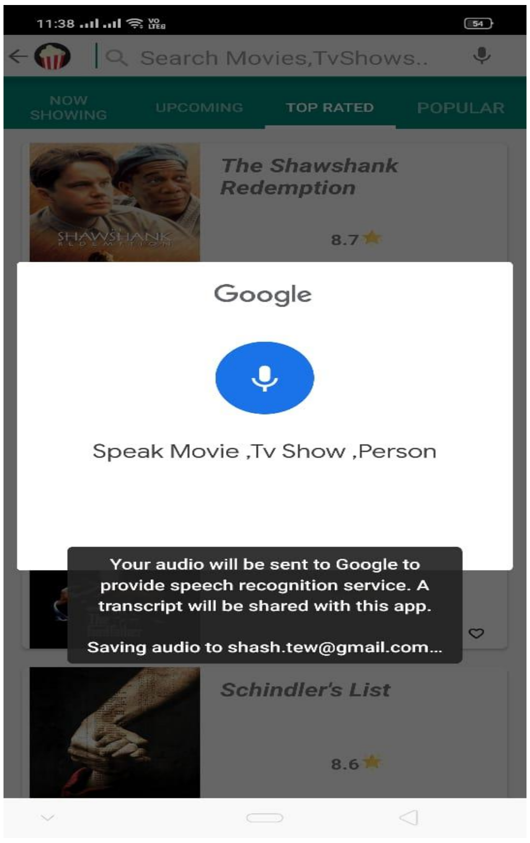


Figure 20 search suggestion



Your audio will be sent to Google to provide speech recognition service. A transcript will be shared with this app.

Saving audio to shash.tew@gmail.com...

Figure 21 voice search

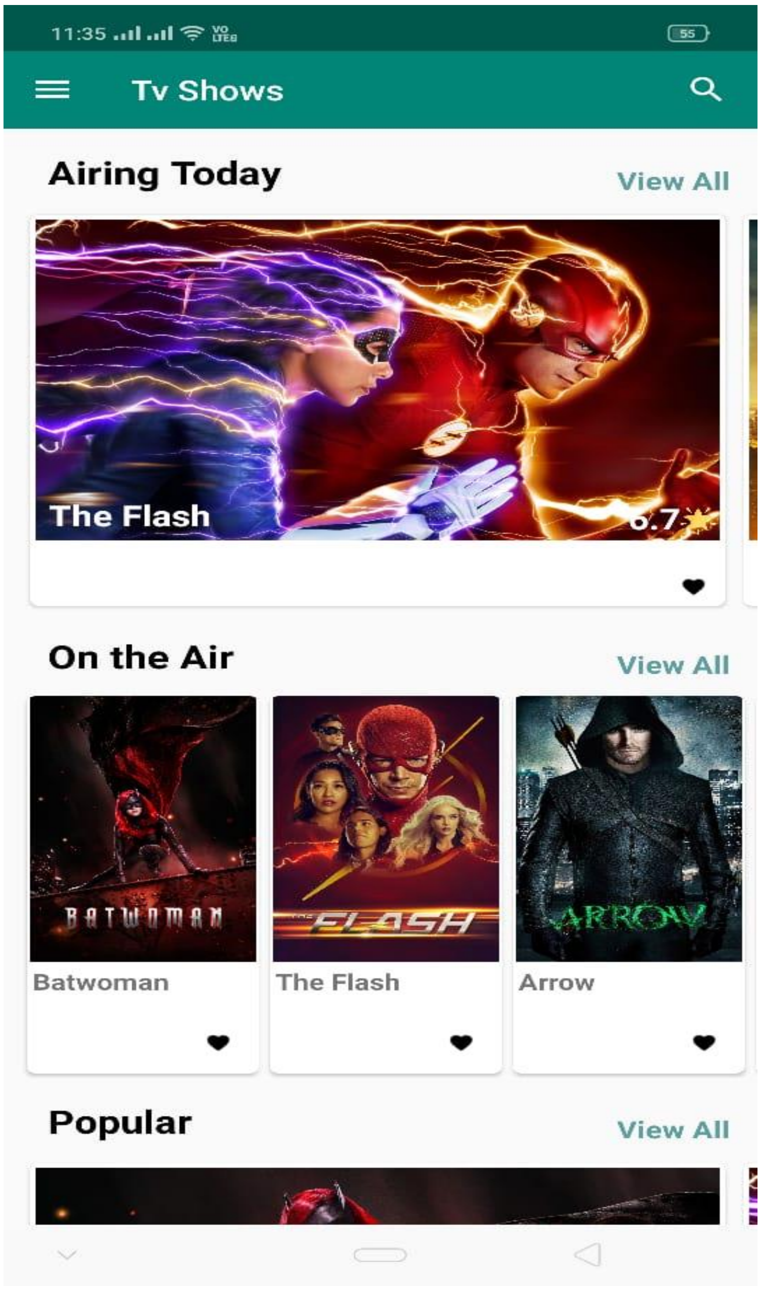


*Figure 22 Tv show*

This figure shows the Home page of the Tv Shows. This is divided into four categories namely Airing Today , On the air , popular and top rated. There is too a view all button which will open up all the list of shows for that category.

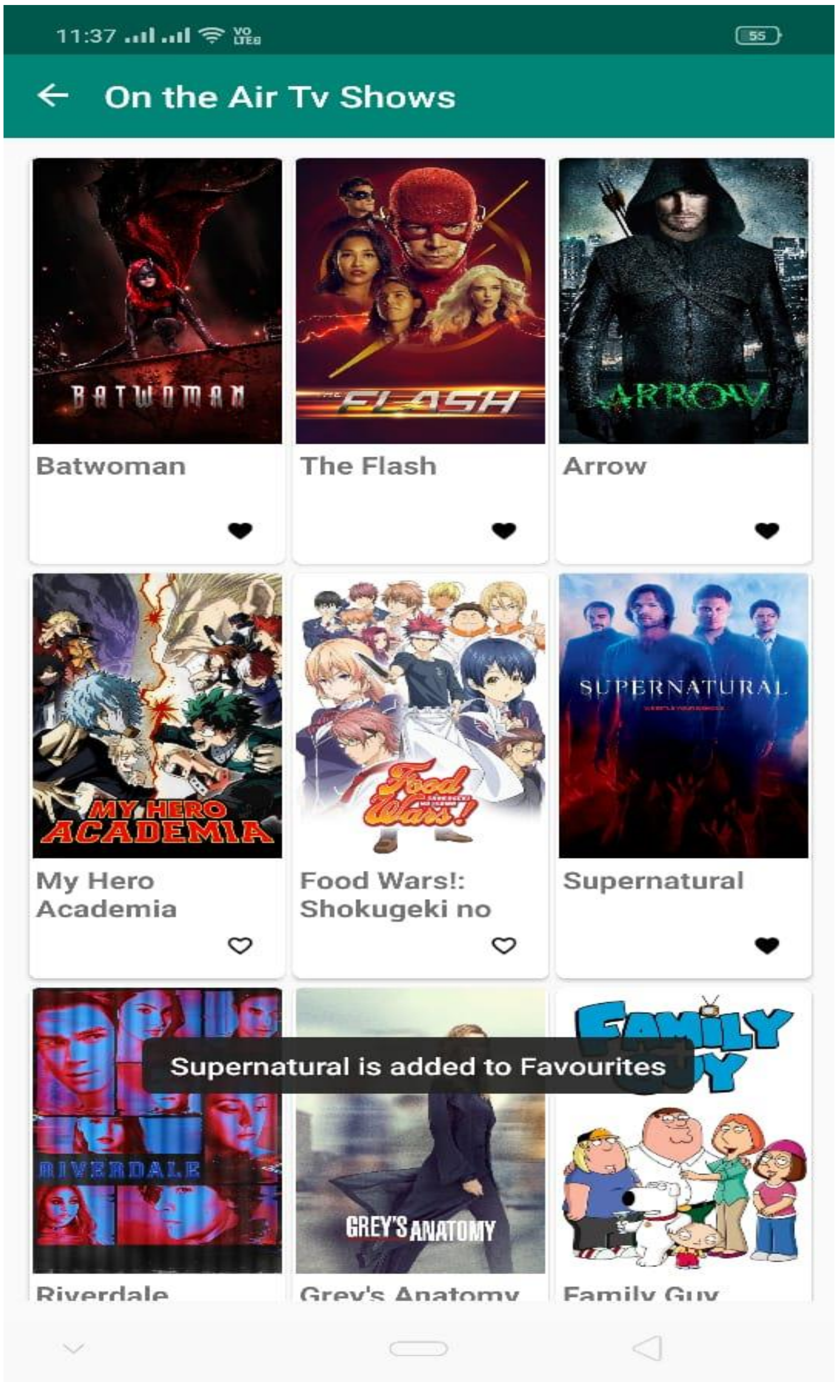


Figure 23 View all Activity

Figure 23 displays the view all activity. So it shows all the On the air Tv shows present in the Grid List form. I have also shown how to add the show/movie to favourites here.

Just click on the heart button and the show/movie will be added to the favourites.

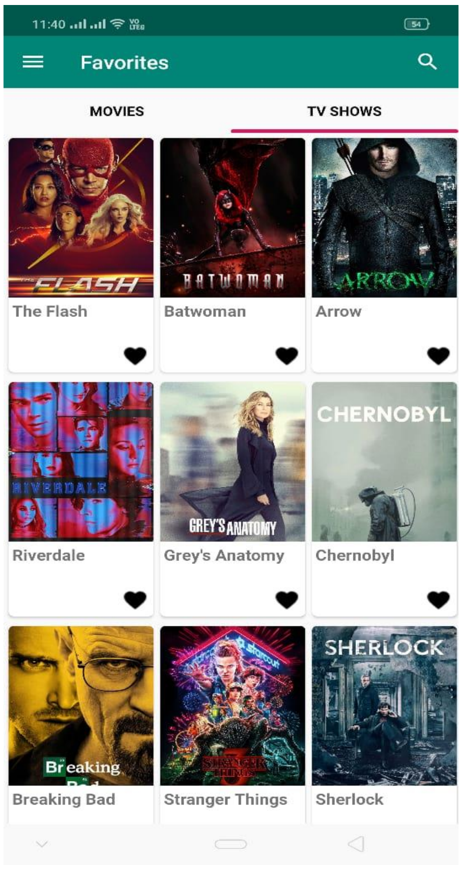
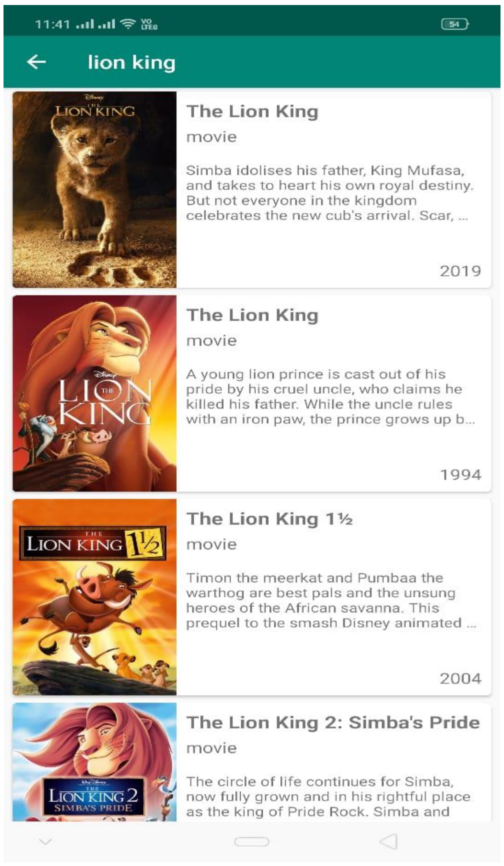


Figure 24 Favourite



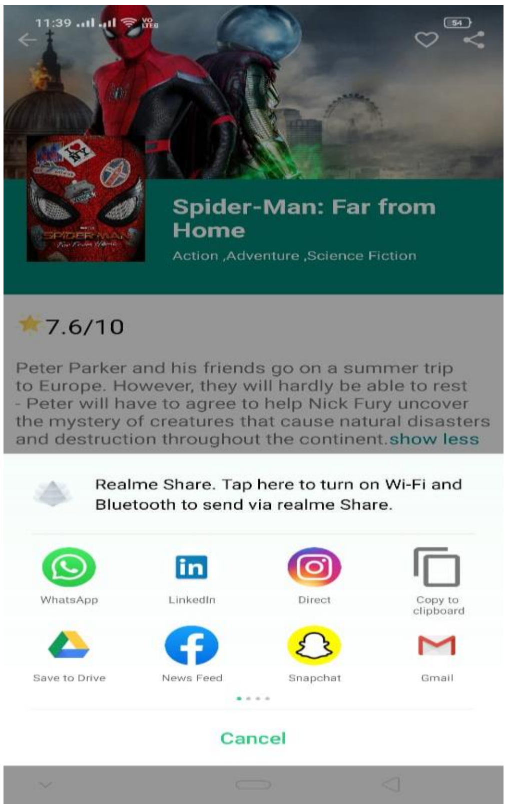
The Lion King
movie
Simba idolises his father, King Mufasa, and takes to heart his own royal destiny. But not everyone in the kingdom celebrates the new cub's arrival. Scar, ...
2019

The Lion King
movie
A young lion prince is cast out of his pride by his cruel uncle, who claims he killed his father. While the uncle rules with an iron paw, the prince grows up b...
1994

The Lion King 1½
movie
Timon the meerkat and Pumbaa the warthog are best pals and the unsung heroes of the African savanna. This prequel to the smash Disney animated ...
2004

The Lion King 2: Simba's Pride
movie
The circle of life continues for Simba, now fully grown and in his rightful place as the king of Pride Rock. Simba and

Figure 25 search result


Spider-Man: Far from Home

Action ,Adventure ,Science Fiction

7.6/10

Peter Parker and his friends go on a summer trip to Europe. However, they will hardly be able to rest - Peter will have to agree to help Nick Fury uncover the mystery of creatures that cause natural disasters and destruction throughout the continent.show less

Realme Share. Tap here to turn on Wi-Fi and Bluetooth to send via realme Share.

*Figure 26 Share functionality*

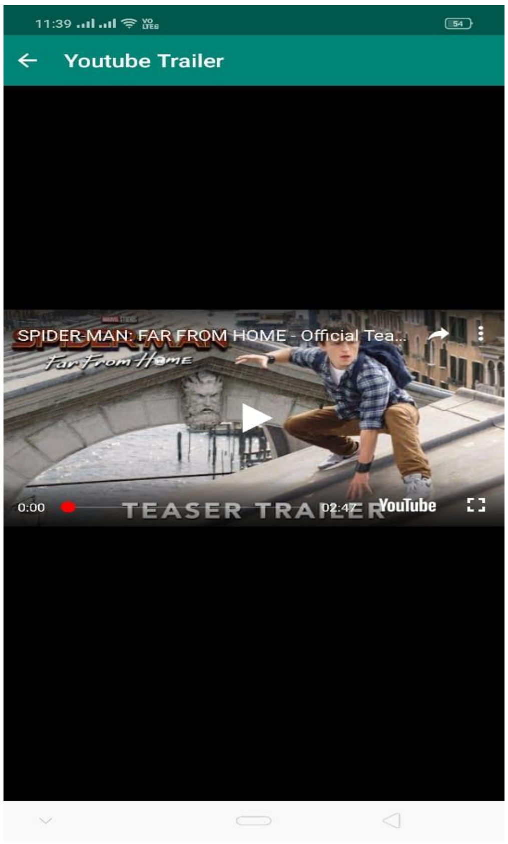


*Figure 27 YouTube in App*

# CHAPTER -5

## CONCLUSION

The current work is the initial background report for the Application development Project. The report tells us about the background of the project and give the insights of the project made. This report is the summary of my work done on the project. This report makes us aware about the technology used in development. The project which is an application made on the best application development platform namely Android Studio. Android Studio is the most powerful and suitable tool for the application development for android. My app namely Movie-DB is the end project made during the training. I have implemented almost all the technology that I learned during my training. I used Kotlin as my language for writing all the code basically logic and Xml as a language for the user Interface part. My application Movie-Db fetches all its data from the internet Using network calls. I have used IMDB api for fetching the contents of the application. The work done by me is solely based for the sake of learning. My app shows us the latest movie, tv shows and their details. We will get all the movies and tv shows on this app and we can know its detail. I have also added a functionality of trailer which means that we can play trailer for that movie/show inside the app (instead of going to YouTube).  The end goal of this app is learning and moreover I can publish my app on playstore, but as publishing on playstore requires membership charges so I have made my app Open Sourced under Apache 2.0 License.

# Bibliography

[1] G. Osahon, “AndroidPub- Consuming Rest Api,” 19 March 2017. [Online]. Available: https://android.jlelse.eu/consuming-rest-api-using-retrofit-library-in-android-ed47aef01ecb.

[2] G. Developers, “Introduction to Activities,” 2018. [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities.

[3] Wikipedia, “Android Operating System,” Wikimedia Foundation, [Online]. Available: https://en.wikipedia.org/wiki/Android_(operating_system). [Accessed August 2019].

[4] C. Blocks, “Coding Block - About Us,” [Online]. Available: https://codingblocks.com/about.html. [Accessed August 2019].

[5] I. Square, “OkHttp,” [Online]. Available: https://square.github.io/okhttp/. [Accessed September 2019].