PLACEMENT APPLICATION
OF
DEPARTMENT OF COMPUTER SCIENCE

MAJOR PROJECT REPORT

SEMESTER IV

M.SC. COMPUTER SCIENCE

(2017-2019)

Ayush Malik (1723907)

Rana Ibrahimi (1723926)

Shivani Tiwary (1723934)

Under the Supervision of

Mr. P.K. HAZRA

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF DELHI
NEW DELHI

CERTIFICATE

This is to certify that the project entitled "**Placement Application of Department Of Computer Science**" is submitted by **Ayush Malik**, **Rana Ibrahimi** and **Shivani Tiwary**, M.Sc. Computer Science, Semester IV, as a Major Project work to the Department of Computer Science, University of Delhi, Delhi, India, under the supervision of **Mr. P. K. Hazra** for the partial fulfillment for the award of the degree of Masters of Science in Computer Science. To the best of our knowledge and belief, this work has not been submitted in part or full to any other University or Institution for the award of any degree or diploma.

Ayush Malik	**Rana Ibrahimi**	**Shivani Tiwary**
(1723907)	(1723926)	(1723934)

Supervisor
Mr. P. K. Hazra
Associate Professor
Department Of Computer Science
University Of Delhi, India

Head of Department
Prof. Vasudha Bhatnagar
Professor
Department Of Computer Science
University Of Delhi, India

ACKNOWLEDGEMENT

We are thankful to our supervisor, **Mr. P.K. Hazra**, Associate Professor, Department of Computer Science, University of Delhi, for his immense support and guidance. We feel honored to have worked under him. He has guided us at each step with his expertise.

We want to thank **Dr. Sushila Madan** for her constant guidance.

We also wish to express our gratitude to the Head of Department, **Prof. Vasudha Bhatnagar** for facilitating us to carry out this project work in the department.

We wish to extend our sincere regards to the laboratory staff of the Department of Computer Science for their help.

We are also thankful to our family members and our friends for their constant moral support and help.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Problem Description

The problem considered in this project is that of simplifying the process of organizing several campus placement drives.

In the Department of Computer Science, each year 6 student *Placement Coordinators (PCs)* are selected to assist and organize the placement drives. These student coordinators are involved throughout the process which involves sending invites to companies, negotiating dates, maintaining a student-company database, arranging for company requirements, asking for permission, answering student queries, marking student attendance, and a lot more. It is a very challenging and time-consuming task for these Placement Coordinators. Not only do they have to manage the placement tasks but they also have to prepare for their own interviews. After some long discussions with the Placement Coordinators about their experiences, in the following paragraphs we have described some of the challenges and problems faced by them:

- One of the most important problems faced by the Placement Coordinators is *mail handling*. The majority of the conversations of the PCs involve e-mails. These include companies, students as well as faculty members. They have to send several e-mails having very similar content and format. They are required to re-compose the mails again and again.
- Every year, new students are selected as PCs. They are guided by their seniors and loads of previous data including that of companies. Apart from this, they also have to collect data from the students. This is used for sending information to the companies. PCs also have to *maintain data* associated with the selection of the students. Till now,

all this data has been collected using a number of spreadsheets. These are difficult to manage and maintain.

- As already stated, all official communication regarding placements uses e-mails. This includes all the information sent by the Placement Coordinators to the students. It has been observed that many students *aren't regular in checking their mail*. This results in delays in the communication of important messages and information.
- Not all students sit in all placement drives. It has been found that some students take *undue advantage of placement events.* They bunk classes during placement drives even when they are not being involved in one.
- Currently, all unplaced students are expected to report for all placement drives. There is *no formal registration mechanism*. The PCs have to submit the data much before the drive. Since there are no registrations, it has been found that there is a lot of difference between the data of the students given to the company and the students who turn up for the drive. This results in a system with less accountability and more confusion.
- Every year some fund is collected by the Placement Coordinators for conducting and managing campus placements. It has been found that the current methods used for *fund management* are not very effective and they don't offer sufficient transparency.

In the following section, we have tried to address all these issues and proposed a solution for them.

1.2 Solution

We have developed an Android Application which we call *'Placement Application of Department Of Computer Science'*.

1.2.1 User Characteristics

The application caters to the needs of 3 types of users/ roles:

- *Placement Coordinators (PCs)*: These are the students who are selected to handle and organize the entire placement process.
- *Faculty*: The members of the faculty include all the teaching staff of the Department of Computer Science, University of Delhi.
- *Students:* These are all the students who apply for placements. These include the students of M.Sc. 2nd Year and M.C.A 3rd Year along with the Placement Coordinators.

There will be one more user called the *Super User***.** The Super User will be responsible for the maintenance of the application (database maintenance, server maintenance, new registrations, and other system administration work). This role can be performed by one of the PCs or by some member of the Department of Computer Science's administrative staff.

1.2.2 Functional Requirements

The functional requirements of the application try to handle all the above- mentioned challenges along with some additional useful features. These have been specified below:

1. *Login and Authentication*

   Each type of user will be offered different functionalities. The Super User is going to store some e-mail addresses with their corresponding roles in the database. Only the users with the stored e-mail addresses will be allowed to sign up for the application.

These e-mail addresses will be verified during sign up. Only people with valid authentication details (e-mail and password) will be able to access the application. The functionalities available to a user will be decided by his/ her login details.

2. *Simplified Mail Handling*

   This will be a feature only for PCs. This will simplify 3 types of repetitive e-mails sent by the PCs:

   - Invitation Mails: Mails sent to different companies inviting them for placement drive and sharing with them the details such as batch strength, and placement coordinators' contacts.
   - Faculty Permission: These are e-mails sent to the faculty before each placement drive. This informs the faculty about the upcoming placement drives and requests their permission for using different rooms.
   - Student Information: These are e-mails sent to students giving them details of the upcoming company. These include job profile, job description, package offered, job location, date and venue of drive, etc.

   All these 3 e-mails are sent many times for different companies. The application will allow the PCs to generate these e-mails by accepting minimal inputs from them. It will automatically generate the content of the e-mails and insert the input at relevant places, thus simplifying the task of sending e-mails.

3. *Artificial Intelligent Hiring Predictor*

   This functionality will be available to students only. The application will study the old data of students and whether or not they were hired. By applying artificial intelligence techniques on this data, the application will try to predict the chance of a

particular student getting hired in the upcoming placement drives.

4. *Live Location Sharing*

   During the time of a placement drive, the faculty/ placement coordinator will be able to view the live location of each student. This will help in preventing the students from taking undue advantage of placement drives.

5. *Real-Time Database*

   The database will be maintained on a server. Changes made to the database will be updated/ synchronized across all the devices using the application. The synchronization will only take place if the device is connected to the Internet. Users will have pre-defined read and write functions that they can perform on the database.

6. *GPS based attendance*

   The Placement Coordinators will take attendance for each placement event. During a placement event, one of the PCs will mark a geographical area around the premises of the placement event. When a student appearing for that placement drive will enter the region specified by the PC, he/she will be able to mark his/her attendance. A PC will also be able to control the time period for which a student will be able to mark his/ her attendance. Whenever a PC will enable or disable attendance for a drive, a notification will be sent to all users.

7. *Student Registrations for Placement Drives*

   Each student has to individually register for each placement event for which he/she wishes to apply. The application will allow the eligible students to register for the placement drive after the registrations for that drive have been opened by a Placement Coordinator. A notification will be sent to all the users whenever

registrations are opened or closed.

8. *Material Design*

   The entire graphical user-interface will follow the material design theme to make the application more visually appealing. The application's design will be consistent with Google's material design guidelines.

9. *Announcements and Notifications*

   The Placement Coordinators will be able to send customized information as an 'Announcement'. Whenever an announcement will be made a new notification will be sent to the receivers (faculty and students).

10. *Announcement Log*

    This functionality will be provided to all 3 types of users. Each new announcement made by the placement coordinators will be stored and logged in the database. All the users will be able to view the previously made announcements.

11. *Expenditure*

    The Placement Coordinators will be able to add the details about how much money was spent, on which date and for what purpose. The students and faculty members will be able to view these expenditures. This will help in ensuring accountability and transparency.

12. *Student Data Queries*

    Placement Coordinators will be able to view each student's data directly from the application. Apart from personal details, contact details, and academic background, this data will also include information about the list of the companies for which a student

registered, the drives which he/she attended and the companies in which he/she got selected. A faculty member will also have access to some portion of this data. A student will be able to view his/her personal data.

13. *Company Data Queries*

    Placement Coordinators will be able to view each company's entire data directly from the application. A company's data will include information about the list of students who registered for its drive, the list of students who attended its drive and the list of the ones who got selected.

14. *Adding Company and Student Records*

    Placement Coordinators will be able to add new companies to the database. Each student will need to add his/her information into the database during registration.

2 . TOOLS AND TECHNOLOGIES

This section briefly describes the platform on which the application is built, that is, Android. It also discusses the basic pre-requisites used for designing an Android application along with other major tools, technologies, libraries and Application Program Interfaces (APIs).

Designing any Android application has 2 pre-requisites: Java and XML.

2.1 Java

Java is a general-purpose object-oriented computer-programming language (except the primitive data types, all elements in Java are objects) created by Sun Microsystems in 1991.

Java is:

- Concurrent.
- Class-based and an object-oriented programming language.
- An Independent programming language that follows the logic of "Write once, Run anywhere". The compiled code can run on all platforms which support Java.

Object-Oriented Programming

Object-oriented programming (OOPs) is a property of Java, that simplifies software development and maintenance by providing some rules. Programs are organized around objects rather than action and logic. Understanding the working of the program becomes easier, as OOPs bring data and its behavior (methods) into a single (objects) location.

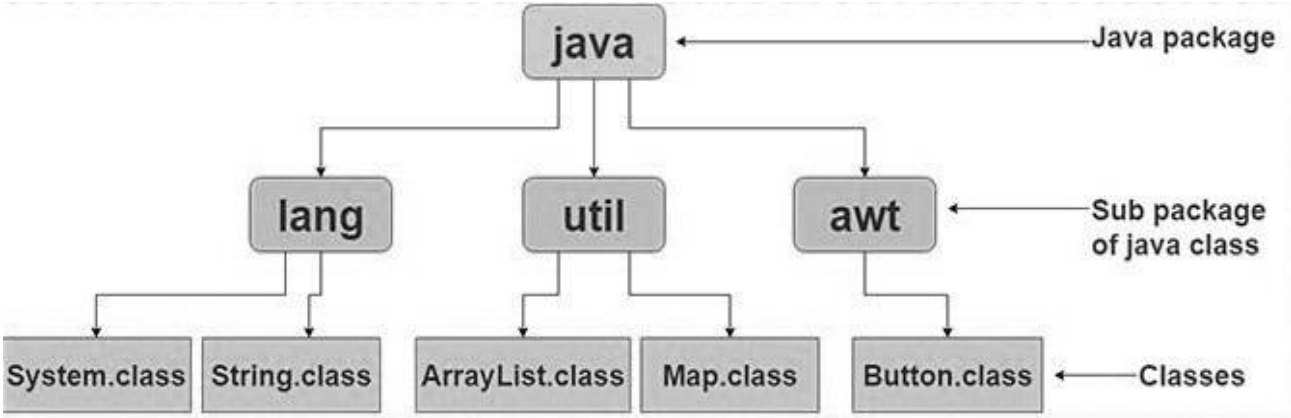
Basic concepts of OOPs are:

- *Objec*t: A basic unit of OOPs and represents the real-life entities.
- *Class:* User-defined prototype from which objects are created.
- *Inheritance:* Used to provide the concept of code-reusability.
- *Polymorphism:* To perform different tasks in different instances.
- *Encapsulation:* Makes data abstraction (privacy to data) possible.
- *Abstraction:* Representing essential features without including the background details.

Packages in Java

A Java package is a group of similar types of classes, interfaces, and sub-packages. A package in Java can be categorized into two forms: a built-in package or a user-defined package.

The following figure illustrates an example of a package and class hierarchy in Java:



import

import keyword is used to import built-in and user-defined packages into Java source files so that a class can refer to other classes of different packages by directly using its name.

```
import java.util.*;
```

here, util is a subpackage created inside java package.

Access Modifiers

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors.

The four access levels are: *default, public, private,* and *protected*.

The following table describes their use:

Accessible(Yes/No)	**Public**	**Protected**	**Default**	**Private**
Same Class	Yes	Yes	Yes	Yes
Same Package Class	Yes	Yes	Yes	No
Same Package Subclass	Yes	Yes	Yes	No
Other Package Class	Yes	No	No	No
Other Package Subclass	Yes	Yes	No	No

Non -Access Modifiers

Java provides a number of non-access modifiers to achieve other functionalities:

- The *static* modifier for creating classes, methods, and variables.
- The *final* modifier for finalizing the implementations of classes, methods, and variables.
- The *abstract* modifier for creating abstract classes and methods.

2.2 XML

Extensible Mark-up Language (XML) is a mark-up language like HTML that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is a textual data format with strong support via Unicode for different human languages.

- It is designed to store and transport data
- It is designed to be self-descriptive

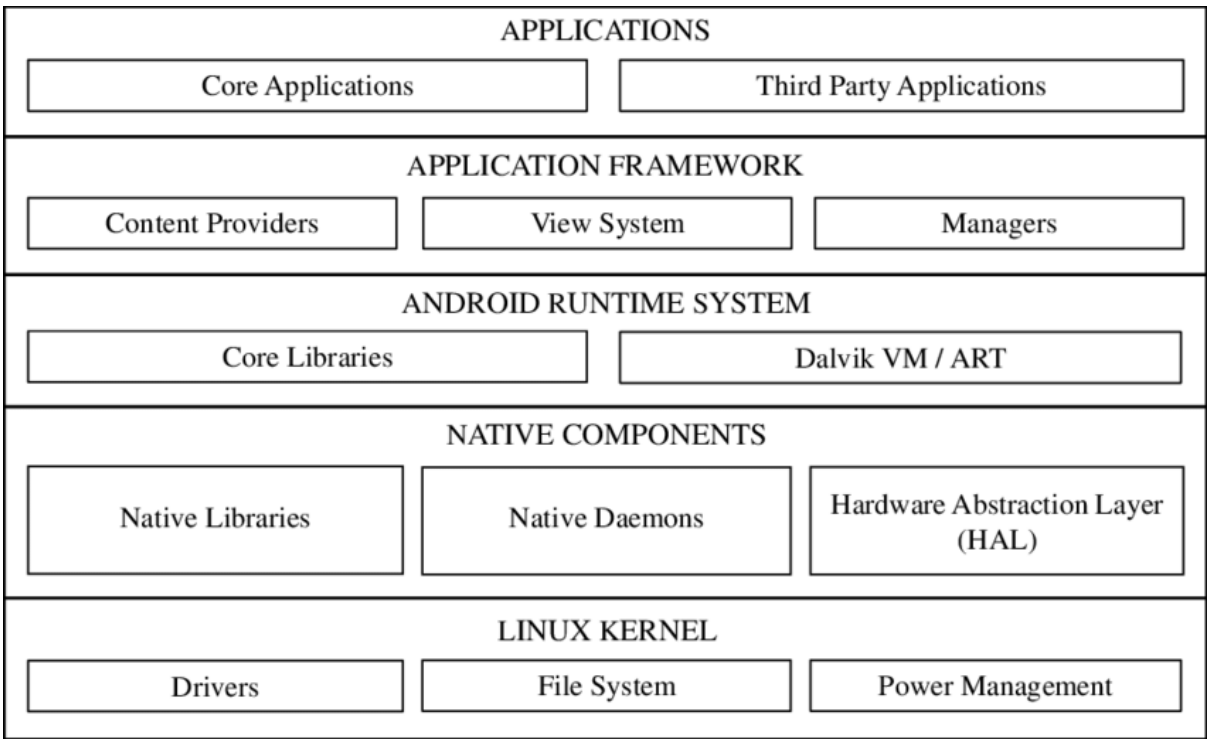
The Android platform uses XML files in projects for providing basic configuration of the application in the Manifest File and using XML Layout Files to define the user interface.

2.3 Android and Android Application Fundamentals

Android is an operating system by *Google* primarily for touchscreen mobile phones. It was developed by Android Inc. which was bought by Google in 2005. It is based on the *Linux Kernel* and is an *Open Source Software*. It is used for touchscreen mobile devices such as smartphones and tablets. It is also used in Android Auto cars, TV, watches and cameras. It has been one of the best-selling operating systems for smartphones.

Android Architecture

It has a *layered architecture*. The following diagram broadly illustrates the different layers of the Android operating system:



Android Architecture

1. System and User Applications

- Android applications can be found at the topmost layer
- System applications have no special status
- System applications provide key capabilities to application developers

2. *Java API Frameworks*

- The entire feature-set of Android is available to us through APIs which are written in the Java language.
- View class hierarchy to create UI screens
- Notification manager
- Activity manager for life cycles and
- Navigation
- Content providers to access data from other applications

3. Android Runtime

- Each application runs in its own environment with its own instance of Android runtime

4. Native Components

- Core C/C++ libraries give access to core native Android system components and services.
- Hardware Abstraction Layer (HAL): Standard interfaces that expose device hardware capabilities as libraries.
  Examples: Camera, Bluetooth module

5. Linux Kernel

- Threading and low-level memory management
- Security features
- Drivers

Android Application

An Android application consists of one or more interactive scenes. It is written using Java Programming Language and XML.
It uses the Android Software Development Kit (SDK) along with Android Libraries and Android Application Framework.

Application Building Blocks

- *Resources*:
  - Resources include folders consisting of various things that are used in the application, sub-folders like drawable, layout, values, etc.
  - Drawable consists of images.
  - Layout consists of XML files that define the user interface.
  - Values store hard-coded strings values, integers, etc.

- *Manifest* is an XML file which is the root of the project source set.
    - It describes the essential information about the application and the Android build tools.
    - It contains the permissions that the application might need in order to perform a specific task.
    - It also contains hardware and software features of the application, which determine the compatibility of the application on Google Play Store.
    - It also includes special activities like services, broadcast receivers, content providers, etc.
- *Build configuration*: APK versions in Grade config files.
    - Each build configuration can define its own set of code and resources while reusing the parts common to all versions of an application.
    - Android plugin for Gradle works with the Build Toolkit to provide processes and configurable settings.
    - Gradle and the Android plugin run independently of Android Studio.
- *Components*: An application's components are the building blocks of Android. Each component has its own role and life-cycle.

Components

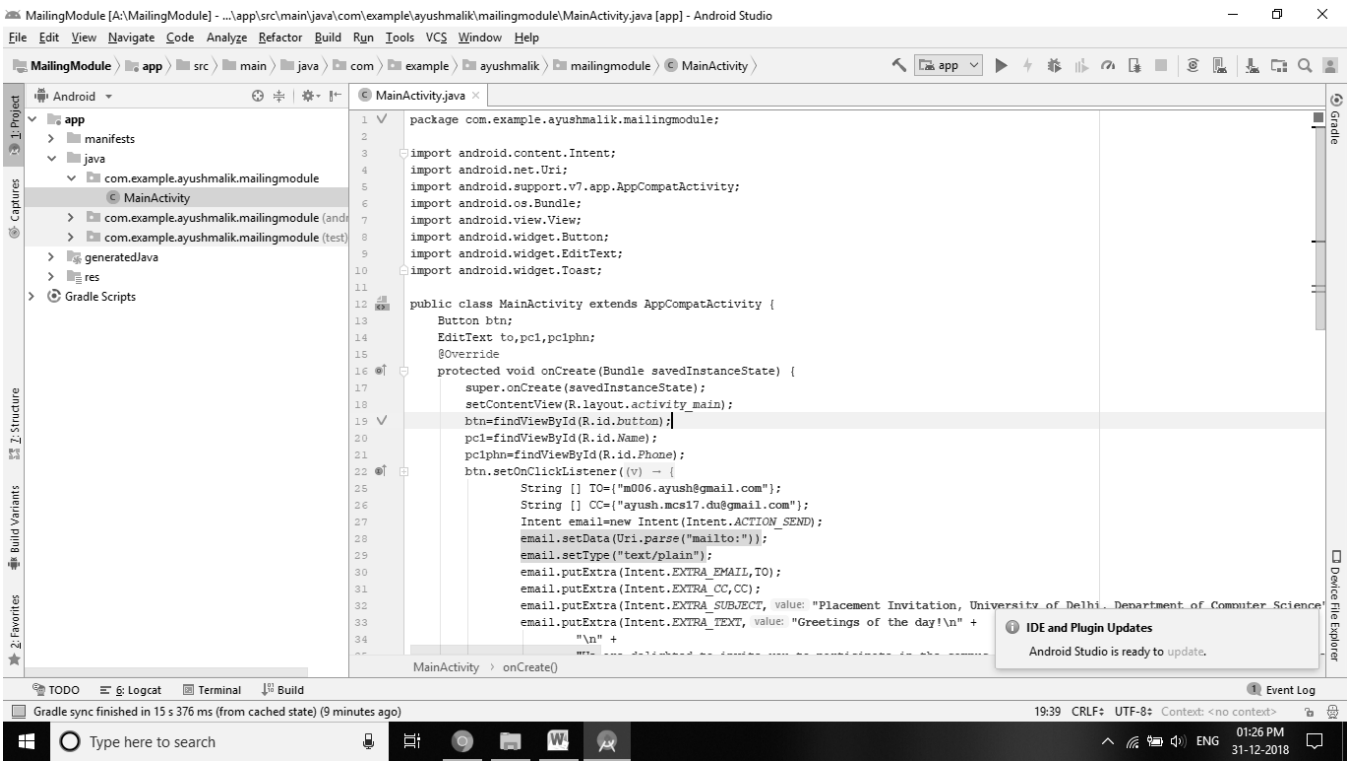
- *Activity:* Activity is a single screen with a user interface. Activities are said to be the presentation layer of applications. The user interface of an application is built around one or more extensions of the Activity class.
- *Service:* Services perform long-running tasks in the background. These components run at the backend, update data sources and Activities, trigger notifications and also broadcast Intents.
- *Content Providers*: They are used to manage application data. They are also responsible for sharing data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured.
- *Broadcast receivers* respond to system-wide announcements. They are known to be Intent listeners. Broadcast Receivers make an application react to any received Intent. They are used in creating event-driven applications.

Android Studio

Android applications are developed using IDE's like Eclipse or Android Studio. We have used Android Studio. Android Studio is the official Android Integrated Development Environment. It is used to develop, run, debug, test and publish Android applications.

It also provides the following:

- Monitoring and performance tools
- Virtual devices
- Project views
- Visual layout editors



Android Studio in Windows 10

2.4 Firebase

Firebase is a *Backend-as-a-Service*—BaaS platform by Google that allows one to develop Android applications quickly. It is a *cross-platform service*. It offers a number of different services built-in, including analytics, database, messaging and crash reporting which allows the developer to focus on other contents of the application. Firebase products work great individually but share data and insights, so they work even better together. Firebase services are used and handled using the *Firebase Console.*

Advantages

- Email & password, Google, Facebook, and Github authentication
- Firebase is built on Google infrastructure and scales automatically
- Real-time data
- Ready-made API's
- Built-in security at the data node level
- File storage backed by Google Cloud Storage
- Static file hosting
- It allows the developer to not worry about infrastructure

Functionalities

The following table summarizes the functionalities offered by Firebase:

Build applications	**Improve quality**	**Scale**
Cloud Firestore	Crash Analytics	Analytics
ML Kit	Performance Monitoring	Predictions
Cloud Functions	Test Lab	Firebase A/B testing
Authentication		Cloud Messaging
Hosting		Remote Config
Cloud Storage		Dynamic Links
Real-time Database		Application indexing
		Invites

We have majorly used the following functionalities offered by Firebase in our project.

Cloud Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It keeps the data in sync across apps through real-time listeners and offers offline support for mobile and web to build responsive apps that work regardless of network latency or Internet connectivity.

Firebase Authentication

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Cloud Functions

Cloud Functions for Firebase lets one automatically run backend code in response to events triggered by Firebase features and HTTPS requests. The code is stored in Google's cloud and runs in a managed environment

ML Kit

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. For an experienced ML developer, ML Kit provides convenient APIs that help to use custom TensorFlow Lite models in mobile apps.

Test Lab

Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, a developer can test his/her Android or iOS app across a wide variety of devices and device configurations, and see the results—including logs, videos, and screenshots—in the Firebase console.

2.5 JavaScript and Node.js

JavaScript

JavaScript is a lightweight interpreted or just-in-time compiled programming language with first-class functions. JavaScript is a prototype-based, multi-paradigm, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it.

Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting.

In our application, JavaScript is used to write Cloud Functions. These functions are deployed on Firebase Server using Node.js.

2.6 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Tensorflow architecture works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

It is called Tensorflow because it takes input as a multidimensional array, also known as tensors. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

Tensorflow is used to design a Neural Network for Company Hiring Prediction in our application.

2.7 Python

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

We have used Python to design a Neural Network using Tensorflow.

2.8 Major Packages and Libraries

The following table briefly describes some major Application Program Interfaces, Packages and Libraries we have used in the application.

Functionality	**Application Progam Interfaces/ Packages/ Libraries**
GPS	com.google.android.gms:play-services-location:16.0.0 com.google.android.gms:play-services-maps:16.1.0'
Artificial Intelligence	com.google.firebase:firebase-ml-vision:18.0.1 com.google.firebase:firebase-ml-model-interpreter:19.0.0
Database	com.google.firebase:firebase-firestore:18.2.0
User Authentication	com.google.firebase:firebase-auth:16.0.5
Cloud Messaging / Push Notifications	com.google.firebase:firebase-messaging:17.3.4
User Interface	com.android.support:recyclerview-v7:28.0.0 com.android.support.constraint:constraint-layout:1.1.3 com.github.PhilJay:MPAndroidChart:v2.2.4 com.android.support:cardview-v7:28.0.0
Others	com.google.gms.google-services

3 . LOGIC DESIGN AND DESIGN METHODOLOGY

This section describes the structure, architecture and/or workflow of major components of our application. It explains the logical design of the functional requirements stated in *section 1.2.1*.

3.1 Cloud Firestore

The application requires the database to be consistent and synchronized across all the devices. We make use of the *Firebase Cloud Firestore database*. The database is maintained and monitored by the *Super User*.

Cloud Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, data is stored in *documents*, which are organized into *collections*.
Each *document* contains a set of key-value pairs. A *document* can also contain a reference to a *sub-collection*.

The database of our application consists of the following collections:

- announce
- company
- expenditure
- mails
- placement_coordinators
- placement_events
- students

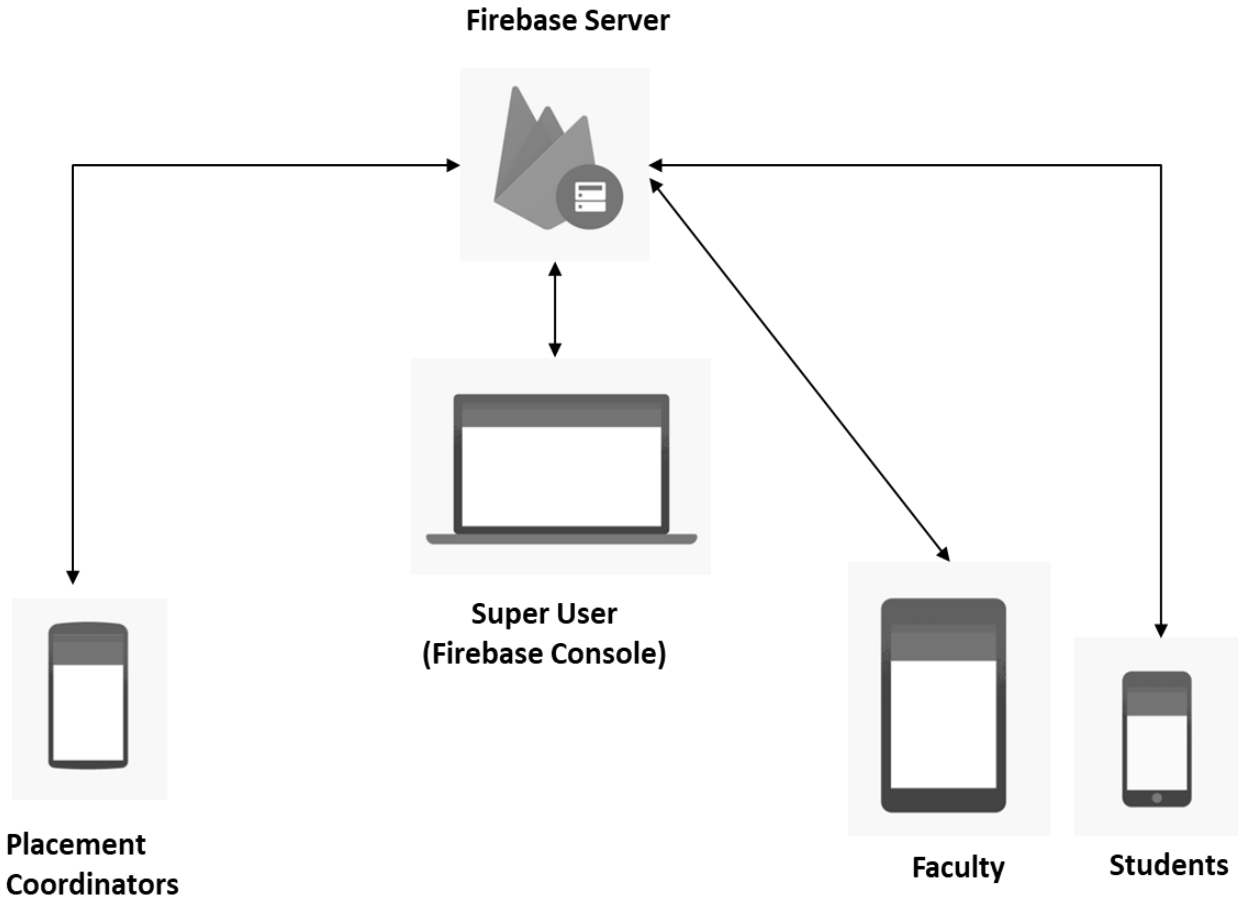
The entire structure of the database has been explained in more detail in *section 4.1*.

Cloud Firestore forms a crucial part of our application. It is not only used for storing data but it is also instrumental for a number of other functionalities in our project. This has been described in detail in the following section.

3.2 Firebase Server

Firebase Services form the backbone of the application. These services will be handled, maintained and monitored by the *Super User* using the *Firebase Console.* The computer via which the *Super User* will interact with the *Firebase Server* will be the *Trusted Environment*.

The following figure explains the logical setup of the above architecture:

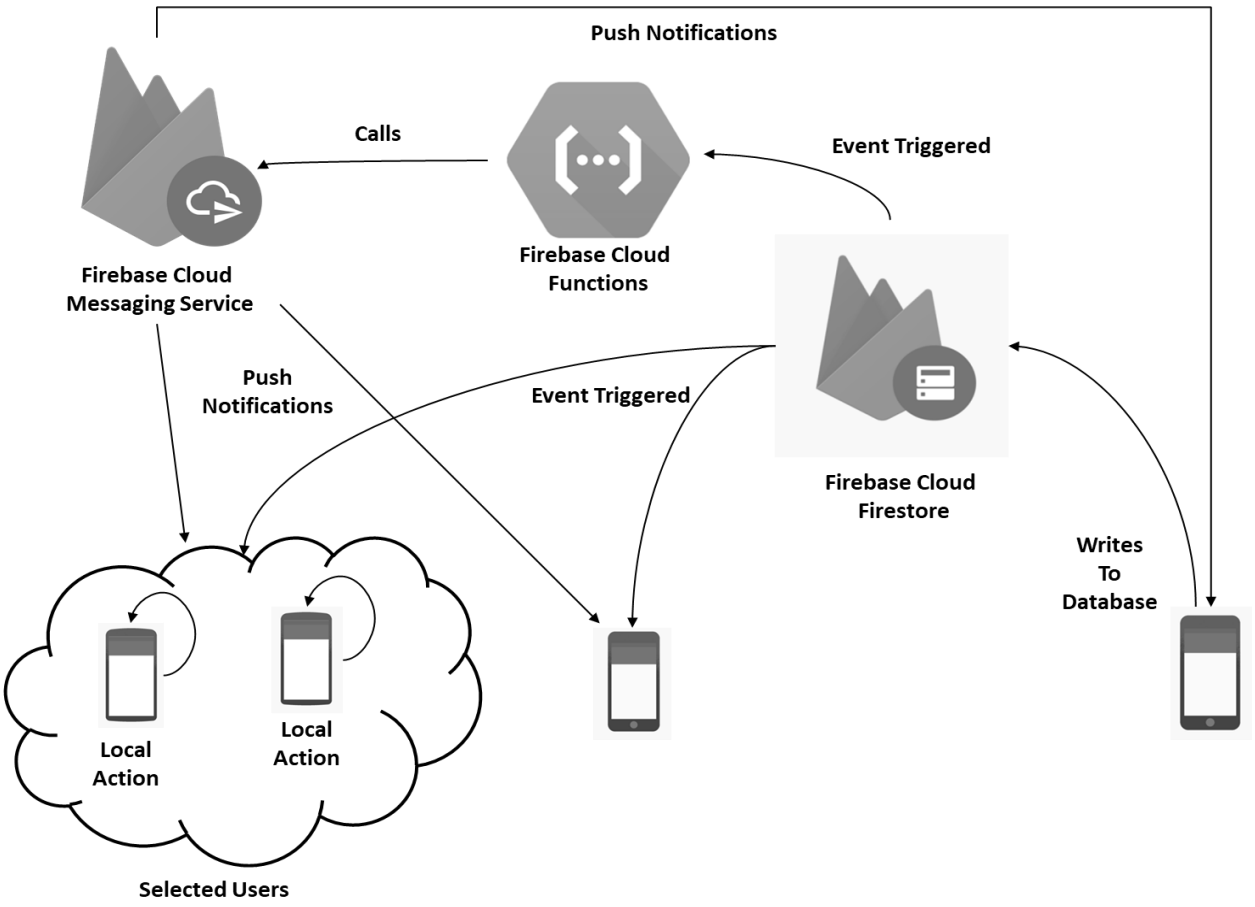


Logical Setup Of The Application

Firebase offers a real-time database called *Cloud Firestore*. Any changes made to the database are immediately reflected across all the devices as soon as they are connected to the Internet. Our application continuously senses different parts of the database for a change in value using *event listeners*.

As soon as there is a change in a concerned value, a corresponding event is triggered. This trigger, in turn, calls other functions of the application and in some cases it also calls *Firebase Cloud Functions*.

These *Firebase Cloud Functions* are pre-defined functions written in JavaScript (Node.js) and are deployed on the Firebase Server. We use these functions to send *Push Notifications* to users using *Firebase Cloud Messaging Service*. The following figure describes this mechanism.



Logical Setup Of Push Notifications And Event Trigger Mechanism Using Firebase

The above mechanism is used to satisfy many functional requirements of our application.

The various functionalities implemented in the application using this concept are discussed below:

3.2.1 Student Registrations

- Our *Cloud Firestore database* has a collection labeled '*placement_events*'. In this collection, there is a document called '*Registrations*'.
- Whenever a PC wants to open registrations for a company, he/ she writes (Company Name, Eligibility Criteria, Course, and Enabled fields) to the '*Registrations'* document in the database.
- This triggers a particular cloud function that sends a push notification using *Firebase Cloud Messaging* to all the users specifying that the registrations for the company are open.
- As soon as a student's application detects that a change has been made in the '*Registrations*' document using event listeners, he/she is allowed access to a registration menu in his/her application.
- The student's application uses the information from the database to determine the eligbility of a student for the drive. It also checks if a student has already registered for the drive. The registration menu is customized depending on a student's eligibility to take part in that drive.
- Eligible students are able to register them by clicking on a button in the menu. This updates an array field named '*company_registered*' in the corresponding user's document in the '*students*' collection.
- Whenever a PC has to close registrations, he/she again writes to

the ‘*Registrations*’ document in the ‘*placement_events’* collection.

- This triggers the same cloud function which sends a push notification to all the users notifying them that the registrations for the company are closed.
- Registrations can be opened only for one company at a time.

3.2.2 GPS Attendance

*G.P.S attendanc*e works on the same lines as Registrations. Instead of determining the eligibility criteria using percentage and course it checks if the student is near the venue and whether the student has registered for the drive or not.

- Our *Cloud Firestore database* has a document called ‘*Attendance*’ in the collection labeled ‘*placement_events*’.
- Whenever a PC wants to enable attendance for a company, he/ she writes (Company Name, his/her Latitude and Longitude, and Radius) to the ‘*Attendance’* document in the database.
- This triggers a particular cloud function that sends a push notification using *Firebase Cloud Messaging* to all the users specifying that the attendance for the company can now be marked.
- As soon as a student’s application detects that a change has been made in the ‘*Attendance*’ document using event listeners, he/she is allowed access to an attendance menu in his/her application.
- The student’s application fetches the current location of the student using *FusedLocationProviderClient* and *Google Play Services.* It computes the distance between the PCs latitude and longitude from the database and the student’s current latitude and longitude using the ‘Distance Formula’. The student is only allowed to proceed if

this distance is less than the *radius* specified by the PC. The application also checks whether the student has registered for the drive or not. The attendance menu is customized depending on a student's eligibility to mark the attendance.

- Registered students within the required distance are allowed to mark their attendance by clicking on a button in the menu. This updates an array field named '*company_attended*' in the corresponding user's document in the '*students*' collection.
- Whenever a PC has to close registrations, he/she will again write to the '*Attendance*' document in the '*placement_events'* collection.
- This triggers the same cloud function which sends a push notification to all the users notifying them that the attendance for the company is over.
- Attendance can be enabled for only one company at a time.

3.2.3 Announcements and Announcement Log

Only a PC can make an '*Announcement*'. All users can see them.

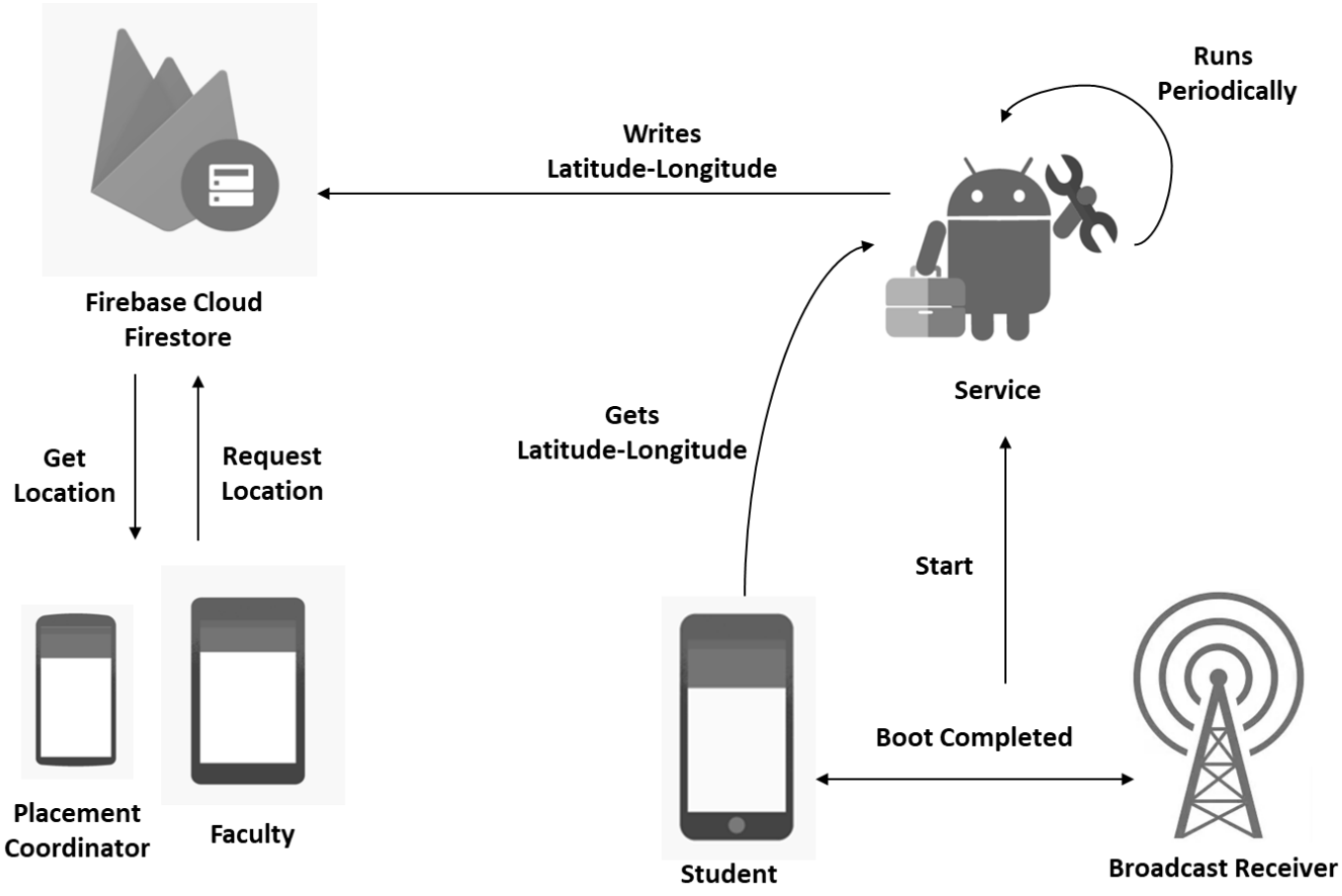
- There is a collection called '*announce*' in the database.
- Whenever a PC wants to make an '*announcement*', he/ she creates a new document in the '*announce*' collection and writes (Title, Type, Content, and Remarks) to it. The date and time at which the PC made an announcement are also written in that document. This data is permanently stored in the *Cloud Firestore database*.
- This creation of a new document triggers a particular cloud function that sends a push notification using *Firebase Cloud Messaging* to all the users specifying that the details of the new announcement.
- Each announcement is stored as a separate document in the

‘announce’ collection.

- Each user can view all the previously made announcements.
- Whenever a user wants to see them, the application fetches all the documents from the ‘*announce*’ collection. It displays them in a visually attractive manner using a *RecyclerView* and a *Master/Detail Flow activity*. This is called the Announcements Log.

3.3 Location Tracking

The following figure helps in understanding how we continuously keep track of each students’ location.



Logical Design Of Location Tracking Functionality

As already specified, the Firebase Cloud Firestore will have a collection named '*students*'. This collection contains a document corresponding to each student. Each student's document contains a Geopoint field named '*Location'* to store the latitude and the longitude of a student's location.

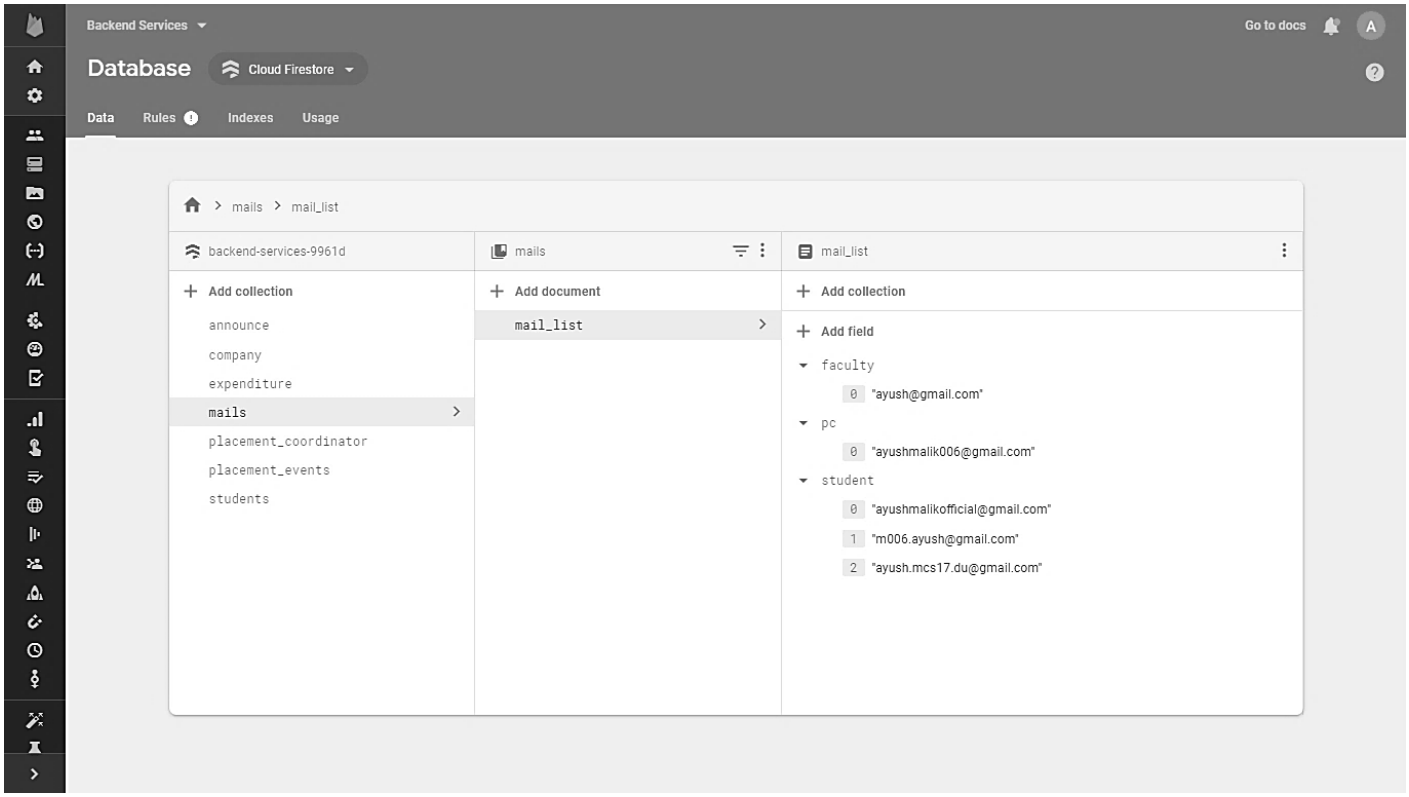
- The application has registered for a *broadcast receiver*. This receiver inform the application when a device's reboot is completed.
- Upon *BOOT_COMPLETED*, the application starts a *service*.
- This *service* is stopped and restarted after regular intervals of time. This is done using *AlarmManager* class.
- Whenever this service is started, it checks if the current user is a student. If he/she is a student, the service fetches the current location (latitude and longitude) of the user using *FusedLocationProviderClient* and *Google Play Services* and writes it to the '*Location*' field in the student's document in the '*students*' collection. This entry is updated after regular intervals of time. For more transparency, the application also stores the time at which the field '*Location*' was last updated.
- When a PC *enables attendance* for a drive, an event listener updates the user interface of all the PCs' and all the faculty members' applications.
- If a *Faculty*/ a *PC* wants to see the location of a particular student, he/she can specify the details of the student. A query to the database is fired which returns the *latitude* and *longitude* of the specified student.
- The latitude and longitude are used to generate an address using *GeoCoder* and are also used to position the marker to the student's current location on a *Google Map*.

- When a PC *disables attendance* for a drive, an event listener again updates the user interface of all the PCs' and all the faculty members' applications. They can no longer access any student's location.

3.4 Login and Authentication

As already explained in *section 2*, the application will use *Firebase Authentication* services to handle login and registration. The *Super User* will handle this functionality by using the *Firebase Console.*

- The *Super User* updates the database each year with the e-mail addresses of all the students, the PCs, and the faculty members.
- The database contains a collection called '*mails*'. In this collection, there is a document named '*mail_list*'.
- This document contains 3 string arrays: *faculty*, *pc,* and *student.* As their name suggests, these arrays contain the e-mail addresses of faculty members, students, and placement coordinators respectively.
- Each user has to register before he/she can use the application.
- We have used a user's *e-mail address* and *password* for authentication.



The Firebase Console Showing The 'mails' Collection

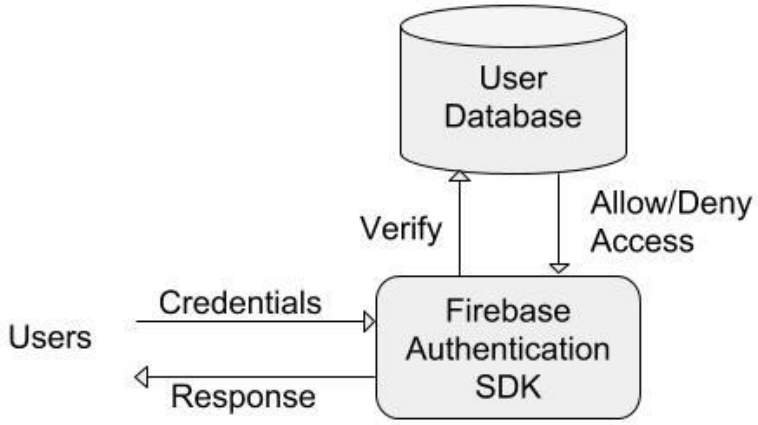
The following sub-sections describe the steps involved in Sign-Up and Sign-In in the application.

3.4.1 Sign Up

- A new user specifies his/her e-mail address. The application searches for this e-mail address in the '*mails*' collection. If it is not found, the user is not allowed to register.
- If found, then the array in which the e-mail address is found determines whether the user is a student, a faculty member or a placement coordinator.
- If the user is a student, then he/she is asked to fill some mandatory information about him/her before completing the sign-up. After this step, all the users set their account's password.
- A verification mail is sent to the e-mail address specified by the user. This completes the sign-up process.

3.4.2 Sign In

- The user is allowed to sign-in only after he/she verifies the link sent to his/her e-mail address.
- After successful verification of the e-mail address, the user enters his/her e-mail address and password.
- These are then verified using *FirebaseAuth*. After this, the role of the user is determined using '*mails*' collection and the user is allowed to access his/her home screen.



Workflow Diagram Of Sign In Using Firebase Authentication Services

3.5 Simplified Mail Handling

This functionality is implemented using *Intents*. There are 3 types of e-mails. Most of the content in each type of e-mail is pre-defined and is thus hard-coded into the application.

- The Placement Coordinator chooses the type of e-mail he/she wishes to send.
- Depending on the type of e-mail, the PC is required to give some input.
- The application then uses this input, pre-defined hard-coded text and some information from the database to create an *Intent(ACTION_SEND).* This intent is used to send information (TO, CC, BCC, SUBJECT, and BODY) to an e-mail application.
- The PC is asked to choose an e-mail client. Upon choosing the e-mail client, the e-mail is automatically composed by the application using the data sent in the *Intent*.

3.6 Artificially Intelligent Hiring Predictor

We have built an *Artificial Neural Network (ANN)* to predict if a student will be hired in his/her placement drive or not.

The ANN is built in *Python* using *TensorFlow* Library. It is trained using a training dataset. The model which was obtained after training is saved as a *Tensor Flow Lite (.tflite)* model in Firebase.

The application makes use of this model using an interface provided by the *Firebase ML Kit*.

We make use of the following qualities of a student as an input to the Neural Network.

- Score in Post Graduation
- Gender
- Communication Skills
- Programming Language Preference
- If the student obtained less than 70 % Marks in 10^{th} or 12^{th}
- Technical Knowledge
- Number of Projects
- Coding Skills
- Course (MCA / M.Sc.)

This module has been described in more detail in *section 4.2*.

3.7 Material Design

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. We have used it for designing the user interface of our application.

Pre-requisites:

- Android Studio version 2.1+ and JDK 8+
- A test device with Android 5.0+

Material UI themes are available in the latest versions of Android Studio. We have referred to https://material.io/ for help.

3.8 Expendtiture

Placement Coordinators will make entries in the *Cloud Firestore* in a collection called '*expenditure*' whenever some money is spent. All the users will be able to view these expenditures. The expenditures can also be viewed using Pie Charts to track and analyze them. Pie-Charts have been designed and implemented using the *MPAndroidChart* library.

3.9 Student and Company Queries

The information about the companies and the students available to a user depends on his/her type. All the students' information will be stored in the's*tudents'* collection and all the companies' information will be stored in the '*company'* collection. First, the user will specify the company's name or the student's name whose information he/she needs. The application will then fetch the information from *Cloud Firestore* and make it available to the user in a visually impressive form.

3.10 Adding Company and Student Records

Only the PCs can add new companies to the database. In order to do so, they simply input the information. The application creates a new document in the '*company*' collection. This information is added to the fields of the newly created document.

Whenever a new student tries to register for the application, in addition to e-mail id and password, he/she has to input his personal and academic details. The application creates a new document in the '*students*' collection. This information is added to the fields of the newly created document. Once saved, this information can't be modified by the student. This is done in order to ensure consistency in the data shared with the companies.

4 . CODING AND IMPLEMENTATION

The complete source code of the application is too large to be included in the report. We have made the source code of our application available at the following link:

https://drive.google.com/drive/folders/1i-SazQDDfEMLTNEyZ3qe6nJ9sTC-tRFP?usp=sharing

In the following sub-sections, we have discussed the implementation details of 3 important components of the application:

- Cloud Firestore Database
- Artificial Intelligence
- G.P.S.

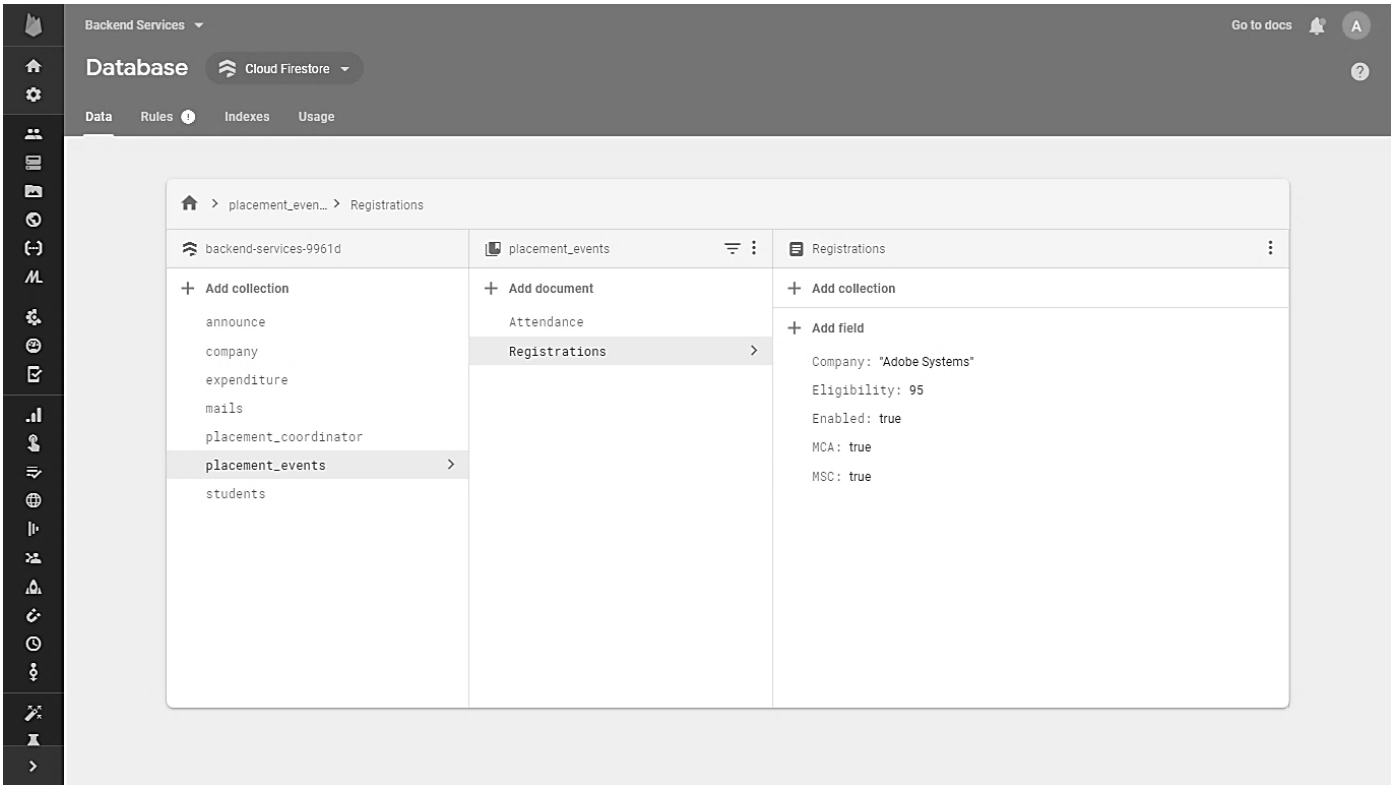
4.1 Cloud Firestore Database

In this section, we describe the structure of our database in detail. We have created the following collections in our Cloud Firestore Database:

- *announce:* This is used to store the information of all the '*Announcements*' made by the PCs. Each announcement is stored in a separate document. Whenever a new document is created a Cloud Function is called to send push notification to users.
- *company:* All the information about the companies is stored in this collection. Each document contains information corresponding to one company.
- *expenditure:* It is a collection store information of all the expenditures.
- *mails:* This collection contains only one document called '*mail_list*'. As described in *section 3.4.*, this document is used to

verify the e-mail addresses during sign-up and assign roles to users based on their e-mail addresses.

- *placement_coordinators:* This collection will always contain 7 documents, one for each PC and one specifying the batch year. This information will be updated annually by the Super User.
- *placement_events:* This collection contains 2 documents: *Registrations* and *Attendance*. The use of these documents has been described in *section 3.3.*
- *student:* All the information of the students is stored in this collection. Each document contains information corresponding to one student.



The Firebase Console Showing The Database

4.2 Artificial Intelligence

In this section, we share the source code used to train and generate the *Artificial Neural Network* we have used in the application. We also describe our training and testing datasets.

- Our training dataset has 250 instances.
- Our testing dataset has 40 instances.
- Each training instance has 9 input features.
- The Neural Network has 2 hidden layers with 10 nodes in each layer.
- The output of the Neural Network gives the probability of the input instance belonging to the 2 classes: *Hired* and *Not-Hired*.

These student's input features used in our prediction model were specified in *section 3.4.* The following table describes them in detail.

Attribute	**Domain**	**Interpretation**
Score	Whole Numbers from 0 to 100 (including 0 and 100)	Percentage of the student in post-graduation
Gender	{0,1}	0- Male, 1-Female
Language	{1,2,3,4}	Preferred programming language 1-C++, 2- Java, 3-Python, 4-Other
Communication Skills	{1,2,3}	1-Average, 2-Good, 3-Excellent

LessMarks	{0,1}	0-Student didn't get less than 70% marks in 10^{th} or 12^{th} 1-Student got less than 70% marks in 10^{th} or 12^{th}
Technical Knowledge	{1,2,3,4,5}	1-Poor ; 5-Excellent
Coding Skills	{1,2,3,4,5}	1-Poor ; 5-Excellent
Course	{0,1}	0-M.Sc. , 1-MCA
Projects	Whole Numbers	It indicates the number of projects a student has worked on
Hired	{0,1}	0-Not Hired,1-Hired

4.3 G.P.S.

In this section, we share the source code of some important classes we have used to implement *Location Tracking* described in *section 3.3*.

5 . TESTING

In this section, we describe the devices on which we have tested our application. We also show sample screenshots of our application running on these devices. The following table summarizes the Android versions and API levels supported by our application.

Codename	**Version**	**API level**
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21

The following table gives details about the devices we have used for testing our application.

Serial Number	**Model Number**	**Android Version**	**Rom**	**Display Size**	**Hardware Specifications**
1	Micromax Canvas A1(AQ4501)	5.0	Android One Stock Rom	4.5 inches, 480 x 854 pixels	Mediatek MT6582, 1 GB Ram, 4GB Storage
2	Micromax Canvas A1 (AQ4501)	6.0.1	Android One Stock Rom	4.5 inches, 480 x 854 pixels	Mediatek MT6582, 1 GB Ram, 8GB Storage
3	Lenovo P2a42	7.0	Lenovo Vibe UI 2.0	5.5 inches, 1080 x 1920 pixels	Qualcomm MSM8953 Snapdragon 625, 3GB Ram, 32 GB Storage
4	Xiaomi Redmi 5a	8.1	MIUI Global 10.1	5.0 inches, 720 x 1280 pixels	Qualcomm MSM8917 Snapdragon 425, 3GB Ram, 32 GB Storage
5	Samsung On6	9.0	Samsung One UI	5.6 inches, 720 x 1480 pixels	Exynos 7870 Octa, 4 GB RAM, 64 GB Storage
6	Nokia 5.1 Plus	9.0	Android One Rom	5.86 inches, 720 x 1520 pixels	Mediatek MT6771 Helio P60, 3GB Ram, 32 GB Storage

In the following sub-sections, we show the screen shots of our running application on 3 of the above 6 devices :

(a) **Micromax Canvas A1** (b) **Lenovo P2** (c) **Samsung On6**

5.1 Opening Screen

5.2 Home Screen

6 . CONCLUSION

In this project, we have built an Android application for handling placements in the Department of Computer Science, University of Delhi. The application has been designed keeping in mind time, cost and knowledge constraints.

This application is almost ready to be deployed and to be used for handling placement events in the Department of Computer Science, University of Delhi. We have paid special attention to various exceptions and test cases, resulting in a robust application.

The artificial intelligent company predictor is not fully developed. It needs more time and more data to give reliable results.

There are only a few issues which need to be addressed before deployment:

- Making the database more secure
- Including more data input validations
- Estimating the right cost of using the application

The application doesn’t take into consideration the people who don’t use Android smartphones. We have designed the code and the application in such a manner that it can be worked upon in the future by other students. The entire backend of the application has been designed with Google Firebase. This offers platform independence. The same structure can be used for designing an iOS application.

The code is highly modular. New features and functionalities can be easily added to it.

The application is not yet deployed, we hope that after some more testing and polishing, it will be deployed and used by our Department.

7 . REFERENCES

1. Android Developers

- https://developers.google.com/android/
- https://developer.android.com/

2. Google Firebase

- https://firebase.google.com/docs/

3. YouTube Channels

- https://www.youtube.com/watch?v=xNPkXGdVw7E&index=2&list=PLlyCyjh2pUe9wv-hU4my-Nen_SvXIzxGB
- https://www.youtube.com/watch?v=k1D0_wFlXgo
- https://www.youtube.com/watch?v=v_hR4K4auoQ&feature=youtu.be

4. Java: The Complete Reference, Ninth Edition by Herbert Schildt, Oracle Press

5. Other Online Resources

- www.stackoverflow.com
- https://www.coursera.org/
- https://www.tutorialspoint.com/android/
- https://www.tensorflow.org
- https://medium.com/
- http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html