DEPARTMENT OF COMPUTER SCIENCE PLACEMENT APPLICATION

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 "**Department Of Computer Science Placement Application**" submitted by **Ayush Malik, Rana Ibrahimi** and **Shivani Tiwary** to the Department of Computer Science, University of Delhi, New Delhi, India, for the award of the degree of Master of Science in Computer Science, is a major project work carried out by them under the supervision of **Mr. P. K. Hazra**. To the best of my knowledge 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	Head of Department
Mr. P. K. Hazra	**Prof. Vasudha Bhatnagar**
Department Of Computer Science	Department Of Computer Science
University Of Delhi	University Of Delhi

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 honoured to have worked under him. He has guided us at each step with his expertise. We wish to express our gratitude to the Head of Department, **Prof. Vasudha Bhatnagar** for facilitating us to carry out this project work.

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. We truly believe that this project would not have been possible without them.

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 Department of Computer Science, each year around 6 student *Placement Coordinators (PCs)* are selected to assist and organize the placement drives. These students coordinators are involved throughout the process which involves sending invites to companies, negotiating dates, maintaining student-company database, arranging for company requirements, asking for permissions, 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.* Majority of the conversations of the PCs involves e-mails. This includes companies, students as well as faculty members. All the coordinators use a single Gmail e-mail id. They have to send several mails which have almost similar content but they are required to *re-compose the mails again and again.*
- Another problem is posed by the fact that all the PCs share a Gmail id. If a mail is read by one PC, it happened many a times that it gets neglected by the others because it was no longer an '*unread' mail*. This introduces many problems including inconsistencies.
- 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 spread sheets. Moreover, different companies require different parts of student data. All this leads to creation of several spread sheets which are incredibly *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 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.
- 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 tried to address all these issues and designed our solution for the same.

1.2 Solution

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

1.2.1 User Characteristics

The application is going to cater 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 students 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, assignment of user id/ passwords, new registrations and other system administration work).

1.2.2 Functional Requirements

The functional requirements of the application tries to handle all the above mentioned challenges. These have been specified below:

1. *Login and Authentication*

   Each type of user will be offered different functionalities. The Super User is going to determine the login details (user identity and password) of the faculty members, placement coordinators and students. Only people with valid authentication details 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 mails sent by the PCs:

   - Invitation Mails: Mails sent to different companies inviting them for placement drive and sharing with them details such as batch strength, placement brochure and placement coordinators’ contacts.
   - Faculty Permission: These are 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 (including timings and purpose).
   - Student Information: These are 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 mails are sent many times for different companies. The application will allow the PCs to generate these mails by accepting minimal inputs from them. It will automatically generate the content of the mails and insert the input at relevant places, thus simplifying the task of sending mails.

3. *Artificial Intelligent Company Hiring Predictor*

   The application will study the previous hiring numbers of known companies. By applying artificial intelligence techniques on this data, the application will try to predict the number of jobs the company will offer in the coming year(s).

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/ synchronised 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 which they can perform on the database.

6. *GPS based attendance*

   The Placement Coordinators will take an 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 timings for which a student will be able to mark his/ her attendance.

7. *Placement Events*

   Only a PC will be able to create a placement event. A PC can add a placement event to the database. Each such event specifies the complete details of a placement drive, including company details, date, package, eligibility criteria, etc. As soon as a PC creates a placement event, a notification will be sent to all.

8. *Mail Log*

   This functionality will be provided to all 3 types of users.All the PCs share a common placement e-mail address. All the mails received by the students and the faculty members from this official placement e-mail address will be logged and maintained separately for easy viewing. All the mails received by the Placement Coordinators from known company e-mail addresses will be logged and mainted for the Placement Coordinators.

9. *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 event has been created by a Placement Coordinator.

10. *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.

11. *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 recievers

(faculty and students). The recievers will also be able to view and check old announcements.

12. *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 accountatbility and transparency.

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

14. *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 eligible its drive, the list of students who registered for the drive, the list of students who attended its drive and the list of the ones who got selected.

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

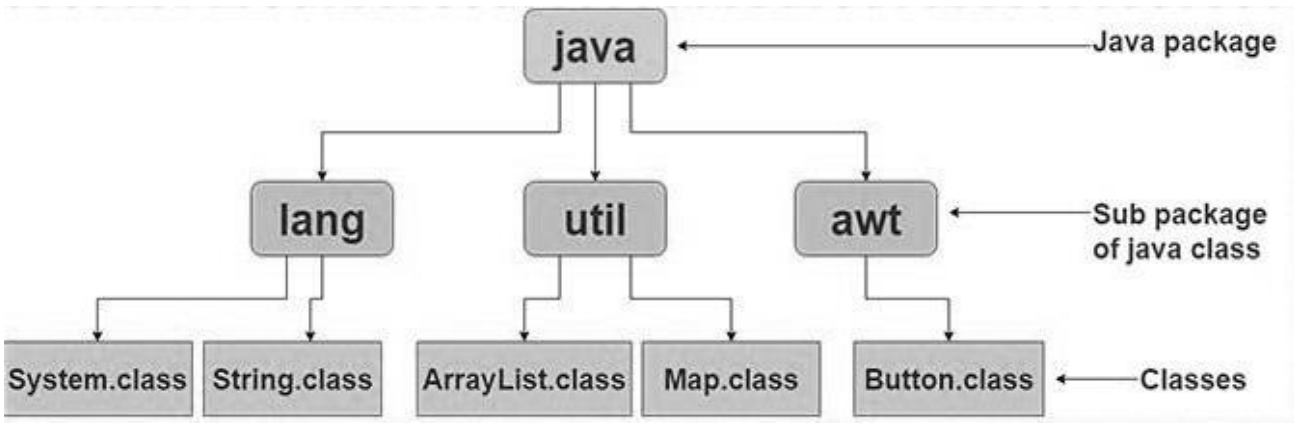
Basic concepts of OOPs are:

- *Objec*t: 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 at 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 in two forms: built-in package and user-defined package.

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



import

import keyword is used to import built-in and user-defined packages into Java source file so that a class can refer to other class of different package 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

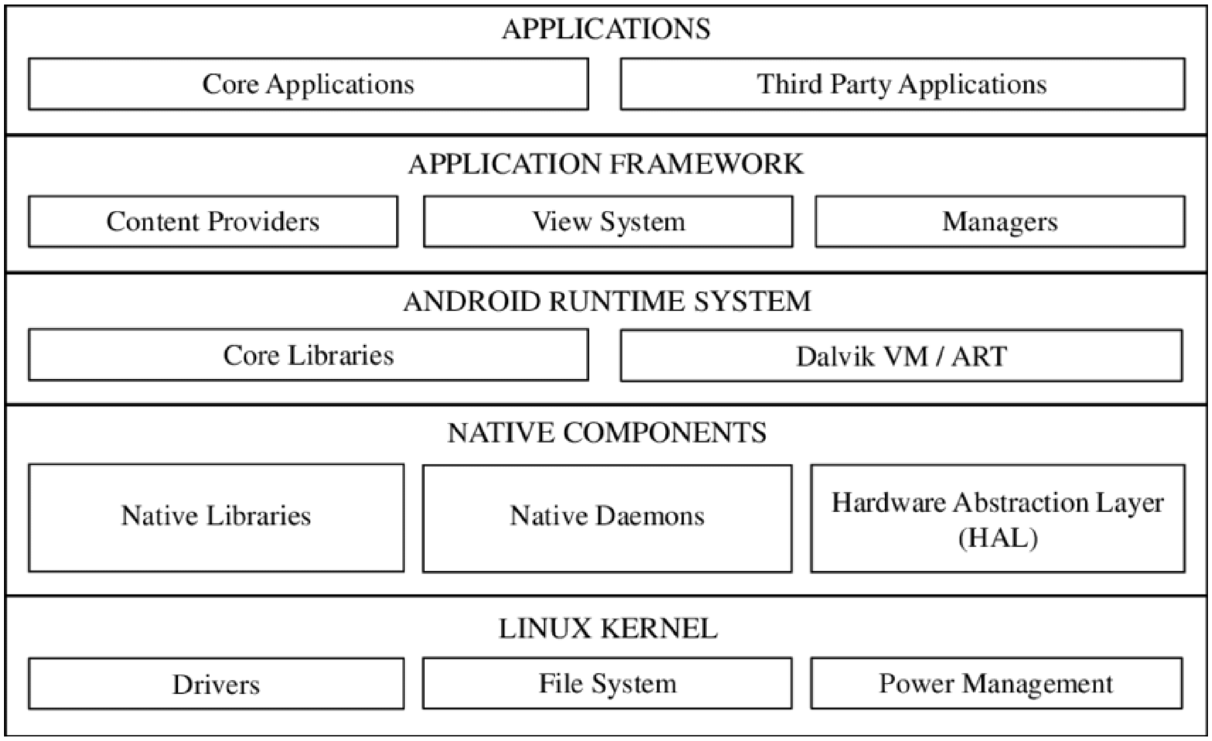
Android platform uses XML files in projects for providing basic configuration of the application in the Manifest File and using XML Layout Files to define the user interface.

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 system for smartphones.

Android Architecture

It has a *layered architecture*. The following diagram broadly illustrates the different layers of 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 provides key capabilities to application developers

2. *Java API Frameworks*

- The entire feature-set of Android is available to us through APIs 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 Play Store.
    - It also includes special activities like services, broadcast receiver, 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 independent of Android Studio.
- *Components*: An application’s components are the building blocks of Android. Each component has its own role and life-cycle.

Components

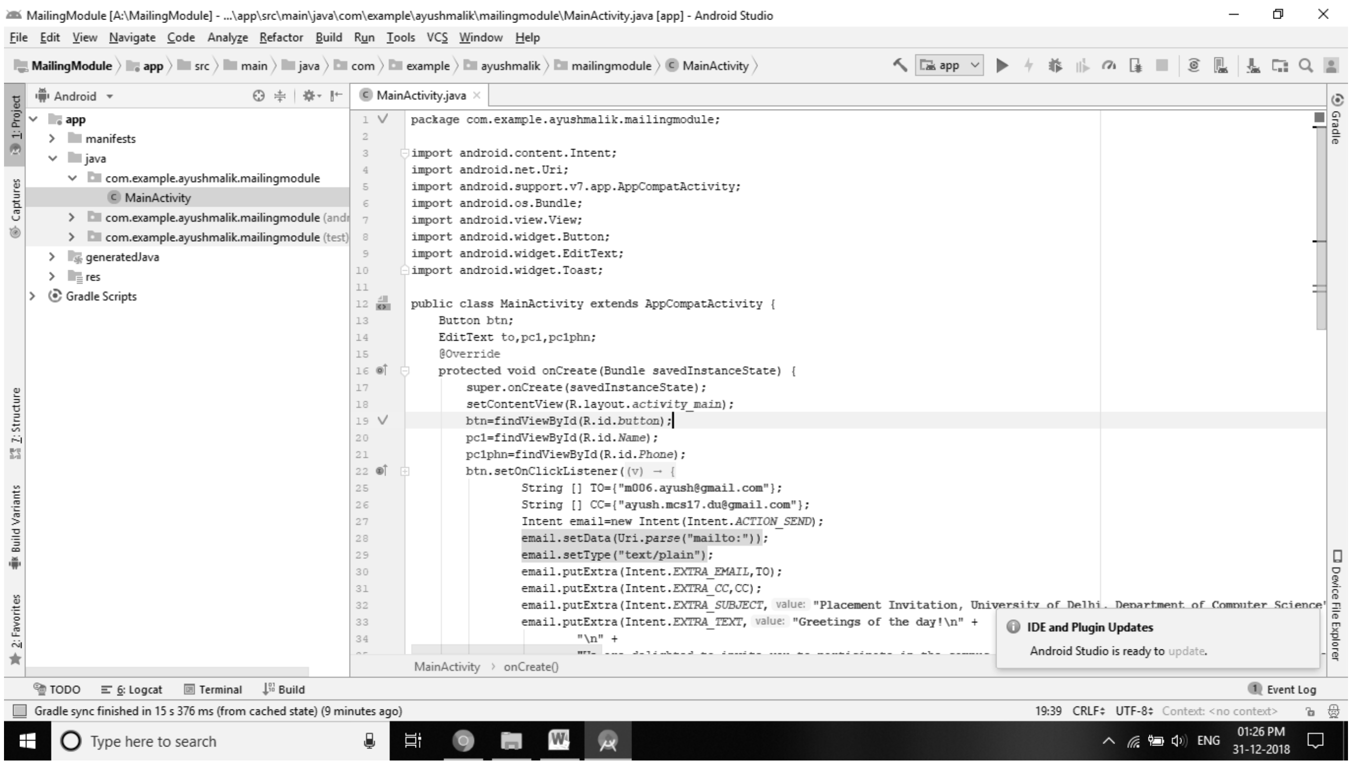
- *Activity:* Activity is a single screen with a user interface. Activities are said to be the presentation layer of applications. The user interface of an application is built around one or more extensions of the Activity class.
- *Service:* Services perform long running tasks in background. These components run at 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



```
package com.example.ayushmalik.mailingmodule;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Button btn;
    EditText to,pc1,pc1phn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn=findViewById(R.id.button);
        pc1=findViewById(R.id.Name);
        pc1phn=findViewById(R.id.Phone);
        btn.setOnClickListener((v) → {
                String [] TO={"m006.ayush@gmail.com"};
                String [] CC={"ayush.mcs17.du@gmail.com"};
                Intent email=new Intent(Intent.ACTION_SEND);
                email.setData(Uri.parse("mailto:"));
                email.setType("text/plain");
                email.putExtra(Intent.EXTRA_EMAIL,TO);
                email.putExtra(Intent.EXTRA_CC,CC);
                email.putExtra(Intent.EXTRA_SUBJECT, value: "Placement Invitation, University of Delhi, Department of Computer Science"
                email.putExtra(Intent.EXTRA_TEXT, value: "Greetings of the day!\n" +
                        "\n" +
```

```
Gradle sync finished in 15 s 376 ms (from cached state) (9 minutes ago)
```

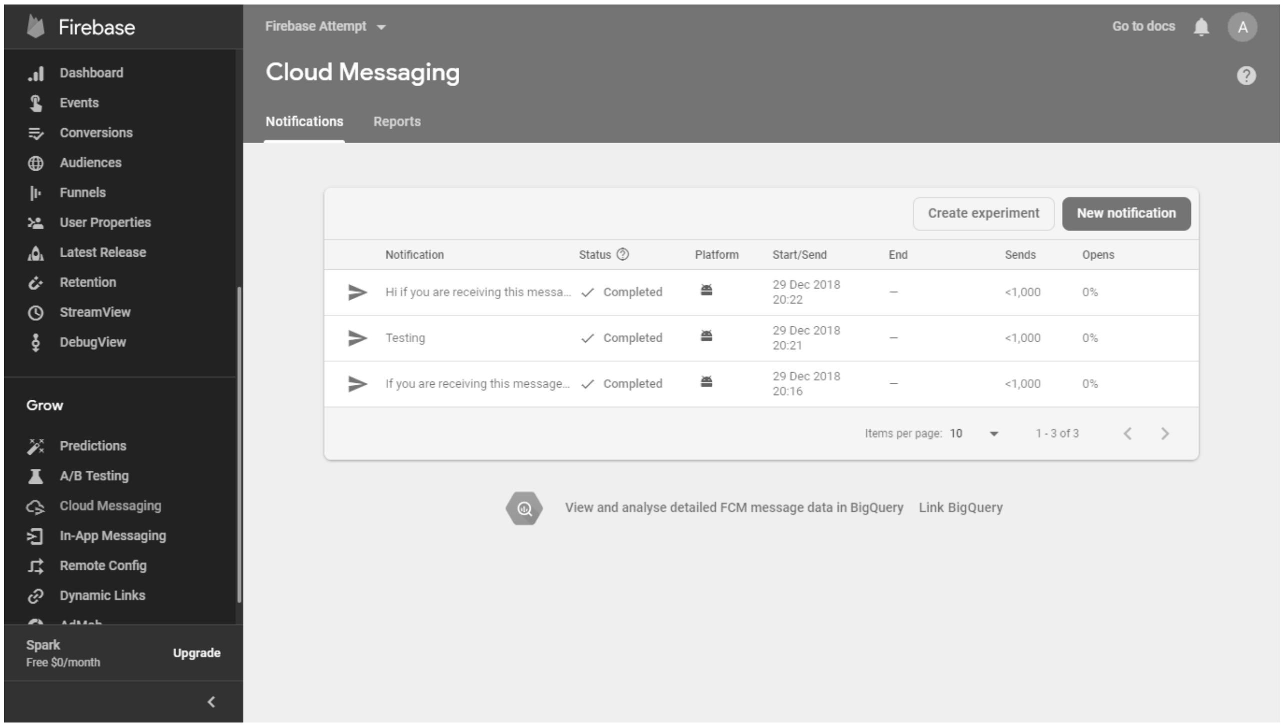
Android Studio in Windows 10

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



	Notification	Status	Platform	Start/Send	End	Sends	Opens
	Hi if you are receiving this messa...	✓ Completed		29 Dec 2018 20:22	–	<1,000	0%
	Testing	✓ Completed		29 Dec 2018 20:21	–	<1,000	0%
	If you are receiving this message...	✓ Completed		29 Dec 2018 20:16	–	<1,000	0%

Items per page: 10 1 - 3 of 3

View and analyse detailed FCM message data in BigQuery Link BigQuery

Figure showing the Firebase Console

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 will majorly use the following functionalities offered by Firebase in our project.

Realtime Database

The Firebase Realtime Database is a *cloud-hosted NoSQL database* that allows one to *store and sync data* between the users in real time.

- *Syncing* allows users to collaborate and access data remotely.
- *Ships* with mobile and web SDKs to *build applications* without the need of servers.
- Users can also execute backend code that responds to events triggered by database using Cloud Functions for Firebase. When a device goes offline, the Realtime Database SDKs use local cache

on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

- Realtime Database with Firebase Authentication provides simple and intuitive authentication for developers.

Firebase Authentication

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

Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like *OAuth 2.0* and *OpenID* Connect.

Firebase Test Lab

Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, you can test your 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 Major API’s, Classes and Libraries

The following table briefly describes some other Tools, API’s, Classes and Libraries we will use in the application.

Functionality	**Tools/ API/ Class/ Library**
Local Notifications	NotificationCompat APIs
GPS	FusedLocationProviderClient LocationServices Geocoder Google Maps SDK
Artificial Intelligence	Android Neural Networks API (NNAPI)
Mail Handling	android.content.Intent android.net.Uri Gmail API
Database	Firebase Real Time Database (NoSQL, JSON)
User Authentication	Firebase Authentication
Cloud Messaging / Push Notification	Firebase Cloud Messaging Service
Material Design	android.support.design
Others	BroadcastReceiver Service MpAndroid

3 . DESIGN METHODOLOGY

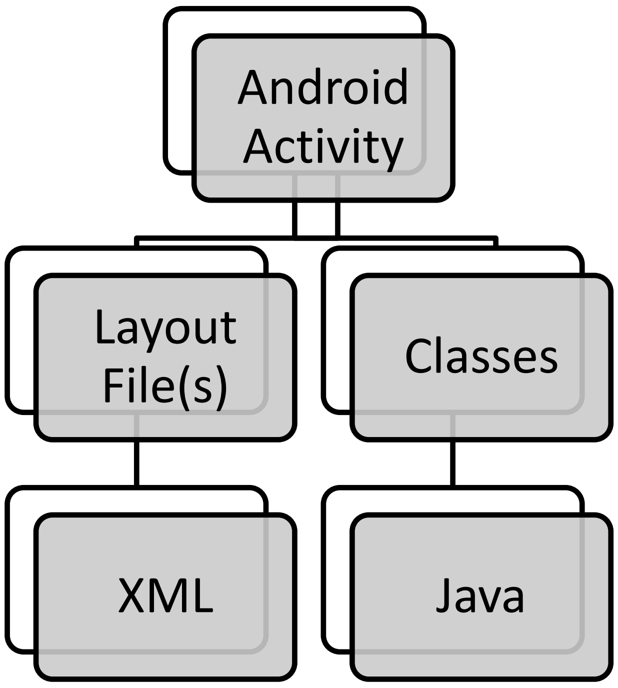
This section describes the structure, architecture and/or work flow of the application. The section explains the logical design of the functional requirements stated in *Section 1.2.1*.

3.1 Front End and Back End

Android broadly makes use of 2 progamming technologies :

- *Java*
- *XML*

XML is used to design the front end, that is, the user interface (with some major exceptions like Android Manifest.xml ,etc.) and *Java* is used to construct the back-end, that is, implementation of classes, methods,etc. These classes and methods may or may not be linked with user interface elements. This can be better understood with the following example.



Constituents of an Android Activity

Whenever, one starts coding in Android, an Activity is one of the first things he/she has to design. An *Android Activity* consists of 2 types of things :

- *Layout File(s)*
- *One or more Java classes*

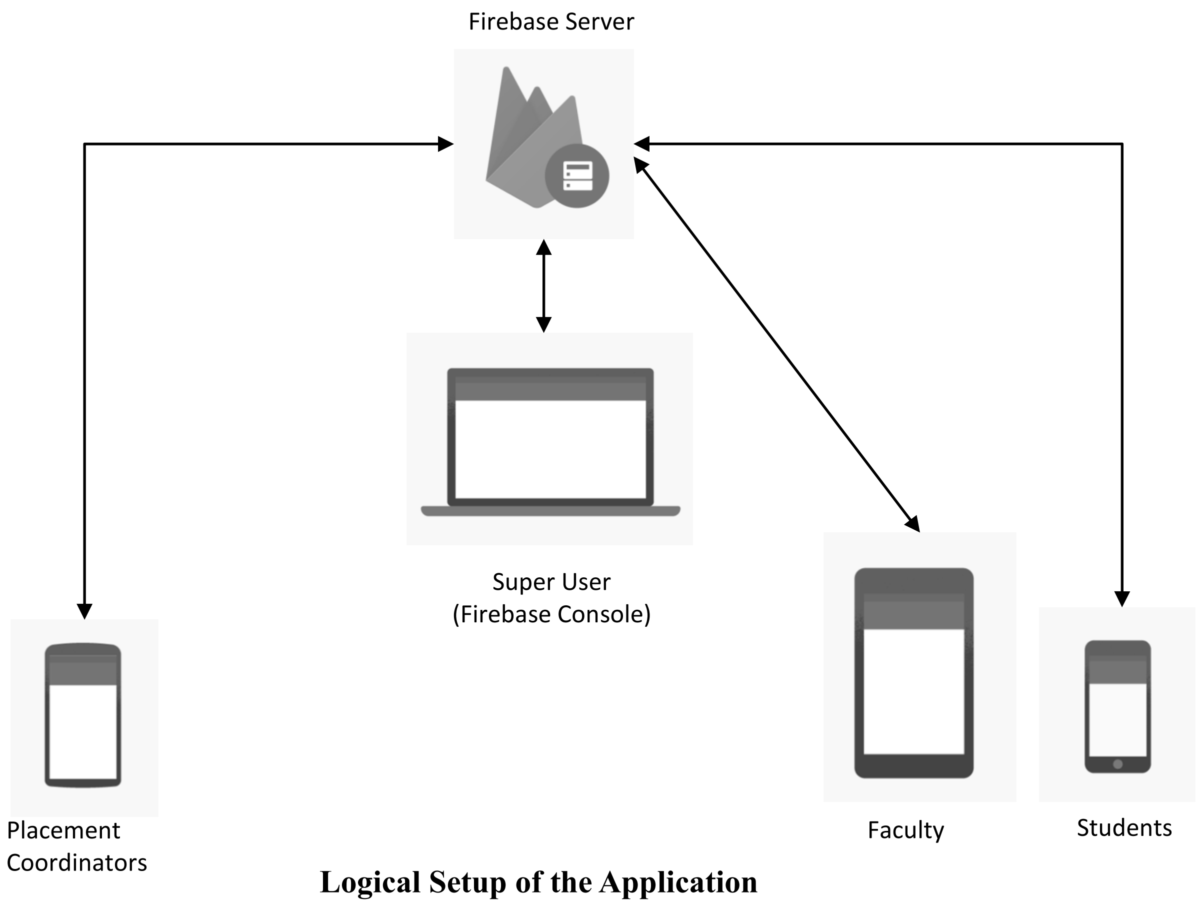
Now, these 2 types of files are connected in such a manner that layout file(s) designs how the activity will appear to the user, hence constituting the front-end. Java classes of an activity will define what will happen when the user interacts with these layout files, hence constituting the back-end.

Apart from what goes on the screen and is visible to the user, there are a lot of other functionalities which an Android application performs in the background or in a manner which is not visible to the users. These include *Services*, *Broadcast Receivers,etc*. These are also a part of back-end and are coded in Java.

3.2 Communication using Firebase Server

Firebase Services form the back-bone 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. Any changes made to the database are immediately reflected across all the devices as soon as they are connected to the Internet. Our application will continuously sense different parts of the database for a change in value. As soon as there will be a change in a concerned value, a corresponding event will be triggered in the application. This trigger will in turn call other functions of the application. This property will be used to satisfy many functional requirements of our application.

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

3.2.1 Placement Event and Student Registrations

- The *Realtime Database* will have an entity storing *Placement Events*.
- Whenever a PC wants to create a *Placement Event*, he/ she will write to this entry in the database.
- As soon as a user's application will detect that a change has been made in the database, it will create a *local push notification* in the user's device.
- There will be an entity for *Student Registrations* for each drive. After an event is created eligible *Students* will be able to register them by creating an entry in the *Student Registrations* entity in the database.

3.2.2 Location Tracking

- The *Realtime Database* will have an entity for storing the *Location of Students*.
- There will a separate entry for each *Student*.
- These entries will remain empty at all times except during the time of a placement drive.
- At the time of a placement drive, the application will write the *latitude* and *longitude* of each student to their corresponding entries in the *Location* entity
- If a *Faculty*/ a *PC* wants to see the location of a particular student. He/ She will fire a query to the database which will return the *latitude/ longitude* of the specified student.
- The *latitude/ longitude* will then be portrayed on Google Map.
- After a placement event is over, the database entries corresponding

to Students’ locations are deleted.

3.1.3 GPS Attendance

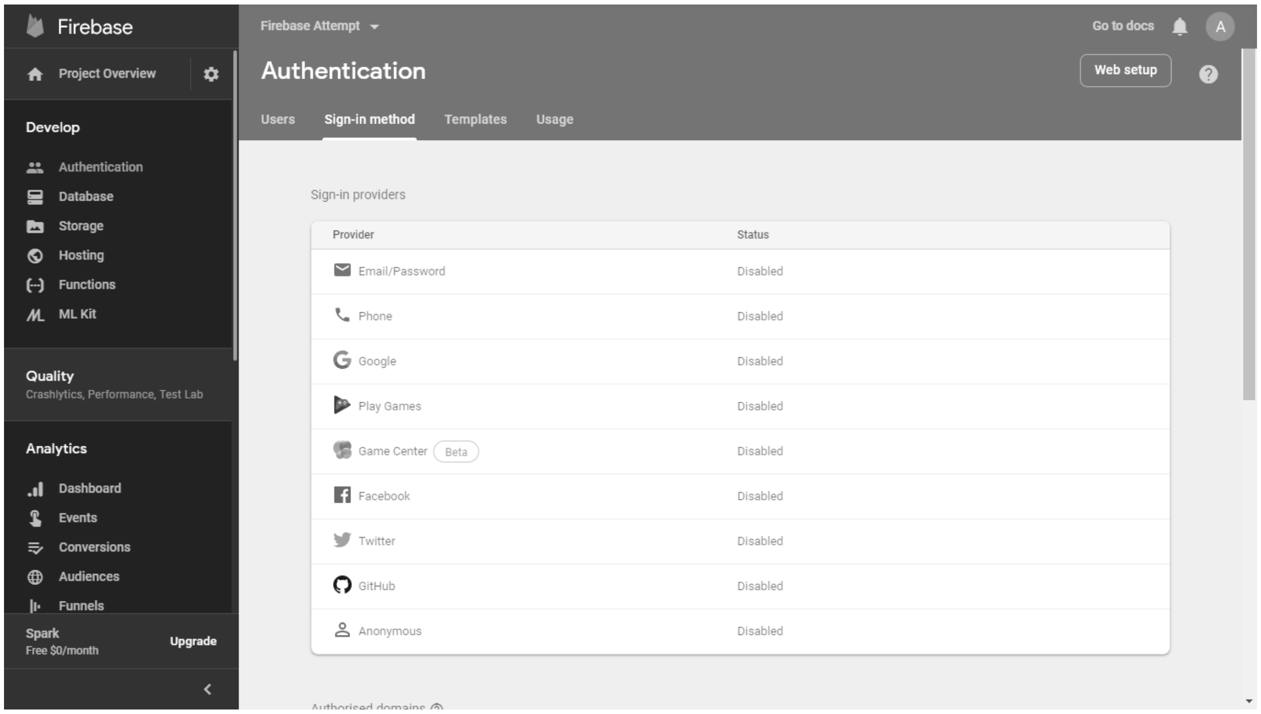
- Each *Placement Event* will have an entry called *Premise.*
- The *Realtime Database* will have another separate entity called *Attendance.*
- The *Attendance Entity* will have entries for each *Student* corresponding to each *Placement Drive*.
- At the time of the placement drive, one of the *PC’s* will write to the *Premise* entry corresponding to that placement drive. The *Premise* defines a circular region. It is defined using a radius and a centre. The centre of the circular region is expressed using *Latitude/ Longitude*.
- During the time of a placement drive, registered *Students* will have an option to mark their attendance. When a *Student* marks his attendance a change is made to the *Attendance* in the database.
- The application locally senses the location of the *Student*. Only when a registered student’s latitude, longitude lies inside the circular region, his/her attendance can be marked.

3.1.4 Announcements to Students/ Faculty

- This will work on same lines as the *Placement Event* functionality discussed in *Section 3.1.1*.
- In this there will be a separate entity for *Announcements* where only a *PC* will be able to write.
- Each announcement will be stored and will be accessible to the users in *Announcements Log*.

3.3 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.*



Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center Beta	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

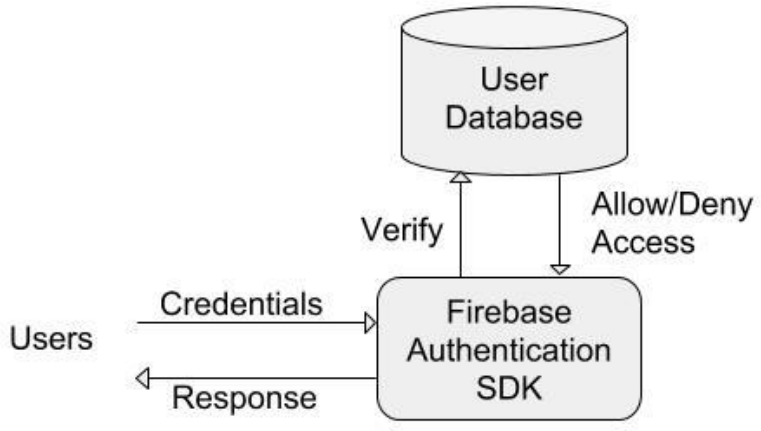
Firebase Console Authentication Menu

Working with Firebase Authentication

- First, the application gets *authentication credentials* (phone number) from the user.
- Then, it passes these credentials to the Firebase Authentication SDK.
- Backend services will then *verify* those credentials and return a *response* (verification code) to the client.

After a successful sign in,

- User's access to the *data stored* in other Firebase products can be controlled.



Work flow diagram of Login and User Authentication

3.4 Simplified Mail Handling

This functionality will be implemented using *Intents*.We have already implemented a prototype of this module which has been described in details.

3.5 Mail Log

The content of mails received from specific e-mail addresses will be accessed and added to the mail log of the user. This list will not be stored. It will be generated everytime from Gmail using *Gmail API* whenever a user opens his/her mail log. This service can be properly accessed only when a user has active internet conncection.

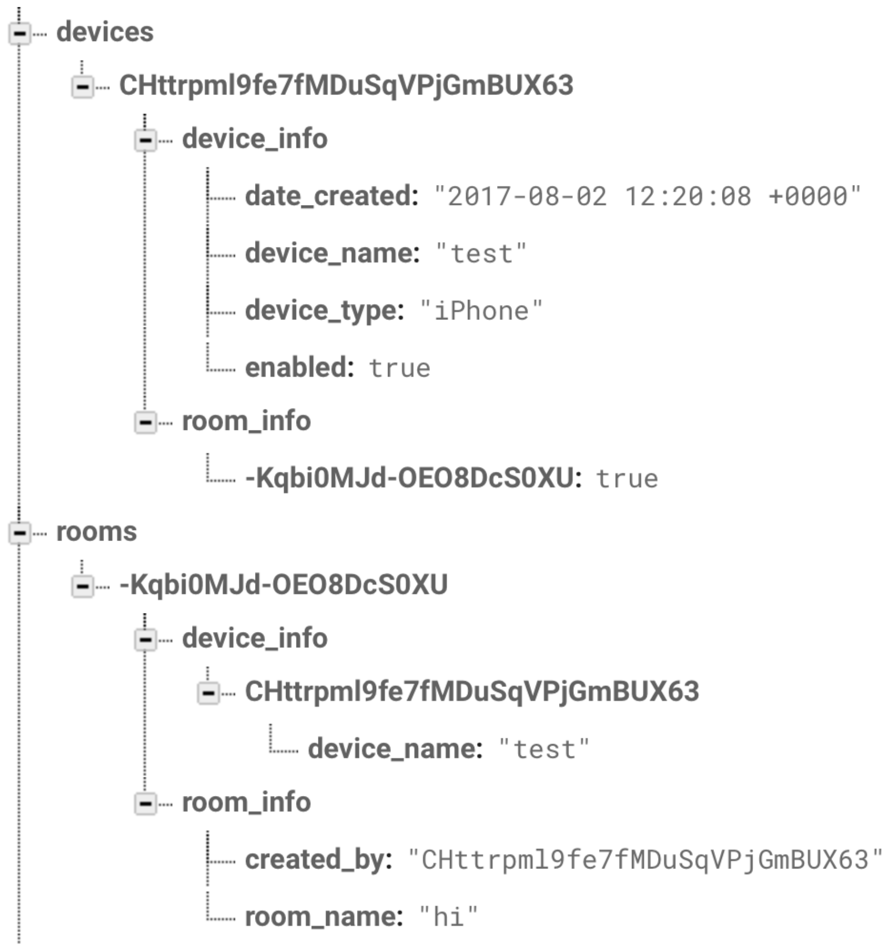
3.6 Real Time Database

The application requires the database to be consistent and synchronized across all the devices. We make use of *Firebase Realtime database*. The database is maintained and monitored by the *Super User*. Other users have pre-defined read and write functions which they can perform on the database.

This database uses/ consists of 2 technologies: JSON and NoSQL.

- **NoSQL** databases are a collection of key-value pairs, documents, graph databases or wide-column stores. They do not have any standard schema definitions which they need to adhere to. They are highly fast as far as searching in database is concerned.
- **JSON** stands for JavaScript Object Notation. It is a lightweight format for storing and transporting data.

The following figures show an instance of the structure of *Firebase Real Time Database:*



```
devices
  CHttrpml9fe7fMDuSqVPjGmBUX63
    device_info
      date_created: "2017-08-02 12:20:08 +0000"
      device_name: "test"
      device_type: "iPhone"
      enabled: true
    room_info
      -Kqbi0MJd-OEO8DcS0XU: true
rooms
  -Kqbi0MJd-OEO8DcS0XU
    device_info
      CHttrpml9fe7fMDuSqVPjGmBUX63
        device_name: "test"
    room_info
      created_by: "CHttrpml9fe7fMDuSqVPjGmBUX63"
      room_name: "hi"
```

Example showing the structure of Firebase Realtime Database

As already explained, the application is highly dependent on the database, not only for storing / receiving information but also for various other functionalities discussed in *Section 3.1*.

Major entities stored in the database include:

- *Student*
- *Faculty*
- *Placement Coordinator*
- *Placement Event*
- *Attendance*
- *Notifications*
- *Expenditure*

- *Student Registrations*
- *Company*
- *Student-Company*

3.7 Artificially Intelligent Company Predictor

This module will be used to create to create a predictor using *Gradient Descent Learning Technique* and *Neural Networks*.

A neural network will be designed and trained using the old hiring numbers of the company. Based on it the neural network will predict the number of job offers the company will make.

We will make use of *Android Neural Networks API (NNAPI)* available in Android.

3.8 Material Design

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices.

Pre-requisites:

- Android Studio version 2.1+ and JDK 8+
- A test device with Android 4.1+

The support libraries provide a number of classes for implementing Material Design. Many of these classes are provided in the *android.support.design* package.

3.9 Expendtiture

Placement Cooridnators will make entries in the database in an entity called '*Expediture*' whenever some money is spent. All the users will be able to view these expenditures. The expeditures can also be viewed using Pie Charts to track and analyse them. Pie-Charts will be designed and implemented using *MPAndroidChart* library.

3.10 Student and Company Queries

The information about the companies and students available to the user will depend on his/her type. All the students' information will stored in *Student* entity and a company's information will stored in *Company* entity. First, the user will specify the company or student whose information he/she needs. The application will then fetch the information from the database and make it available to the user in a visually impressive form.

4 . IMPLEMENTATION

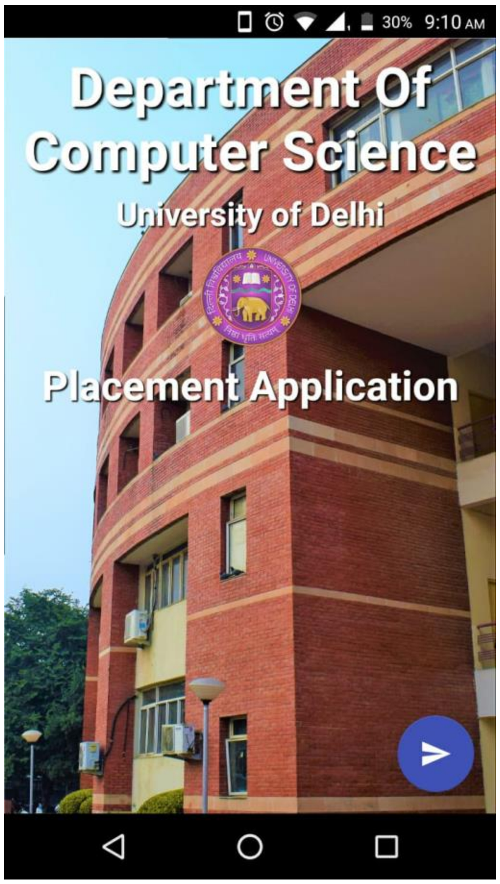
In this section, following the discussion of the logical design in *Section 3*,we have discussed the implementation in details.

We divided our work into 2 parts:

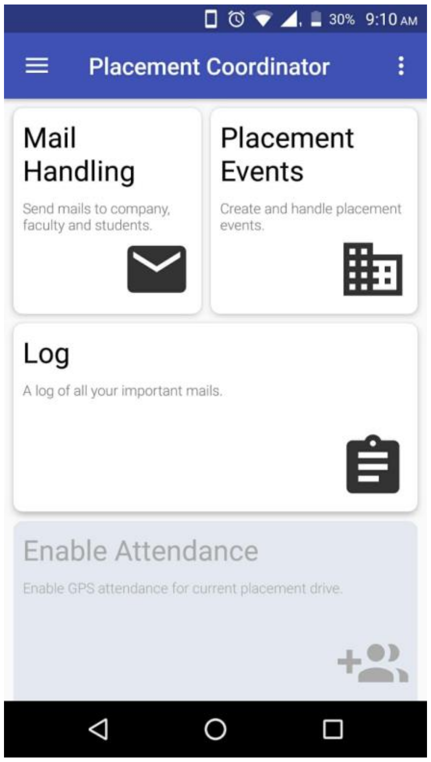
- User Interface
- Functionality Modules

All the elements of the final user interface are designed in one single application. The back-end modules are developed and tested individually and are later added on the user-interface application. We describe the implementation of various components, methods, etc. below.

4.1 Load Screen



4.2 User Interface

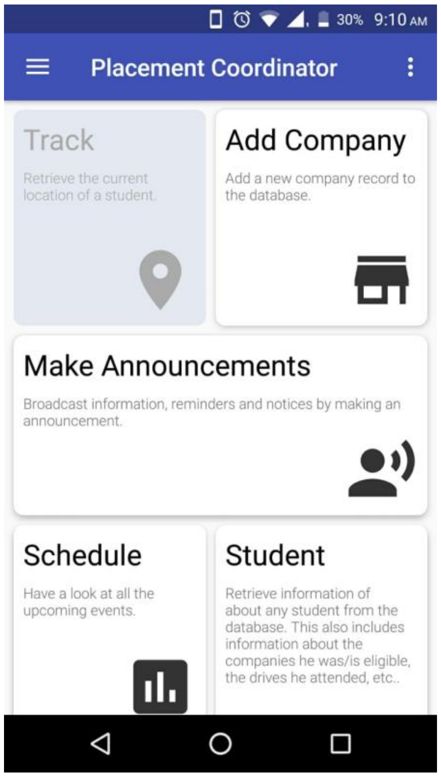


Mail Handling
Send mails to company, faculty and students.

Placement Events
Create and handle placement events.

Log
A log of all your important mails.

Enable Attendance
Enable GPS attendance for current placement drive.



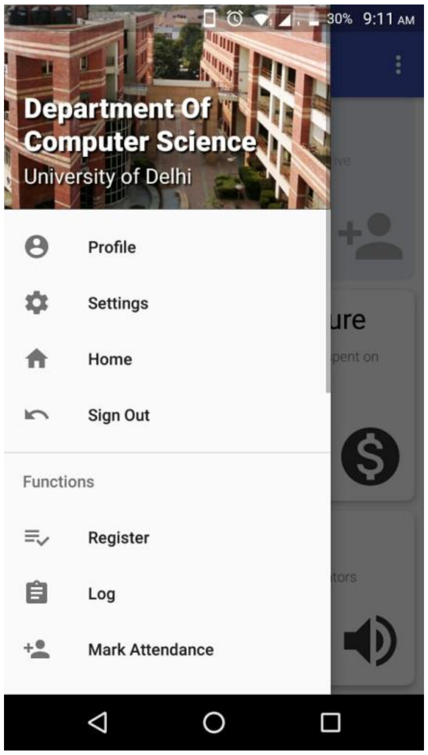
Track
Retrieve the current location of a student.

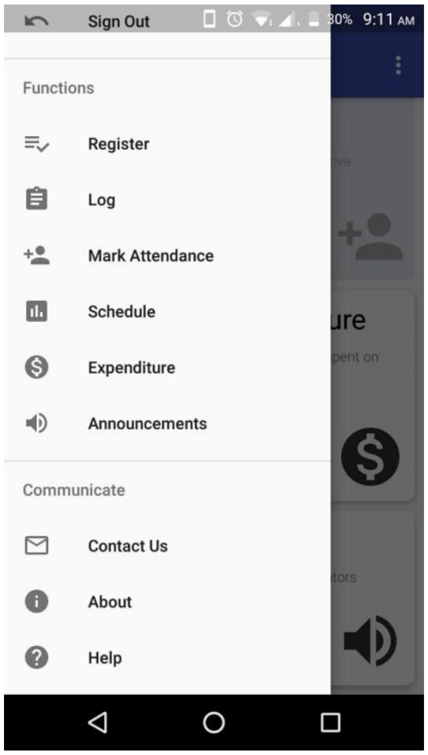
Add Company
Add a new company record to the database.

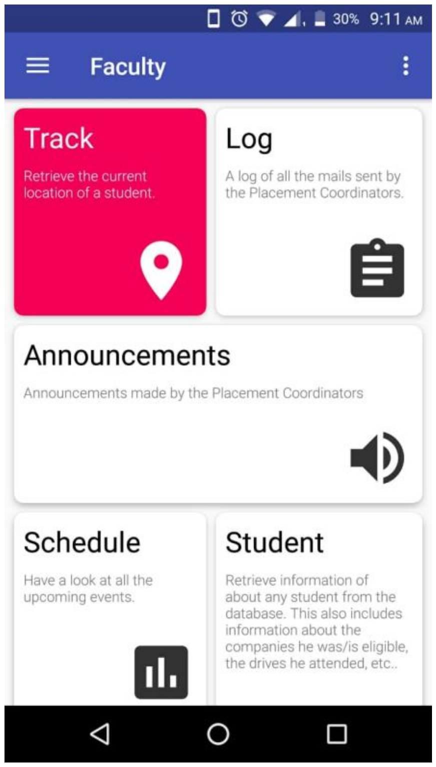
Make Announcements
Broadcast information, reminders and notices by making an announcement.

Schedule
Have a look at all the upcoming events.

Student
Retrieve information of about any student from the database. This also includes information about the companies he was/is eligible, the drives he attended, etc..






Faculty

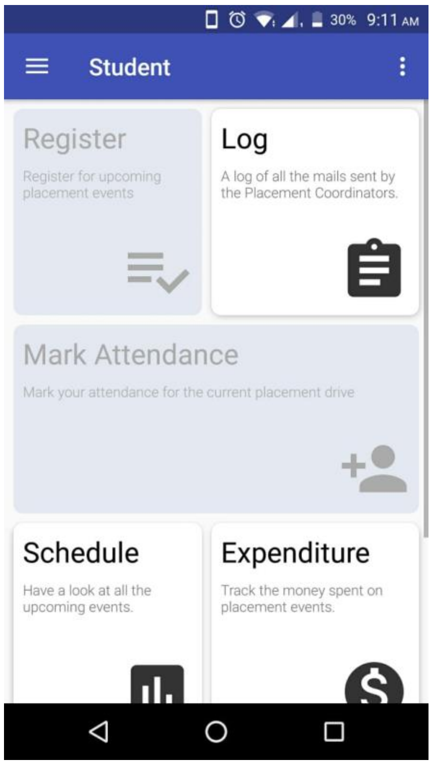
Track
Retrieve the current location of a student.

Log
A log of all the mails sent by the Placement Coordinators.

Announcements
Announcements made by the Placement Coordinators

Schedule
Have a look at all the upcoming events.

Student
Retrieve information of about any student from the database. This also includes information about the companies he was/is eligible, the drives he attended, etc..


Student

Register
Register for upcoming placement events

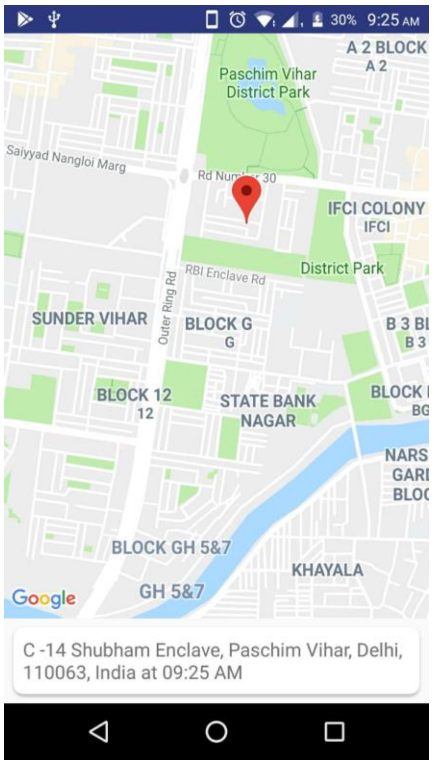
Log
A log of all the mails sent by the Placement Coordinators.

Mark Attendance
Mark your attendance for the current placement drive

Schedule
Have a look at all the upcoming events.

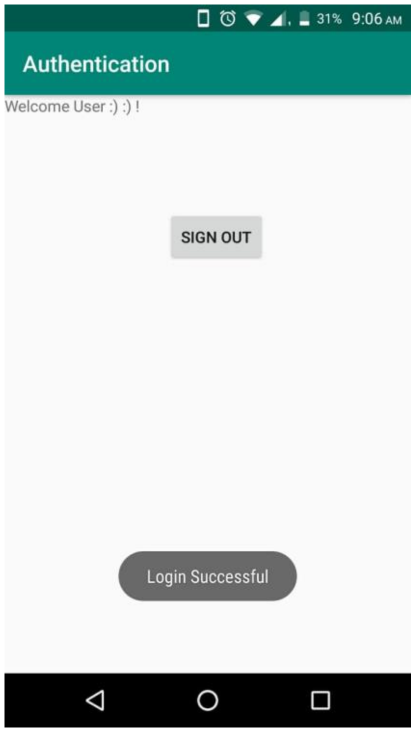
Expenditure
Track the money spent on placement events.

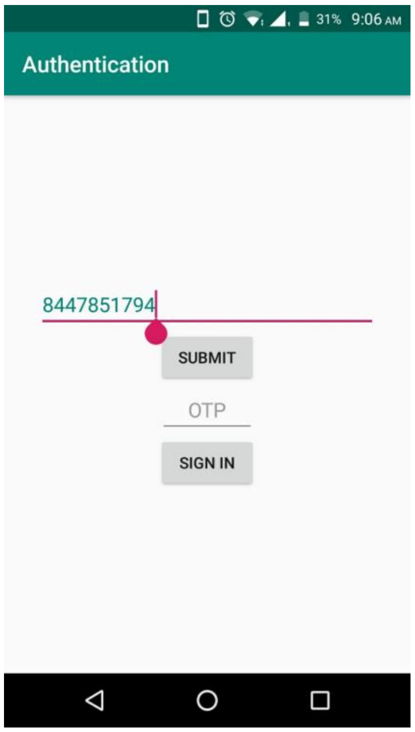
4.3 Track Student

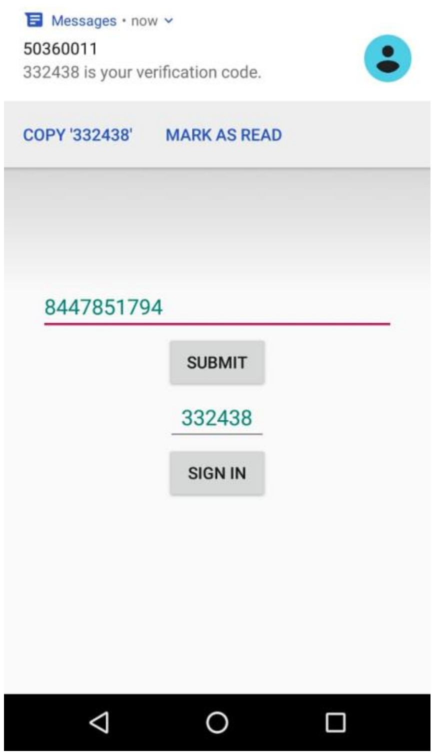


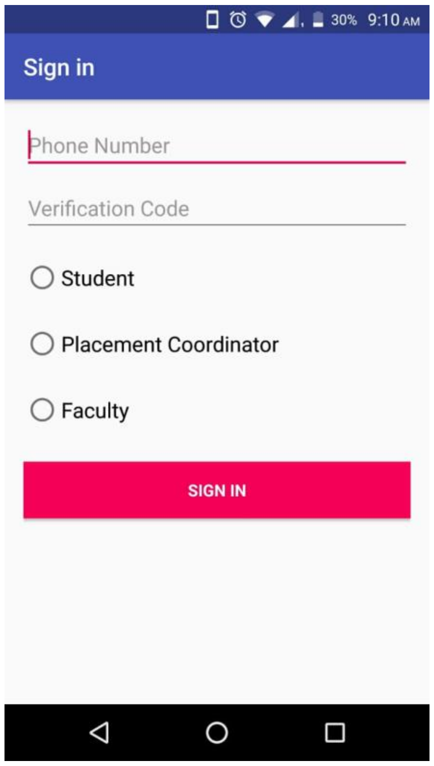
C -14 Shubham Enclave, Paschim Vihar, Delhi, 110063, India at 09:25 AM

4.4 Login and Authentication

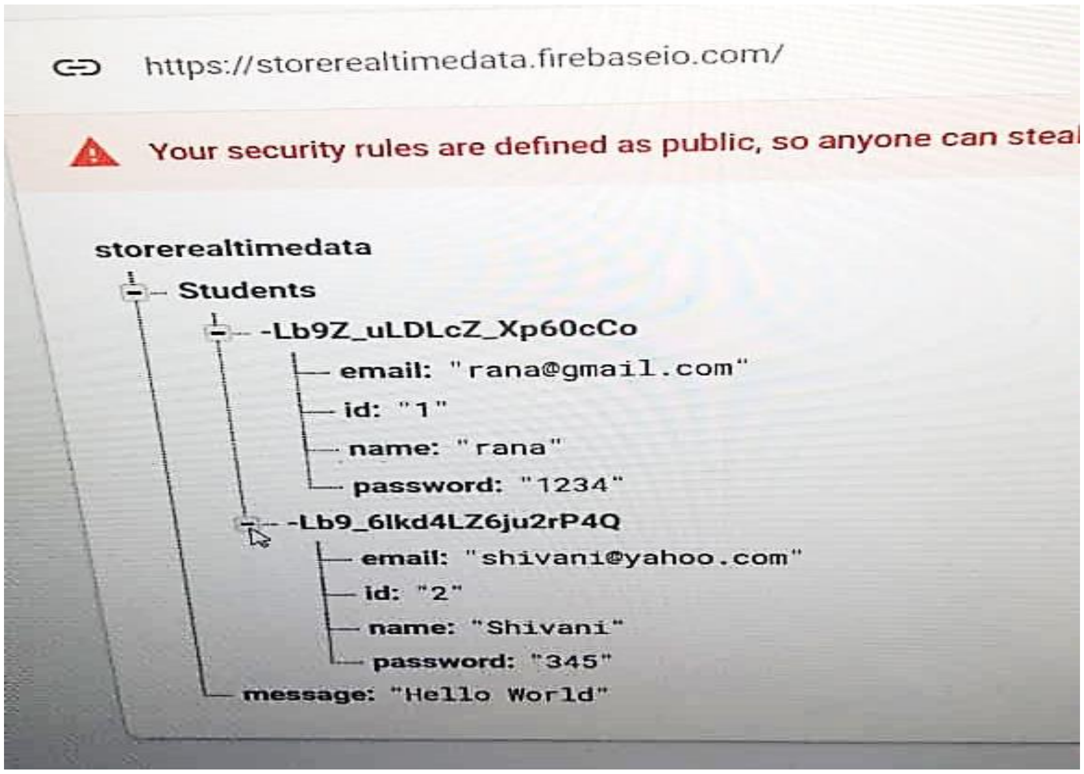








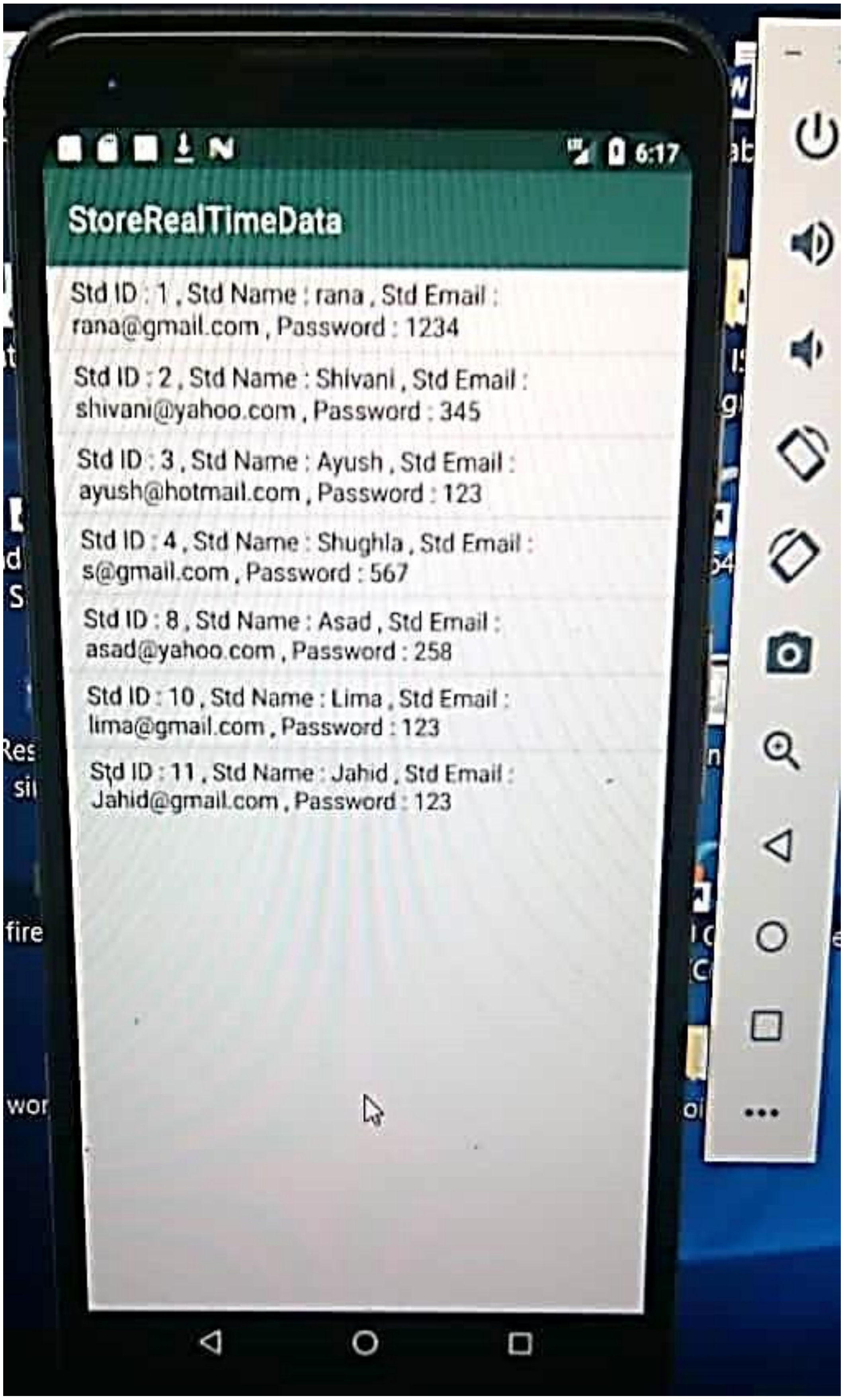
4.5 Storing and Retrieving Data from Database



https://storerealtimedata.firebaseio.com/

Your security rules are defined as public, so anyone can steal

```
storerealtimedata
 ⊟— Students
     ⊟— -Lb9Z_uLDLcZ_Xp60cCo
          ├— email: "rana@gmail.com"
          ├— id: "1"
          ├— name: "rana"
          └— password: "1234"
     ⊟— -Lb9_6lkd4LZ6ju2rP4Q
          ├— email: "shivani@yahoo.com"
          ├— id: "2"
          ├— name: "Shivani"
          └— password: "345"
 └— message: "Hello World"
```



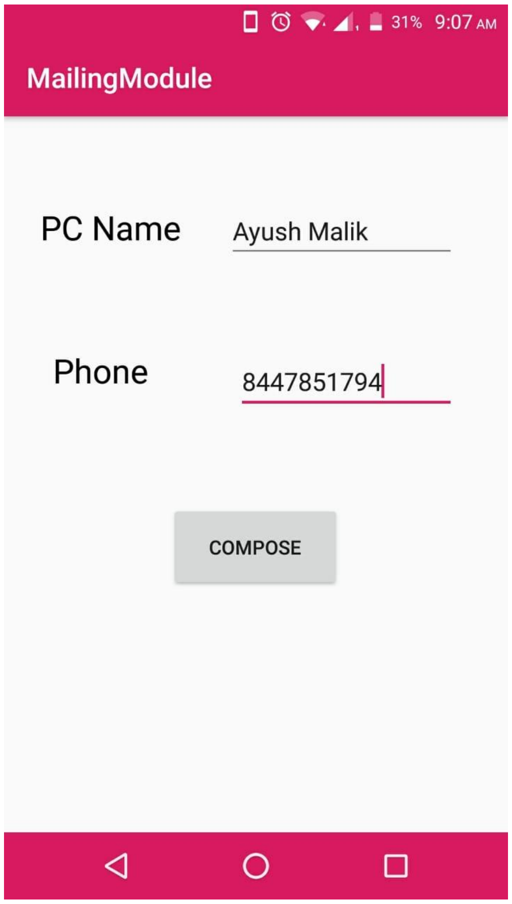
```
Std ID : 1 , Std Name : rana , Std Email :
rana@gmail.com , Password : 1234

Std ID : 2 , Std Name : Shivani , Std Email :
shivani@yahoo.com , Password : 345

Std ID : 3 , Std Name : Ayush , Std Email :
ayush@hotmail.com , Password : 123

Std ID : 4 , Std Name : Shughla , Std Email :
s@gmail.com , Password : 567

Std ID : 8 , Std Name : Asad , Std Email :
asad@yahoo.com , Password : 258

Std ID : 10 , Std Name : Lima , Std Email :
lima@gmail.com , Password : 123

Std ID : 11 , Std Name : Jahid , Std Email :
Jahid@gmail.com , Password : 123
```

4.6 Simplified Mail Handling

This module first asks the Placement Cooridnator to choose ampng the 3 types of mails he/she can send :

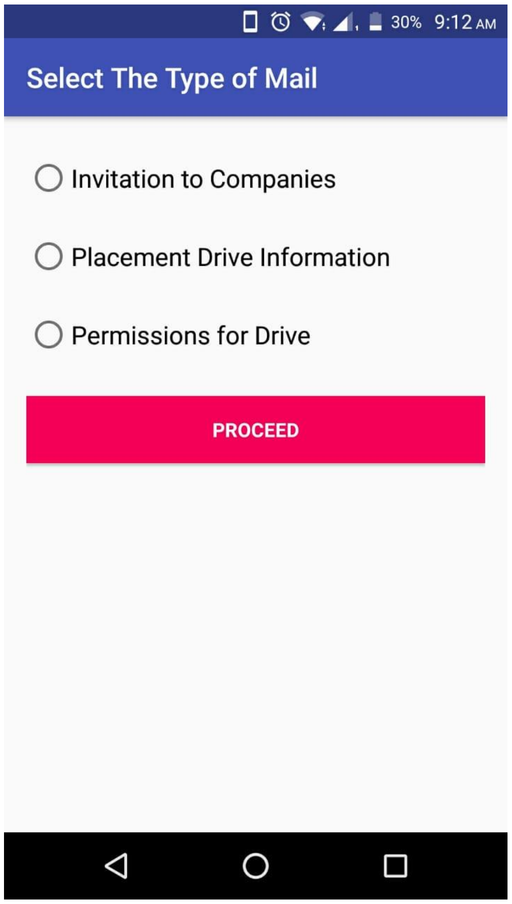
- Inviation to Companies
- Placement Drive Information
- Permissions for Drive

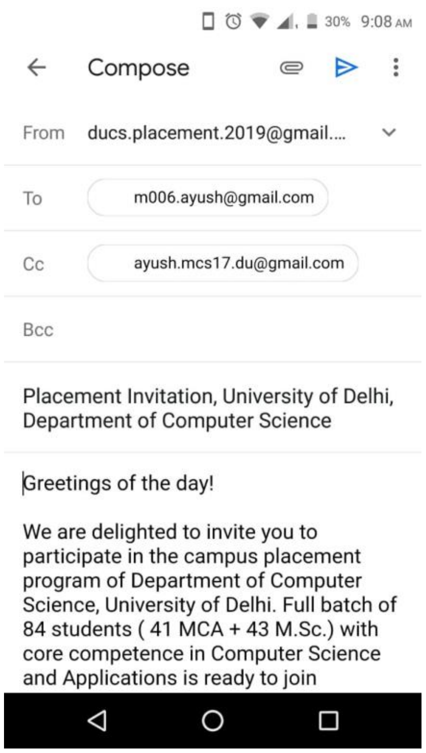
After that depending on the choice, the user is asked to enter some information. The module automatically creates an entire mail an entire which includes Subject, To, CC and Mail body.



PC Name Ayush Malik

Phone 8447851794





From ducs.placement.2019@gmail....

To m006.ayush@gmail.com

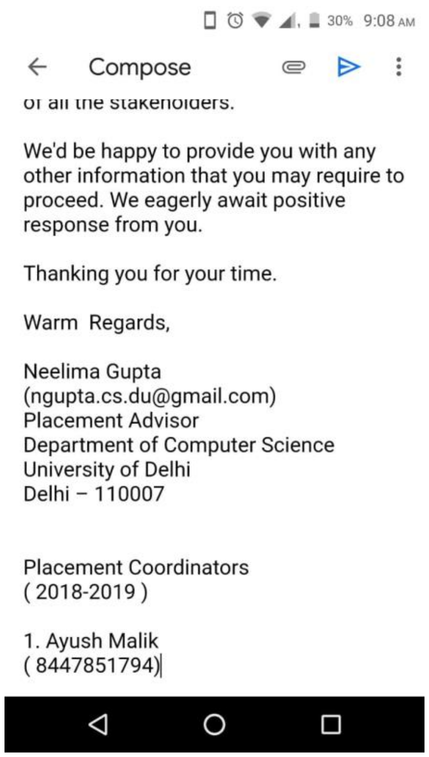
Cc ayush.mcs17.du@gmail.com

Bcc

Placement Invitation, University of Delhi, Department of Computer Science

Greetings of the day!

We are delighted to invite you to participate in the campus placement program of Department of Computer Science, University of Delhi. Full batch of 84 students (41 MCA + 43 M.Sc.) with core competence in Computer Science and Applications is ready to join



of all the stakeholders.

We'd be happy to provide you with any other information that you may require to proceed. We eagerly await positive response from you.

Thanking you for your time.

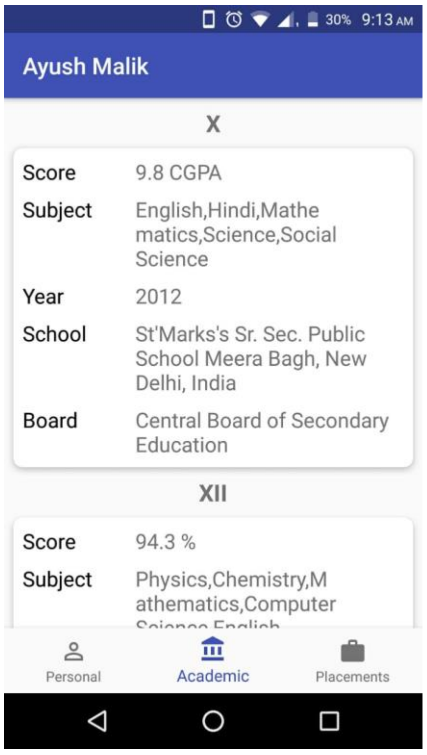
Warm Regards,

Neelima Gupta
(ngupta.cs.du@gmail.com)
Placement Advisor
Department of Computer Science
University of Delhi
Delhi – 110007

Placement Coordinators
(2018-2019)

1. Ayush Malik
(8447851794)

4.7 Student Information

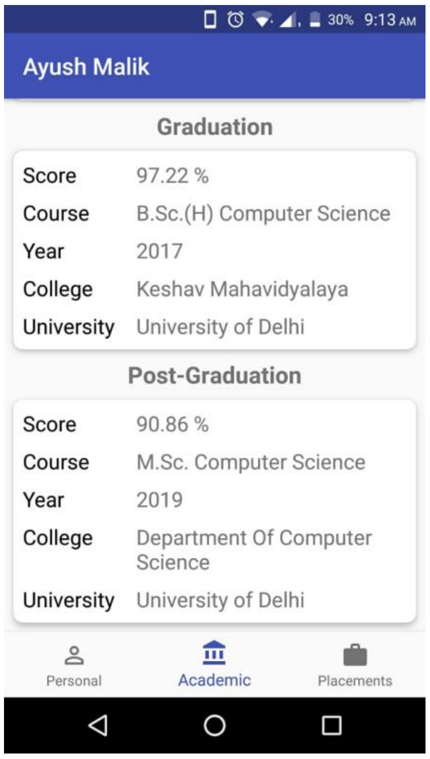


X

Score	9.8 CGPA
Subject	English,Hindi,Mathematics,Science,Social Science
Year	2012
School	St'Marks's Sr. Sec. Public School Meera Bagh, New Delhi, India
Board	Central Board of Secondary Education

XII

Score	94.3 %
Subject	Physics,Chemistry,Mathematics,Computer



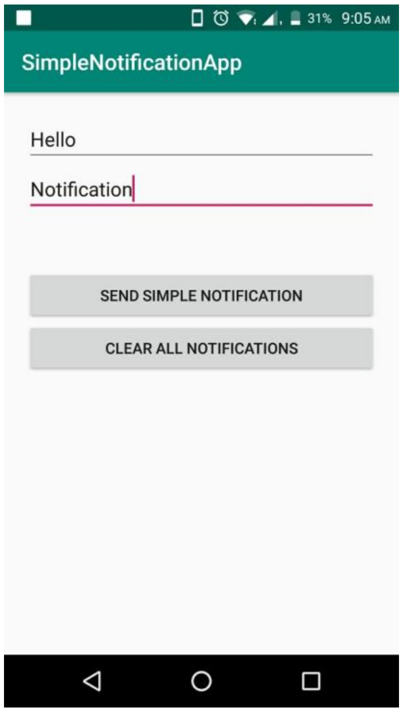
Graduation

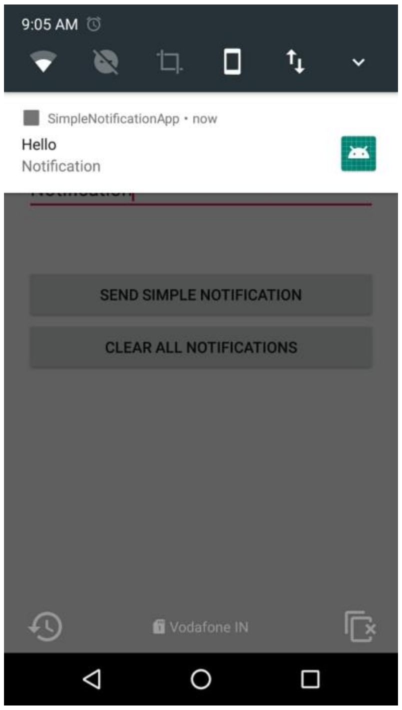
Score	97.22 %
Course	B.Sc.(H) Computer Science
Year	2017
College	Keshav Mahavidyalaya
University	University of Delhi

Post-Graduation

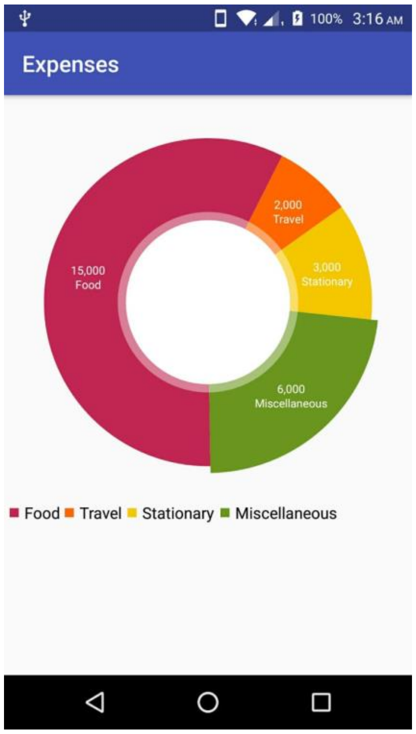
Score	90.86 %
Course	M.Sc. Computer Science
Year	2019
College	Department Of Computer Science
University	University of Delhi

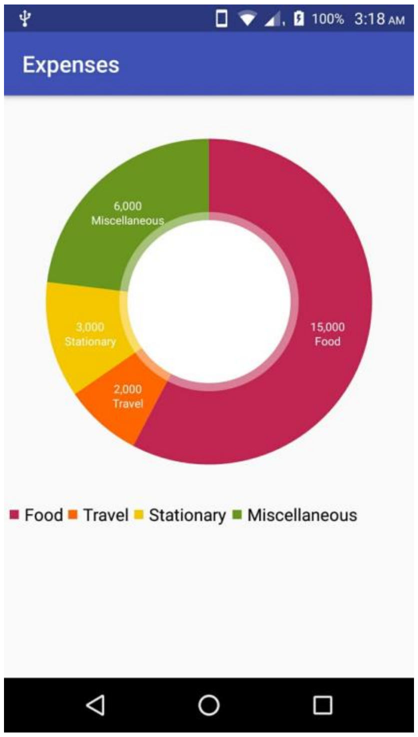
4.8 Local Notification Generator



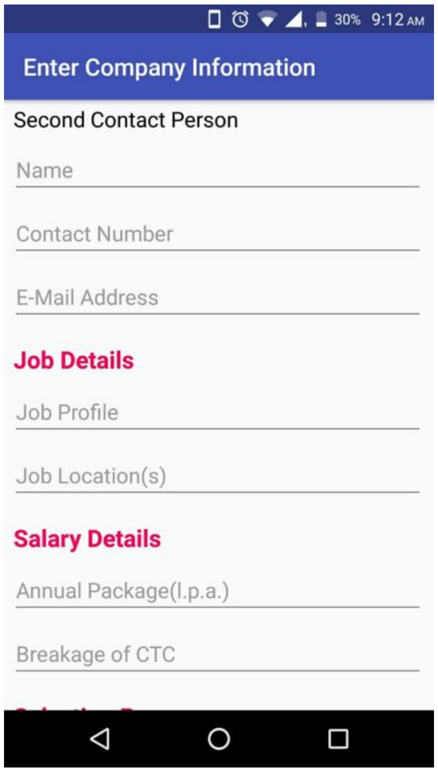


4.9 Expenditure





4.10 Adding a Company



9:12 AM

Enter Company Information

Second Contact Person

Name

Contact Number

E-Mail Address

Job Details

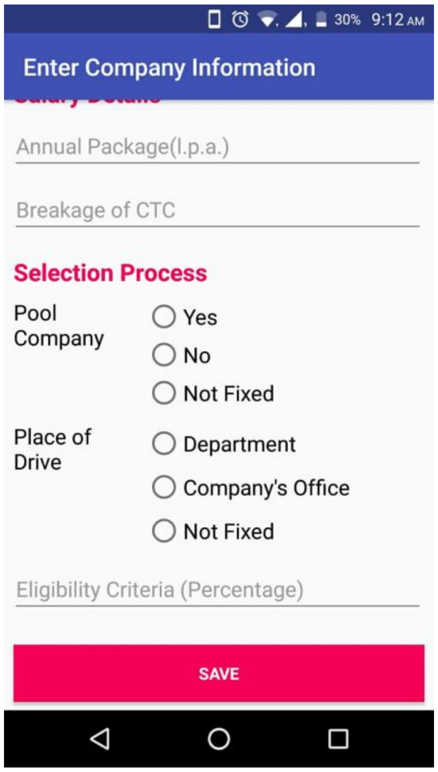
Job Profile

Job Location(s)

Salary Details

Annual Package(l.p.a.)

Breakage of CTC



9:12 AM

Enter Company Information

Annual Package(l.p.a.)

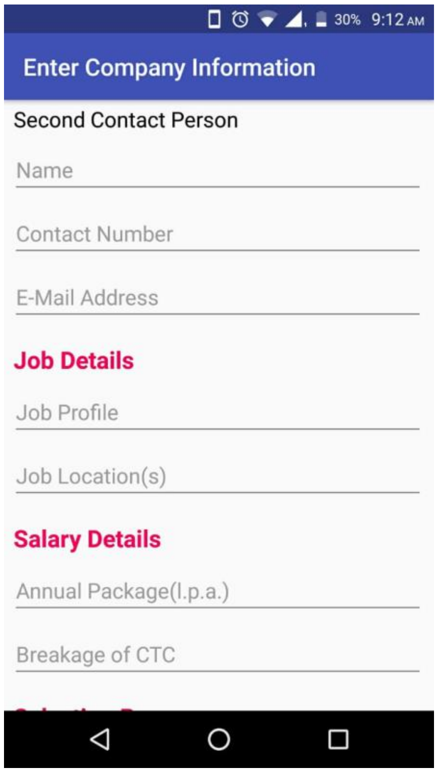
Breakage of CTC

Selection Process

Pool Company	◯ Yes ◯ No ◯ Not Fixed
Place of Drive	◯ Department ◯ Company's Office ◯ Not Fixed

Eligibility Criteria (Percentage)

SAVE



9:12 AM

Enter Company Information

Second Contact Person

Name

Contact Number

E-Mail Address

Job Details

Job Profile

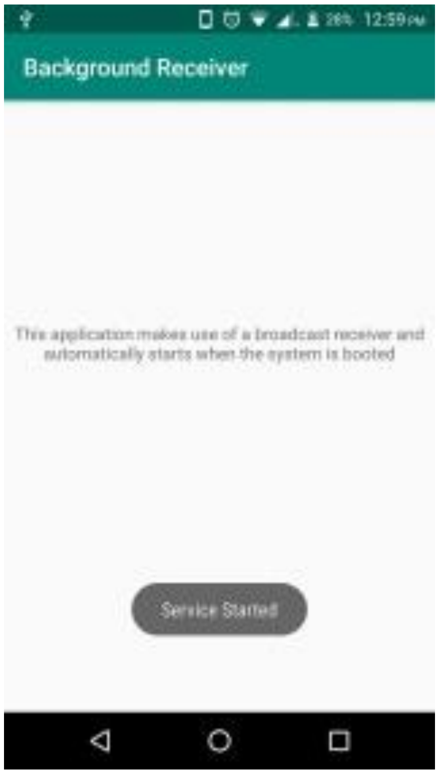
Job Location(s)

Salary Details

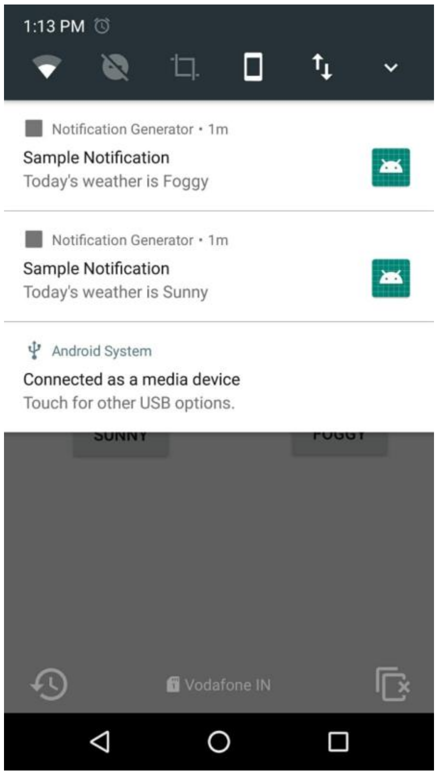
Annual Package(l.p.a.)

Breakage of CTC

4.11 Announcement Architecture Design



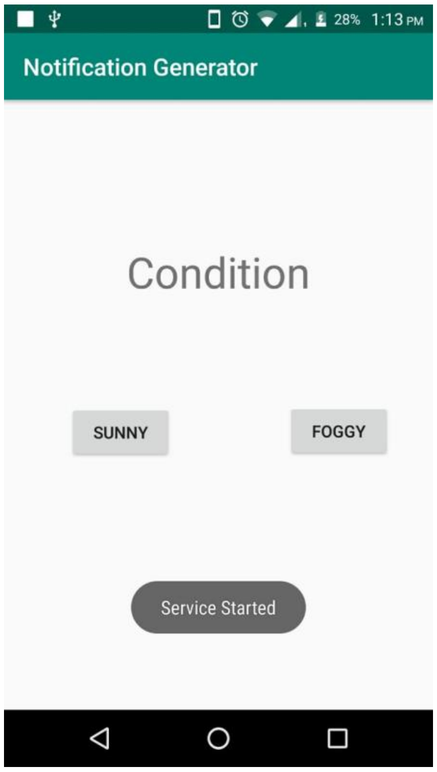
This application makes use of a broadcast receiver and automatically starts when the system is booted

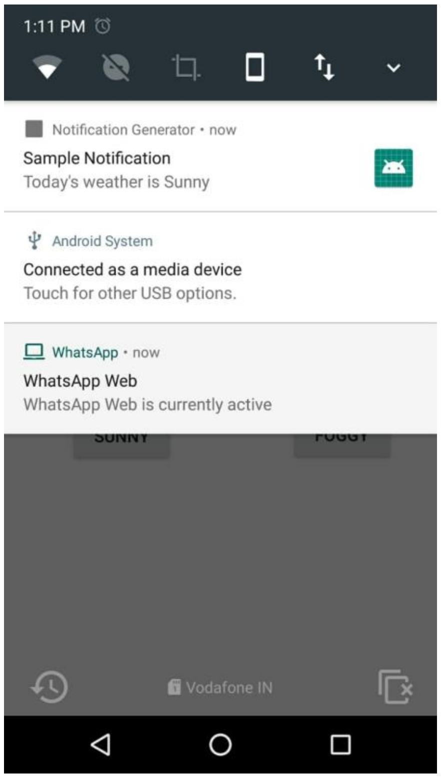

1:13 PM

Notification Generator • 1m
Sample Notification
Today's weather is Foggy

Notification Generator • 1m
Sample Notification
Today's weather is Sunny

Android System
Connected as a media device
Touch for other USB options.



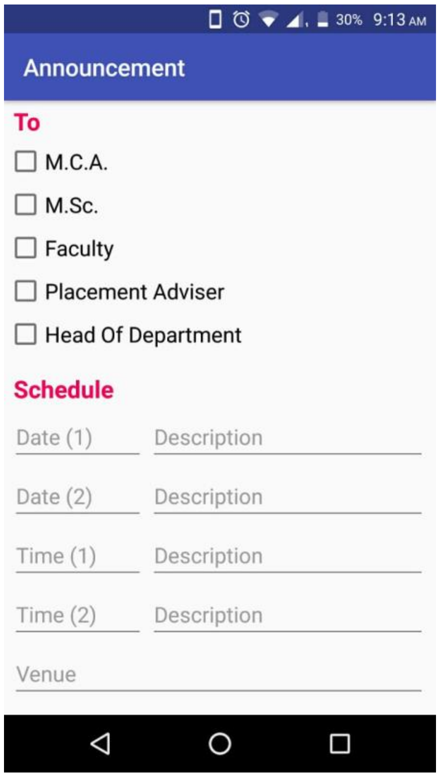

1:11 PM

Notification Generator • now
Sample Notification
Today's weather is Sunny

Android System
Connected as a media device
Touch for other USB options.

WhatsApp • now
WhatsApp Web
WhatsApp Web is currently active

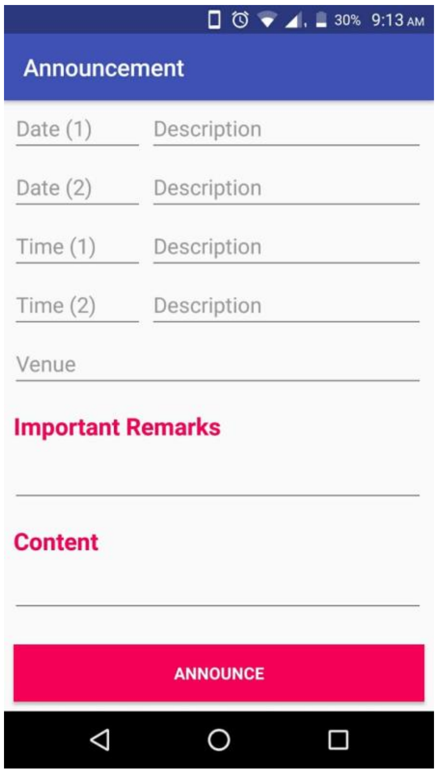
4.12 Announcements



To
- ☐ M.C.A.
- ☐ M.Sc.
- ☐ Faculty
- ☐ Placement Adviser
- ☐ Head Of Department

Schedule

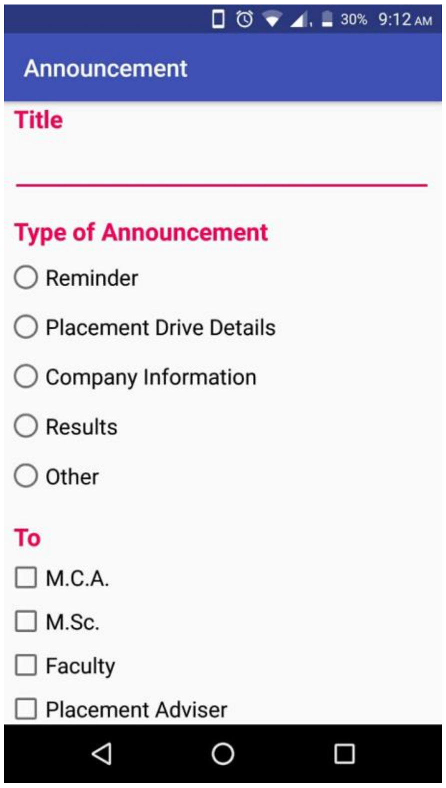
Date (1)	Description
Date (2)	Description
Time (1)	Description
Time (2)	Description
Venue	



Date (1)	Description
Date (2)	Description
Time (1)	Description
Time (2)	Description
Venue	

Important Remarks

Content

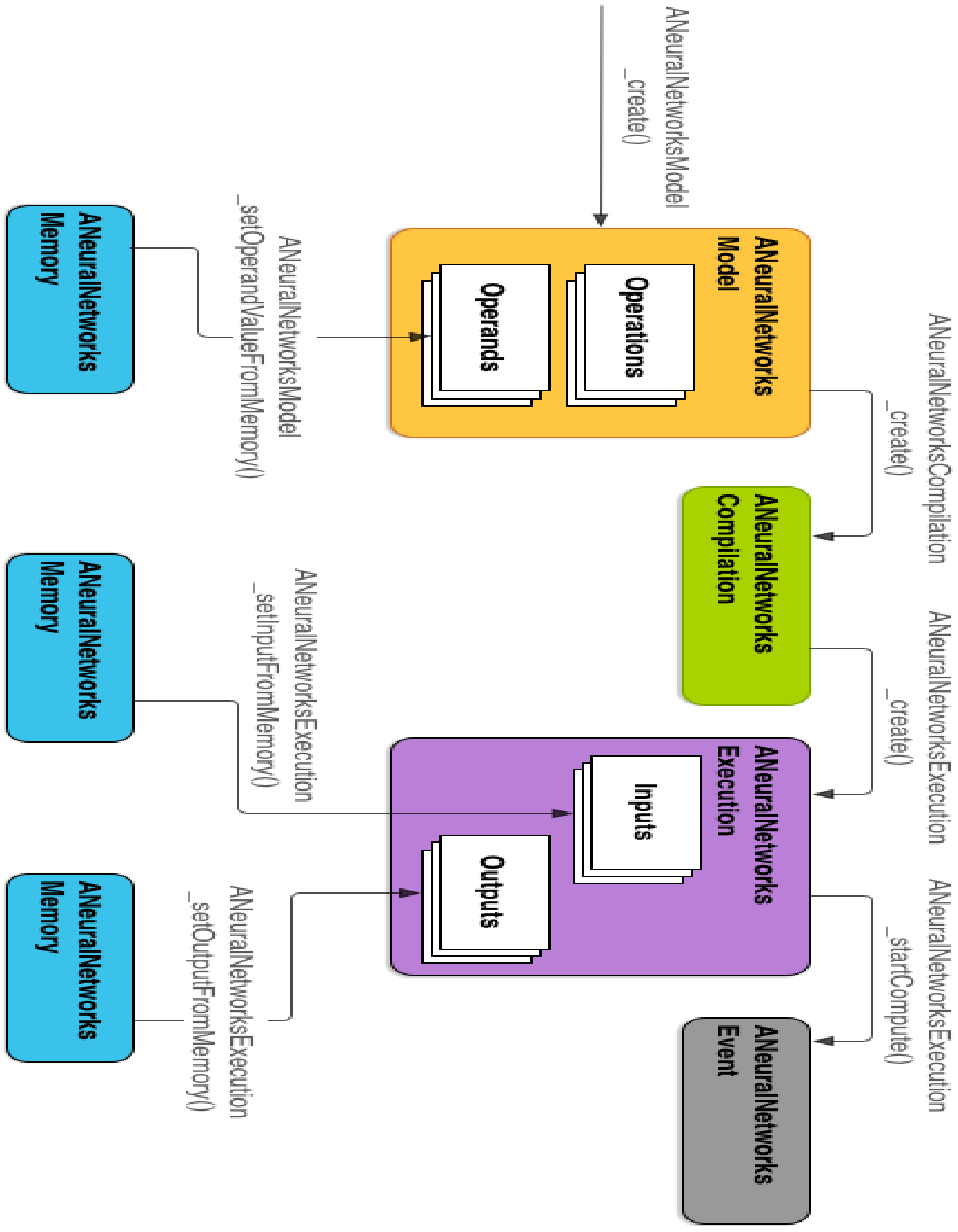


Title

Type of Announcement
- ○ Reminder
- ○ Placement Drive Details
- ○ Company Information
- ○ Results
- ○ Other

To
- ☐ M.C.A.
- ☐ M.Sc.
- ☐ Faculty
- ☐ Placement Adviser

4.13 Artificially Intelligent Company Predictor


ANeuralNetworksModel_create()
ANeuralNetworks Model
Operations
Operands
ANeuralNetworksModel_setOperandValueFromMemory()
ANeuralNetworks Memory
ANeuralNetworksCompilation_create()
ANeuralNetworks Compilation
ANeuralNetworksExecution_create()
ANeuralNetworks Execution
Inputs
Outputs
ANeuralNetworksExecution_setInputFromMemory()
ANeuralNetworks Memory
ANeuralNetworksExecution_setOutputFromMemory()
ANeuralNetworks Memory
ANeuralNetworksExecution_startCompute()
ANeuralNetworks Event

4.14 Setting Up Broadcast Receiver

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. These broadcasts are sent when an event of interest occurs. In this module, our event of interst was BOOT_COMPLETED. We registered a broad cast receiver which starts a service whenever the system is booted.

Activities

1.	MainActivity	This activity is called using context.startActivity( ) whenever the system is booted.

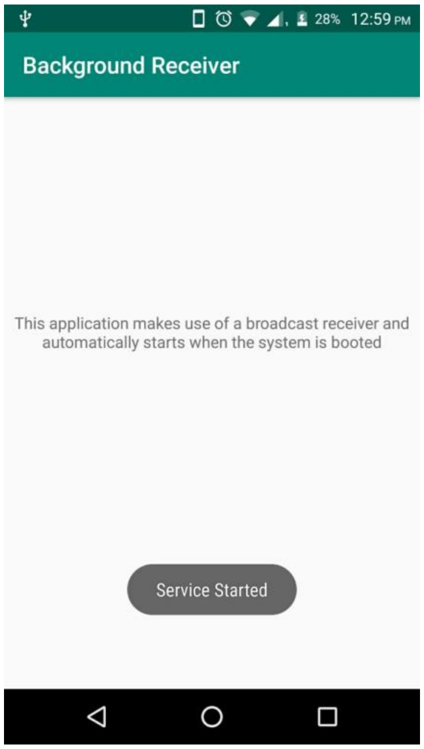
Layout Files

1.	activity_main.xml	The layout file associated with MainActivity specifying using a textView that the application is started.

Classes

1.	MainActivity.java	Java class associated with MainActivity.
2.	MyBroadcastReceiver.java	The class defines the onReceive ( ) function of the broadcast receiver of the application.
3.	ServiceNoDelay.java	This service is started by the broadcast receiver on boot completion using context.startService( ).

Screenshots



This application makes use of a broadcast receiver and automatically starts when the system is booted

5 . CONCLUSION

In this project, we have built an Android application for handling placements in Department Of Computer Science, University of Delhi.

The application has been designed keeping in mind time, cost and knowledge constraints.

It has some *drawbacks* and *shortcomings* :

- It doesn’t take into consideration the people who don’t use Android smartphones.
- It is highly dependent on Firebase Services.
- It doesn’t provide service to the Students and Faculties to ask and answer queries.
- More testing is required.
- Many ease of use functionalities have been omitted because of time-constraints.

We have designed the code and the application in such a manner that it can be worked upon in future by other students.

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

6 . REFERENCES

1. **Android Developers**

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

2. **Google Firebase**

   - https://firebase.google.com/docs/
   - https://firebase.googleblog.com/2016/08/sending-notifications-between-android.html

3. **YouTube**

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

4. **Tutorials Point**

   - https://www.tutorialspoint.com/android/

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

6. **Other Online Resources**

   - https://developers.google.com/gmail/api/
   - https://software.intel.com/en-us/android/articles/implementing-map-and-geofence-features-in-android-business-apps
   - https://www.cleveroad.com/blog/realm-vs-sqlite-what-is-the-best-database-for-android-app-development
   - http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html