A PROJECT REPORT ON

NGO CONNECT: AN ANDROID APPLICATION USING CLOUD BASED SERVICE AND LOCATION BASED SERVICE

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILMENT FOR THE AWARD OF DEGREE

OF

BACHELORS OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

ALI ASGHAR

MADHURI AWACHAR

BALAJI DAVANGAVE

ROHAN GAIKWAD

UNDER THE GUIDANCE OF

B.P.VASGI

Sinhgad Institutes

Sinhgad College of Engineering, Pune
INFORMATION TECHNOLOGY – 2018-19

DEPARTMENT OF INFORMATION TECHNOLOGY

SINHGAD COLLEGE OF ENGINEERING, PUNE

VADGOAN (BK), PUNE 411041

2018-19

CERTIFICATE

This is to certify that the project report entitled

NGO Connect: An Android Application using Cloud Based Service and Location based Service

Submitted by

ALI ASGHAR
MADHURI AWACHAR
BALAJI DAVANGAVE
ROHAN GAIKWAD

Is a bonafide work carried out by them under the supervision of Prof. B.P. Vasgi and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University for the award of the Degree of Bachelor of Engineering (Information Technology)

This project report has not been earlier submitted to any other Institute or University for the award of any degree or diploma.

Prof. B.P.VASGI	Prof. G.R.PATHAK
Internal Guide	Head of Department
Department of Information Technology	Department of Information Technology

Dr. S.D.LOKHANDE

External Examiner — Principal

Date: — SCOE

Place: Pune

ACKNOWLEDGEMENT

I feel great pleasure in expressing my deepest sense of gratitude and sincere thanks to my guide **Prof. B.P.Vasgi** for the valuable guidance during the Project work, without which it would have been very difficult task. I have no words to express my sincere thanks for valuable guidance, extreme assistance and cooperation extended to all the **Staff Members** of my Department.

This acknowledgement would be incomplete without expressing my special thanks to **Prof. G.R.Pathak** Head of the Department (Information Technology) for their support during the work. I would also like to extend my heartfelt gratitude to my Principal, **Dr. S.D.Lokhande** who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

Last but not least I would like to thanks all the Teaching, Non- Teaching staff members of my Department, my parent and my colleagues those who helped me directly or indirectly for completing of this Project successfully.

Name of Students

Ali Asghar

Madhuri Awachar

Balaji Davangave

Rohan Gaikwad

ABSTRACT

Now a days, we see most of the people want to serve the nation, but due to their busy schedule they can't do work for NGO all the time. But they do want to work for these NGOs as and when they get time, but the problem is that they can't find a way to connect with these NGOs. This application will help people find NGOs and work for them on holidays. It will also be helpful for NGOs to find volunteers as and when required.

Many a times, we see, there is huge waste of food in restaurants, hotels and hostel messes, and on the other hand many people couldn't get food. This application will help these hotels and restaurants to connect with NGOs and help these poor people get food.

By using Haversine method, we can calculate the distance of NGOs from the restaurant and can be able to allocate the restaurant to NGOs. We can store the data of interested persons, restaurants, NGOs on Cloud.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

Developing Android application using cloud based data storage and using location based service to locate NGOs, volunteers and restaurants so that they can work together in the process of recruiting volunteers and reducing wastage of food.

1.2 AIM

People wants to contribute to the society but they did not get the time and when they get the time on weekend or on holiday they have no idea how to contribute and where to contribute. Many times in mess/restaurant or in ceremony a huge amount of food get wasted.

So, we are developing an android application in which we will design the user interface through which people who are willing to contribute to the society can register and NGOs with their domain will be register to the application as NGOs will hire the willing people according to their demand or need. Restaurant or mess will also register to the application. If restaurant and owner wants to donate food they can just login to the application and application will locate the NGOs who are willing to take the food.

We are using location based service to locate the NGOs and restaurant and also using Google Firebase cloud service to store and fetch the data.

1.3 OBJECTIVE

An android application which will add the willing people, NGOs, and restaurant. Willing People can register to the application and browse the NGOs list. People can select the NGOs according to the domain of NGOs and can send the request and it depends on the NGOs to accept the request of willing people or not. If NGO will reject the request then people can search another NGO and

send the request. Restaurant owner can also search NGOs to donate the remaining food as to avoid wastage of food.

1.4 MOTIVATION

As smart phone is getting very handy these days so we are developing android application which can work on smart phone having android as an operating system whose version is 5.0 or upgraded version.

It provides a way to the people so that they can connect with NGOs on adhoc or on regular basis. It also provide a solution for food wastage. It can be helpful for gathering the volunteer in situations like earthquake, flood. It will be helpful to the government for tracking the records of NGOs.

1.5 BACKGROUND AND BASICS

1.5.1 CLOUD COMPUTING

Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

A cloud service has three distinct characteristics that differentiate it from traditional web hosting. It is sold on demand, typically by the minute or the hour; it is elastic -- a user can have as much or as little of a service as they want at any given time; and the service is fully managed by the provider (the consumer needs nothing but a personal computer and Internet access). Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet, have accelerated interest in cloud computing.

Types of cloud computing services:

IaaS supply a virtual server instance and storage, as well as APIs that enable users to migrate workloads to a VM.

In the PaaS model, cloud providers host development tools on their infrastructures.

SaaS is a distribution model that delivers software applications over the internet; these applications are often called web services.

Cloud computing boasts several attractive benefits for businesses and end users. Five of the main benefits of cloud computing are:

- **Self-service provisioning:** End users can spin up compute resources for almost any type of workload on demand. This eliminates the traditional need for IT administrators to provision and manage compute resources.
- **Elasticity:** Companies can scale up as computing needs increase and scale down again as demands decrease. This eliminates the need for massive investments in local infrastructure, which may or may not remain active.
- **Pay per use:** Compute resources are measured at a granular level, enabling users to pay only for the resources and workloads they use.
- **Workload resilience:** Cloud service providers often implement redundant resources to ensure resilient storage and to keep users' important workloads running -- often across multiple global regions.
- **Migration flexibility:** Organizations can move certain workloads to or from the cloud or to different cloud platforms as desired or automatically for better cost savings or to use new services as they emerge.

1.5.2 LOCATION BASED SERVICES

A location-based service (LBS) is the name for a general class of policies in software-level services that provide for accessing data, files, pipes, memory objects, streams and other or online services. Access policies are controlled by location data and/or time-of-day constraints, or a combination thereof. As such, an LBS is an information service and has a number of uses in social networking today as information, in entertainment or security, which is accessible with mobile devices through the mobile network and which uses information on the geographical position of the mobile device.

LBS can be used in a variety of contexts, such as health, indoor object search, entertainment, work, personal life, etc.

LBS include services to identify the location of a person or object, such as discovering the nearest cash machine (ATM) or the whereabouts of a friend or employee. LBS include parcel tracking and vehicle tracking services. LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location. They include personalized weather services and even location-based games. They are an example of telecommunication convergence.

Google Map API:

The Maps JavaScript API lets us customize maps with our own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite, hybrid, and terrain) which we can modify using layers and styles, controls and events, and various services and libraries.

When loading the Maps JavaScript API via the URL you may optionally load additional libraries through use of the libraries URL parameter. Libraries are modules of code that provide additional functionality to the main Maps JavaScript API but are not loaded unless you specifically request them. For more information, see Libraries in the Maps JavaScript API.

CHAPTER 2

LITERATURE SURVEY OF NGO CONNECT ANDROID APPLICATION

A literature survey in a project report is that section which shows the various analysis and research made in the field of the project interest and the result already published, taking into the account the various parameter of the project and extent of the project.

Paper 1:

S. K. Tiwari, G. K. Varshney, M. A. Qadeer and M. S. Umar: Intelligent LBS using Android with GPS and Geo-Tagging Applications.

JustQuick is an android based application that will provide details about the user's nearby services like ATM, bank, cafe, hospital, shops, etc. of a particular city. It will also provide a path from the user's current location to that particular services' location. Information provided by app from our own database. We also provide permission to
authorized users to insert a new object to our database and able to modify a specific field at regular interval.

- Retrieve information from server.
- Insert And Update information available on server.

Paper 2:

E. Winarno, W. Hadikurniawati and R. N. Rosso, Location based service for presence system using haversine method.

Location-Based Services (LBS) stands for the provision of information about the position of a device or a user often offered as a service via various means of media. Haversine formula is the method used to calculate the distance from two different points. In this research, Haversine method applied on calculating the distance for presence system based on the distance of location. This system allows users in presenting using mobile applications in addition to facilitate the

presenting of data preservation. Use of LBS technology to detect users in an area of allowable locations. This detection is done by looking at the closest point from the predetermined center point. Center point will store the coordinates and radius of the presence area.

- Haversine method for measurement and estimation of distance with limitation in localization.
- Implementation model includes signal measurements with wifi position and GPS values to identify the precise location for human presence system.

Paper 3:

W. Li, C. Yen, Y. Lin, S. Tung and S. Huang: JustIoT Internet of Things based on the Firebase Real-time Database.

This paper designs an Internet of Things system which is mainly divided into four parts; Back-end Google Firebase real-time database, front-end SPA (Single Page Application) web monitoring program (including mobile monitoring App), controller software-hardware, and Intelligence server that support MQTT connection and condition control.

- The use of the Firebase cloud system begins with enabling the user authentication and planning database. JustIoT uses email/password authentication.
- Firebase real-time database is a NoSQL-type database. In order to reduce traffic, the structure of the database is designed as flat and does not conform to the normalization required by the general database.
- Before using the Firebase real-time database, user must set up access rules according to the user's database plan. The settings are somewhat complicated, and the Bolt compiler is used to assist the user by setting the access rules in a simple syntax.

Paper 4:

Sandeep Kumar, Mohammed Abdul Qadeer, Archana Gupta: Location Based Services using Android

Mobile computing is researched as the combination of computing and mobile communication technology. Mobile communication gradually changes its trends from simple voice communications to complex data communications. Short messages and mobile internet business have been widely used in China, which are the best cases.

Android provides access to the following components to facilitate the implementation of LBS services:

1. Location Manager
2. Location Provider
3. Geocoding
4. Google-Map

Paper 5:

Pradnya Battin, Dr. S.D.Markande: Location Based Reminder Android Application Using Google Maps API

To remind modern people of something at a specific time and location, Smart Location Reminder is a boon . To serve the purpose, implementing an application for Android-based Smart phones and tablets which is not only time based but also location based. The system uses free public API service from Google Maps.

The smartphone contains built-in GPS receiver which receives signal from GPS receiver. The application performs Geolocationing based on GPS reading to detect present location of the user. The desired locations & tasks are stored in the database. If task to be reminded is available in the database, after that application will perform comparison of the location which is identified with the location related with the desired task.

Table 2.1: Literature Survey

S.No	Title	Year	Author Name	Description
1	Intelligent LBS using Android with GPS and Geo-Tagging Applications	2016	S. K. Tiwari, G. K. Varshney, M. A. Qadeer and M. S. Umar	Retrieve information from server, Insert And Update information available on server
2	JustIoT Internet of Things based on the Firebase Real-time Database	2018	W. Li, C. Yen, Y. Lin, S. Tung and S. Huang	The use of the Firebase cloud system begins with enabling user authentication and planning database
3	Location Based Services using Android	2009	Sandeep Kumar, Mohammed Abdul Qadeer, Archana Gupta	Location-based service (LBS) provides a user with contents customized by the user's current location
4	Location based service for presence system using haversine method	2016	E. Winarno, W. Hadikurniawati and R. N. Rosso	Haversine formula is the method used to calculate the distance from two different points
5	Location Based Reminder Android Application Using Google Maps API	2016	Pradnya Battin, Dr. S.D.Markande	Uses free public API service from Google Maps

CHAPTER 3

SPECIFICATIONS OF NGO CONNECT ANDROID APPLICATION

A Project Specification (or spec) is a comprehensive description of objectives and the requirements for a development project. It contains all goals, functionality, and details required for a development team to fulfill the vision of the client.

3.1 Software and Hardware requirements

3.1.1 Hardware Requirements

The proposed system must satisfy the following hardware requirements.

1. RAM: 8 GB recommended and 1 GB for Android Emulator
2. Hard Disk: 512 MB or higher

3.1.2 Software Requirements

A Software Requirements describes the nature of a project, software or application. These software requirements along with its specifications are present in a document which is known as SRS report or software document. The proposed system must satisfy the following software requirements.

1. OS: Android 5.0(Lollipop) or higher
2. Java Development Kit
3. Windows7 or later

3.2 Functional and Non-functional requirements

3.2.1 Functional Requirements

Functional requirements include various functions performed by specific screens, outlines of workflows performed by the system. Following are the functional requirements of the proposed system:

- NGOs, volunteers and restaurants will be able to upload their details on Firebase cloud
- NGOs will be able to upload the details of vacancies in volunteering
- NGOs could upload the images explaining the nature of work, also volunteers will be uploading their profile pictures while creating the account
- Restaurants as well as volunteers will be able to upload the details of the food to be donated
- NGOs will be able to find nearby food donors and choose among them
- Volunteers will be able to search NGOs according to the distance and the choice of domain

3.2.2 Non-Functional Requirements

Software Quality Attributes:

Quality attributes are the overall factors that affect run-time behavior, system design, and user experience. They represent areas of concern that have the potential application wide impact across layers and tiers. Some of these attributes are related to the overall system design, while others are specific to run time, design time, or user centric issues. The extent to which the application possesses a desired combination of quality attributes such as usability, performance, reliability, and security indicates the success of the design and the overall quality of the software application.

a.) Reusability:

Reusability defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time.

b.) Availability:

Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.

c.) Performance:

Performance is an indication of the responsiveness of a system to execute any action within a given time interval. It can be measured in terms of latency or throughput.

d.) Reliability:

Reliability is the ability of a system to remain operational over time. Reliability is measured as the probability that a system will not fail to perform its intended functions over a specified time interval.

e.) Scalability:

Scalability is ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged. Scalability describes the ability of a process, network, software or organization to grow and manage increased demand. A system, business or software that is described as scalable has an advantage because it is more adaptable to the changing needs or demands of its users or clients

f.) Testability:

Testability is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if the criteria are met. Good testability makes it more likely that faults in a system can be isolated in a timely and effective manner.

g.) Usability:

Usability defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize, providing good access for disabled users, and resulting in a good overall user experience.

3.3 Security Requirements

Requirements concerning confidentiality, integrity, availability, and quality are basic requirements are called as security requirements. Security requirements for the proposed system are as follows:

Firebase Security Services

We are using Firebase Security Services to secure the data from getting misused by the attackers. Using this security measure, unless a user gets logged into the system, he/she will not be able to view or edit the data present on Firebase cloud. Moreover, Firebase provide secured authentication, which ensures the genuineness of the user by their email id.

Security rules are managed by a single JSON object that we can edit on our Realtime Database console or using the Firebase CLI. And as a handy bonus, the console and the CLI will warn us if our rules are malformed.

This data starts with four top-level nodes:

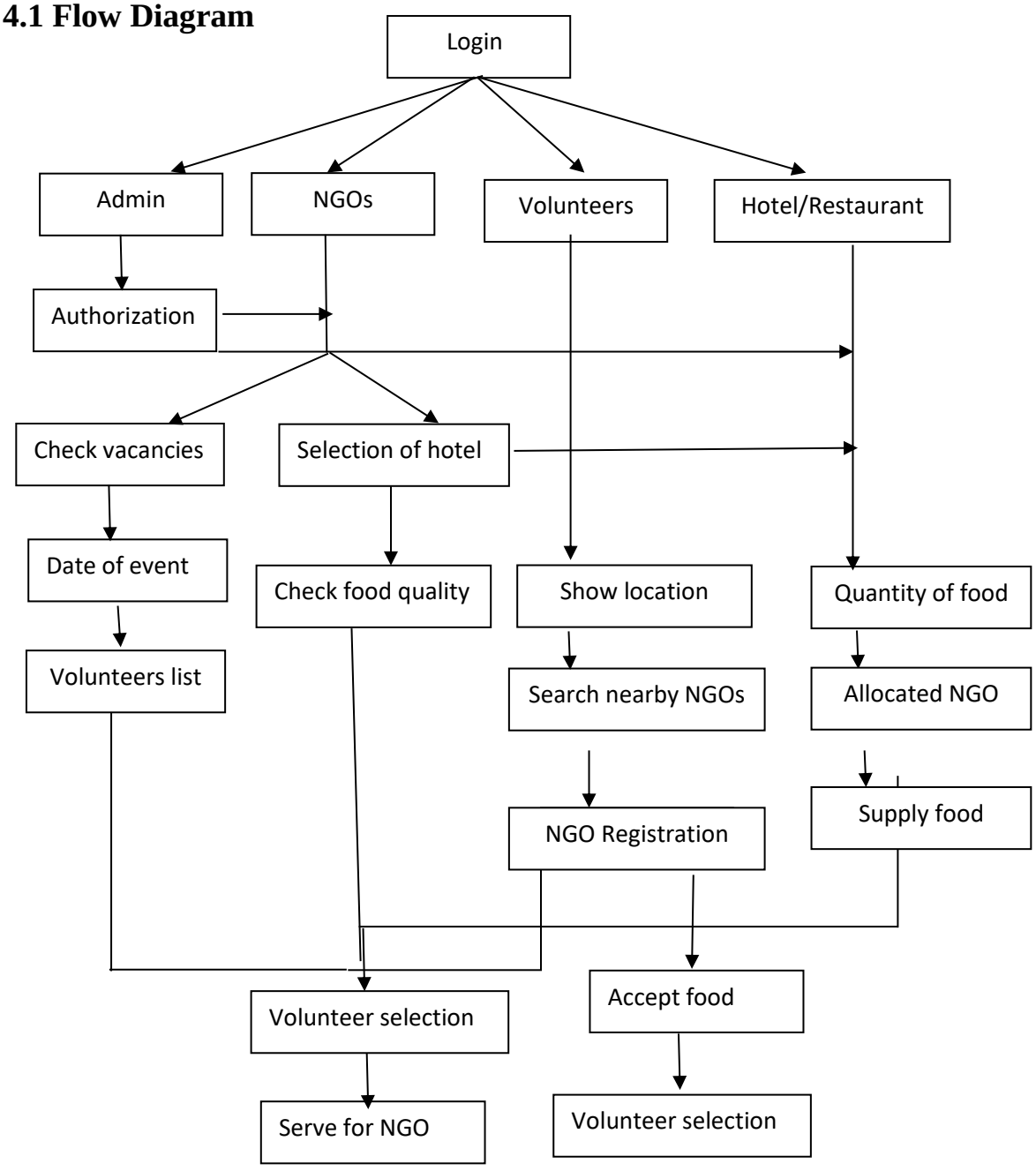
1. users
2. userReadable
3. userWriteable
4. userOwned

CHAPTER 4

DESIGN OF NGO CONNECT ANDROID APPLICATION

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

4.1 Flow Diagram



4.2 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating. Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior.

In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them.

The following is the use case diagram for our system. The actors involved in the system are Admin, Volunteers, and the owner of hotel/restaurant. The use cases associated with them are nothing but are the operations performed by them.

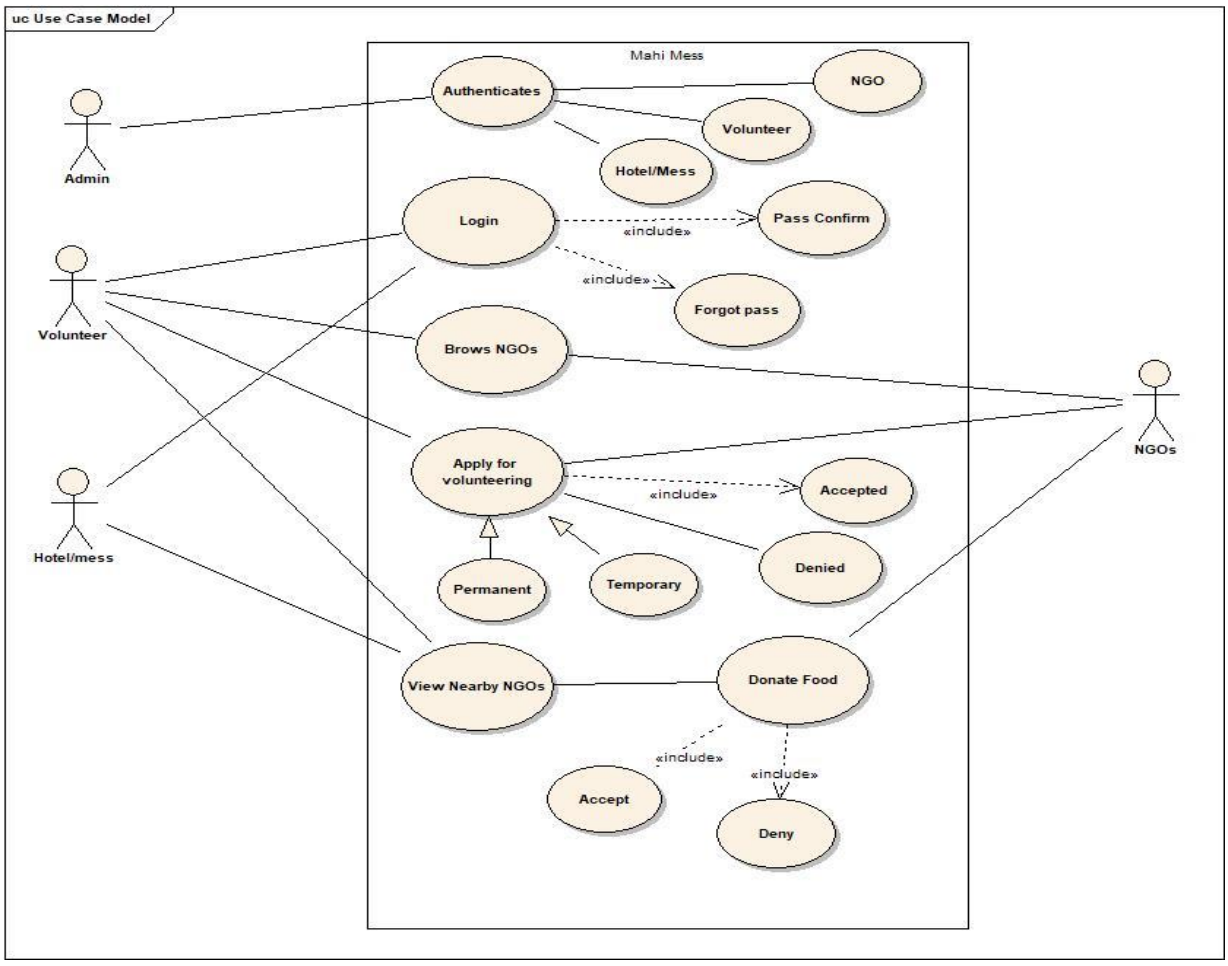


Fig. 4.1 Use Case Diagram

4.3 Activity Diagram

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. Control flow is drawn from one operation to another. Basic workflow of our system is as shown in below figure.

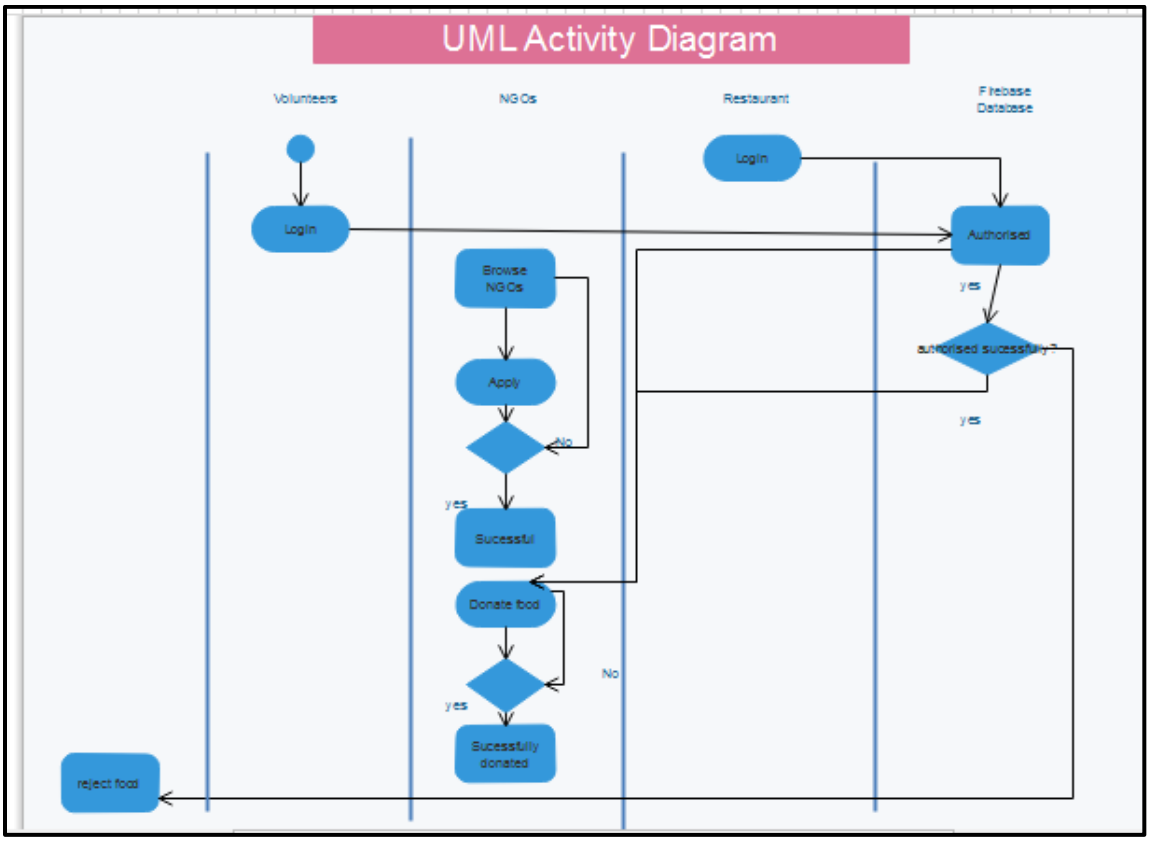


Fig. 4.2 Activity Diagram

Workflow of our system starts with outsourcing the data which will be in encrypted form to the cloud server. This is done by the Owner. The owner also generates the searchable index file which is also uploaded to the cloud server. The owner also sends the decryption key to the User, because the data received by the user will be in encrypted format which needs to be decrypted

Now, the cloud Server has the searchable index file which is uploaded by owner. It also receives the query in encrypted format, which is fired by user. Then, it will generate the intermediate results for the query which will be passed to the user, from where the final generation of result would take place.

Once the cloud server has intermediate results for the query and the encrypted data ready with it, it will compose the final results. These results will also be in encrypted format which will be given to the user.

The user fires the encrypted query. Also it receives the final result from cloud server, which needs to be decrypted by the decryption key. It has the decryption key with it, and thus user decrypts the final result.

4.4 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. A collection of class diagrams represent the whole system.

Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

It describes the particular aspect of the entire application. The classes are connected to each other through association links.

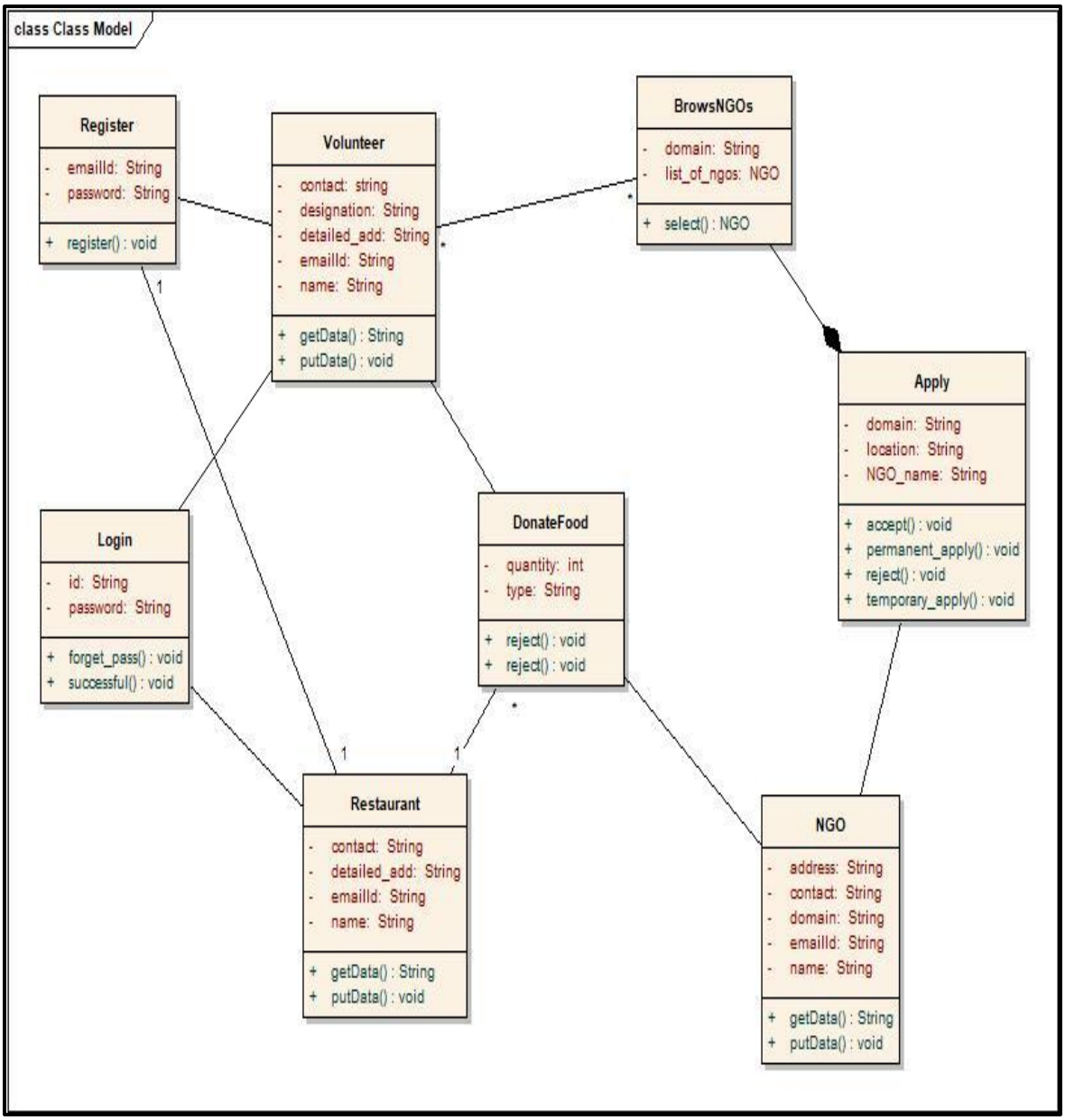


Fig . 4.3 Class Diagram

The owner and user are associated with the cloud server. The owner is also associated with the cloud server with one as to many relationship.

Various attributes and the operations for the different classes are also mentioned into the class diagram. Thus, the static view for the system is visualized and further it can be used for generating executable code.

4.5 State Diagram

A State diagram shows a state machine. Usually the state machine in a state diagram models the behavior of a reactive object, whose behavior is best characterized by its response to events dispatched from outside its context. The object has a clear lifetime whose current behavior is affected by its past. State diagrams are important for constructing executable systems through forward and reverse engineering.

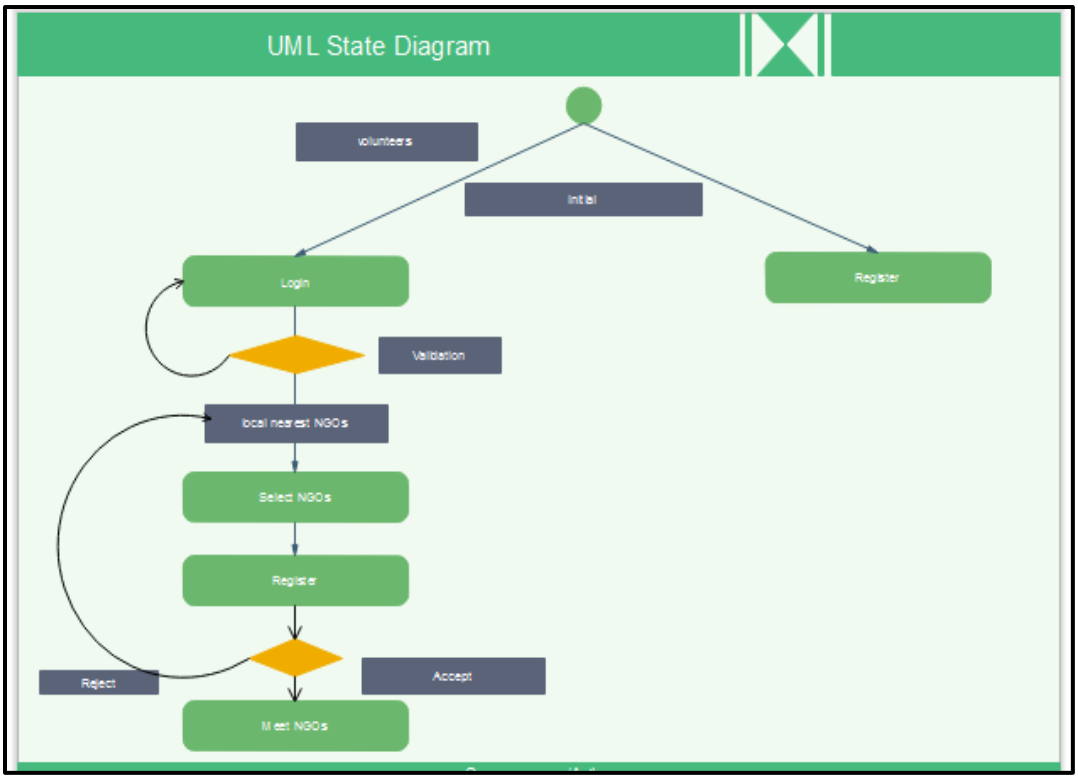


Fig . 4.4 State Diagram

The figure shows the state diagram for information retrieval. It has four main states as:

1. Login
2. Register
3. Select NGO
4. Meet NGOs

4.6 Sequence Diagram

A Sequence diagramshows,as parallel vertical lines,different processes or objects that live simultaneously,and,as horizontal arrows, the messages exchanged between them,in the order in whichbthey occur.

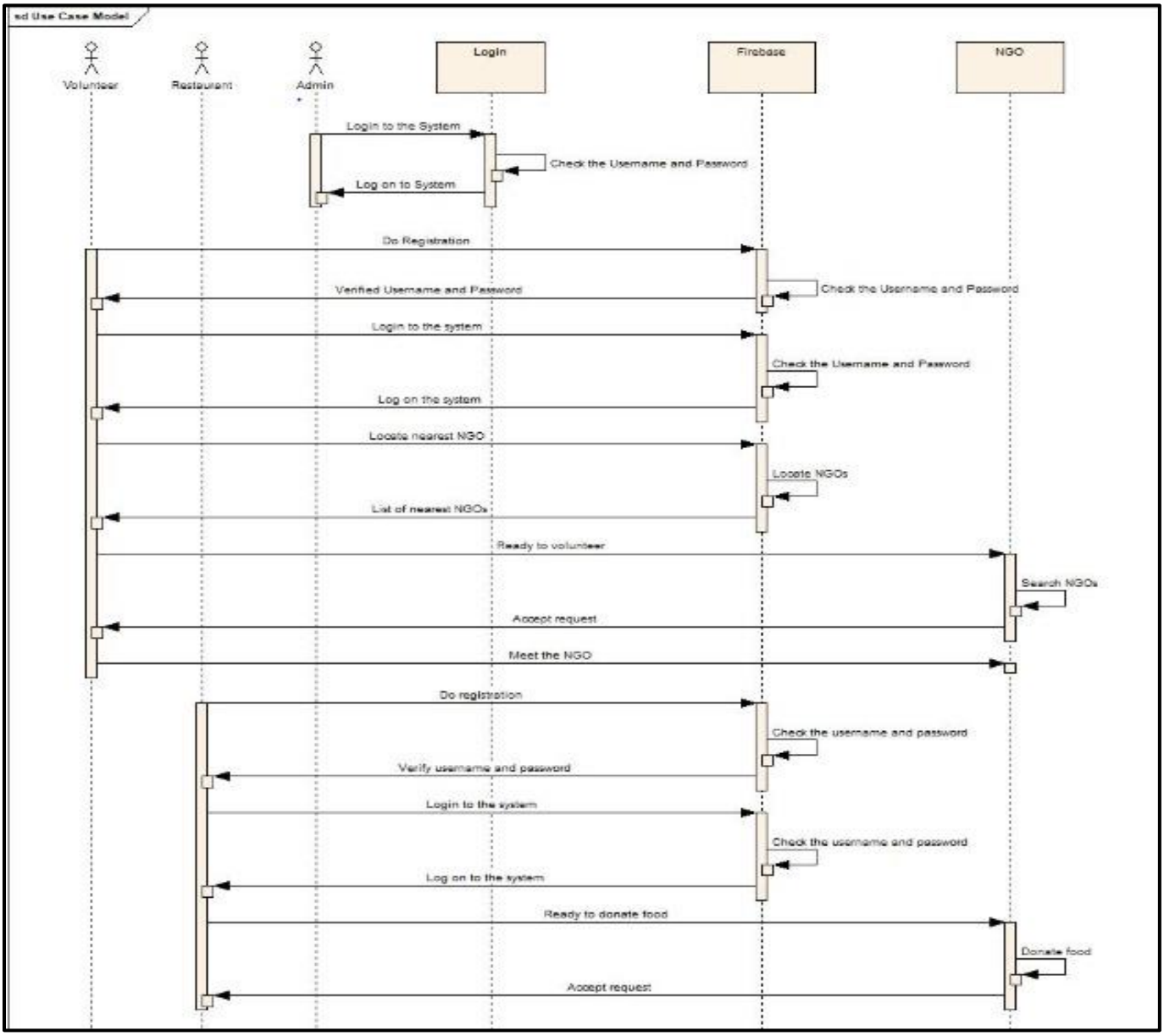


Fig . 4.5 Sequence Diagram

CHAPTER 5

Methodology Used For Implementation

Android application will fetch the user location and store it to the database that will be the cloud storage. In the database, there will be the location co-ordinates of all restaurants and NGOs which are registered on the application.

User will define the radius for the NGOs selection. Distance calculation method i.e. Haversine Algorithm will calculate the distance from user to every NGOs and list out the NGOs whose distance from the user is less than defined distance by the user.

User will select the NGOs according to the domain or the vacancy and send the request to NGOs for the adhoc or for the permanent basis. NGO will accept or deny the request according to their need.

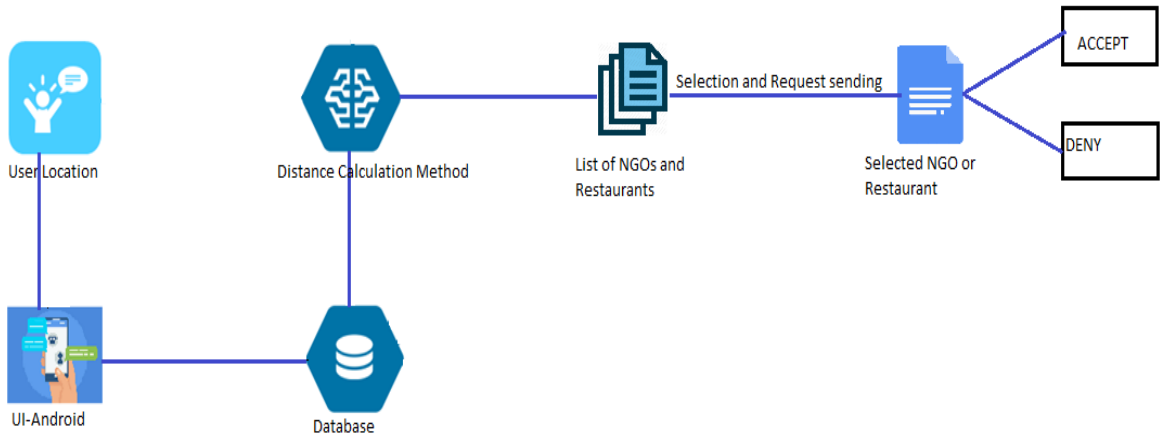


Fig. 5. Architecture

5.1 Android

Android is an operating system based on the Linux kernel. Android is developed in the Android Open Source Project (AOSP). This project is led by Google.

The Android operating system can be divided into the four areas as depicted in the following graphic. An Android application developer typically works with the two layers on top to create new Android applications.

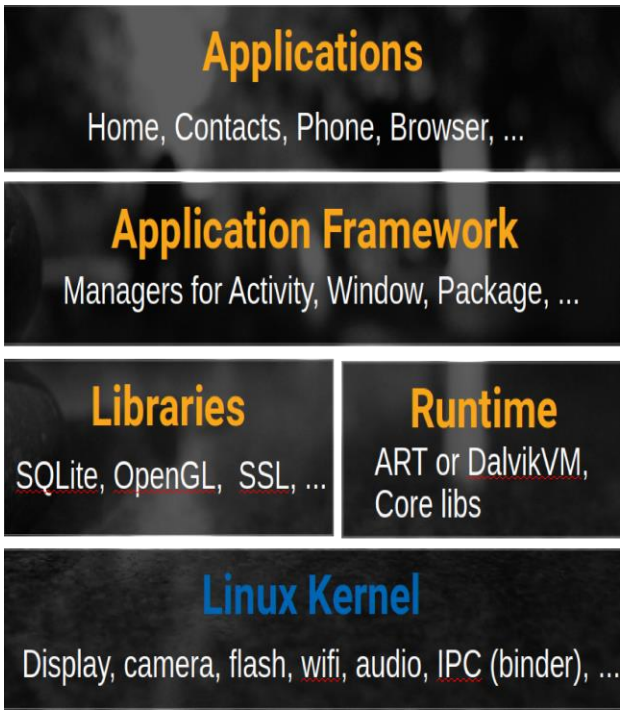


Fig 5.1.1 Architecture of Android

The levels can be described as:

- Applications - Contains the applications, like the Browser, Camera, Gallery, Music and Phone
- Application framework - An API which allows high-level interactions with the Android system
- Libraries and runtime - The libraries for many common framework functions, like graphic rendering, data storage, web browsing. Also contains the Android runtime, as well as the core Java libraries for running Android applications.
- Linux kernel - Communication layer for the underlying hardware.

5.1.1 Android Application

An Android application (app) is a single installable unit which can be started and used independently. An Android application consists of configuration files, Java source and resource files. We can define the components in our configuration files.

5.1.2 Components of Android:

1. **Activities:** It deals with the UI and the User interactions to the screen. In other words, it is a User Interface that contains activities. These can be one or more depending upon the App. It starts when the application is launched. At least one activity is always present which is known as MainActivity.
2. **Services:** Services are the background actions performed by the app, these might be long running operations like a user playing music while surfing the Internet. A service might need another sub-services so as to perform a specific tasks. The main purpose of the Services is to provide non-stop working of the app without breaking any interaction with the user.
3. **Broadcast Receivers:** A Broadcast is used to respond to the messages from other applications or from the System. For example, when the Battery of the phone is low, then the Android OS fires a Broadcasting message to launch Battery Saver function or app, after receiving the message the appropriate action is taken by the app. Broadcast Receiver is the subclass of BroadcastReceiver class and each object is represented by an Intent objects.
4. **Content Provider:** Content Provider is used to transfer the data from one application to the others on request of the other application. These are handled by the class ContentResolver class. These class implements a set of APIs(Application Programming Interface) that enables the other applications to perform the transactions. Any Content Provider must implement the Parent Class of ContentProvider class.

5.1.3 Structure of an application:

1. **Android Mainfest** is a XML file which is the root of the project source set. It describes the essential information about the app and the Android build tools, the Android Operating System. It contains the permission that an app might need in order to perform the specific task. It also includes the special activities like services, broadcast receiver, content providers, package name,etc.

2. **The JAVA folder** consist of the java files that are required to perform the background task of the app. It consist of the functionality of the buttons, calculation, storing, variables, toast(small popup message) , programming function, etc. The number of these files depends upon the type of activities created.
3. **Res** or Resource folder consist of the various resources that are used in the app. This consist of sub-folders like drawable, layout, mipmap, raw and values. The drawable consist of the images.
4. **Gradle**: Gradle is an advance toolkit, which is used to manage the build process, that allows to define the flexible custom build configurations. Each build configuration can define its own set of code and resources, while reusing the parts common to all versions of your app.

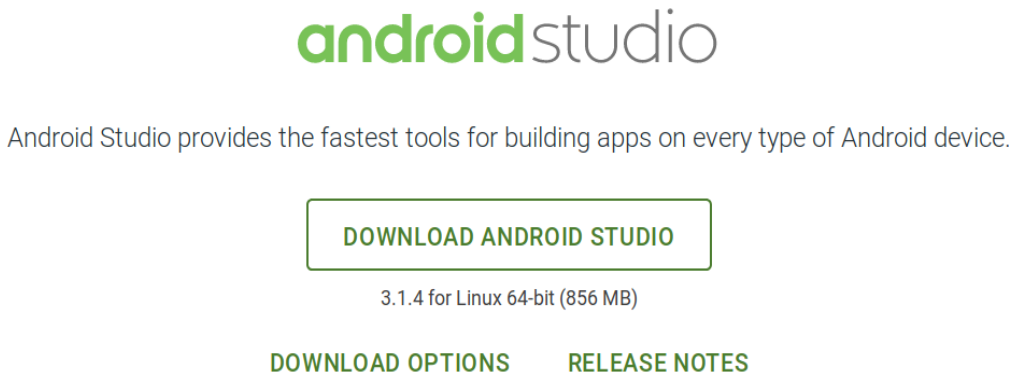
5.2 Android Studio

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

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.

5.2.1 Download and installation

Download Android Studio from the Android Studio website.



Installation for Windows is simple, just launch the .exe you downloaded. On Max OSX drag and drop Android Studio into the Applications folder.

On Linux unpack the downloaded ZIP file into an appropriate location for your applications. To launch Android Studio, navigate to the android-studio/bin/ directory in a terminal and execute studio.sh.

5.2.2 Features

- The following features are provided in the current stable version:
- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps

- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

5.1.2 System Requirements

Table 5.1.2.1: System requirement for Android Studio

Criterion	Description
OS version	Windows 7 or later Mac OS X 10.9.5 or later GNOME or KDE desktop Linux
RAM	8 GB RAM recommended; plus 1 GB for the Android Emulator
Disk space	500 MB disk space for Android Studio, at least 1.5 GB for Android SDK, emulator system images, and caches
Java version	Java Development Kit (JDK) 8, use of bundled OpenJDK (version 2.2 and later) is recommended.[24]
Screen resolution	1280×800 minimum screen resolution.

5.3 Firebase

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.

5.3.1 Services

- Firebase Cloud Messaging

  Formerly known as Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost.

- **Firebase Authentication**

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

- **Real-time Database**

  Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. Developers using the real-time database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the real time Database was released for beta use.

- **Firebase Storage**

  Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

- **Firebase Hosting**

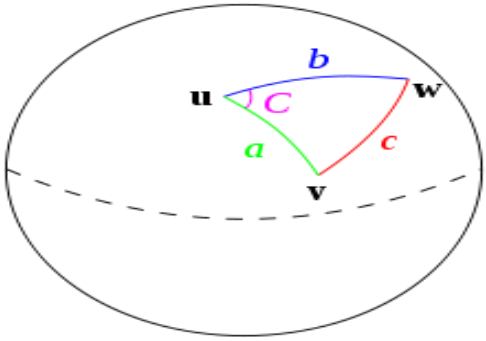
  Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firebase partners with Fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

5.4 Distance Calculation Method

Generally, Haversine is used for computing the great circle distances between two pairs of coordinates on a sphere.

There is no geometric shape that the Earth can fall perfectly under its calculations. Haversine formula perfectly applied to calculate distance on a spherical shape.

The shortest distance between two points on the surface of an earth or a sphere measured along the surface and can be displayed on the Google map.



5.4.1 Formula

dlat = lat2 - lat1

dlon = lon2 - lon1

a = sin (dlat/2)^2 + sin (dlon/2)^2 * cos (lat1) * cos (lat2)

c = 2 * sin (sqrt (a))

d = R * c

5.4.2 Distance calculating steps:

Step 1: Express location coordinates in decimal format

Step 2: Express Longitudes and latitudes

Step 3: Express Location coordinates as radians

Step 4: Determine the initial latitude and initial longitude

Step 5: Calculate the total distance from to the optimal location using the Haversine formula.

5.4.3 Pseudo Code for Haversine Algorithm

```
static const double DEG_TO_RAD = 0.017453292519943295769236907684886;

static const double EARTH_RADIUS_IN_METERS = 6372797.560856;

double latitudeArc  = (from.coordinate.latitude - to.coordinate.latitude) * DEG_TO_RAD;

double longitudeArc = (from.coordinate.longitude - to.coordinate.longitude) * DEG_TO_RAD;

double latitudeH = sin(latitudeArc * 0.5);

latitudeH *= latitudeH;

double lontitudeH = sin(longitudeArc * 0.5);

lontitudeH *= lontitudeH;

double tmp = cos(from.coordinate.latitude*DEG_TO_RAD) * cos(to.coordinate.latitude*DEG_TO_RAD);

return EARTH_RADIUS_IN_METERS * 2.0 * asin(sqrt(latitudeH + tmp*lontitudeH)); }
```

CHAPTER 6

RESULTS AND OUTPUTS

It describes the final outcome of the project. How the project is going to work and the activity will function in the project.

6.1 Admin Page

By using this page admin can add or delete the restaurant or NGOs. Admin can update the co-ordinates of the location of the NGOs or the restaurant and store the data on the cloud.

Fig. 6.1.1 Admin Page

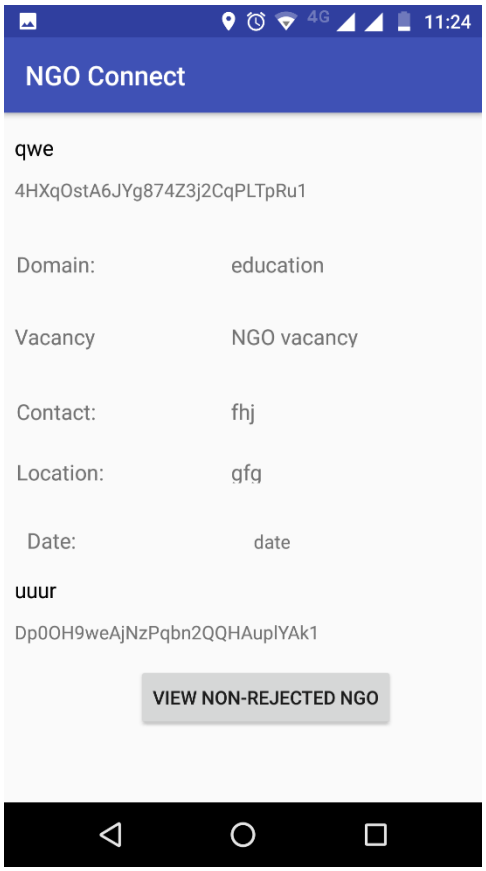


Fig 6.1.2 Addition of NGOs or Restaurants

Form Name: Addition of NGOs and Restaurant

Purpose: To add NGOs and Restaurant in the System

Code:

```
package com.example.bmrd.stesbuddy;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.HashMap;
public class AddNGOActivity extends AppCompatActivity {
    private TextView NGOid;
    private TextView NGOname;
    private TextView NGOvacancy;
    private TextView NGOlocation;
    private TextView NGOlon;
    private TextView NGOlat;
    private TextView NGOdate;
    private TextView NGOdomain;
    private DatabaseReference mUserDatabase;
    private Button addNGObtn;
    private Button finishButton;
```

```
private DatabaseReference mDatabase;
private Task<Void> mRESTDatabase;
private Button addNGOlocbtn;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_ngo);
    NGOid = (TextView) findViewById(R.id.idText);
    NGOname = (TextView) findViewById(R.id.nameText);
    NGOlocation = (TextView) findViewById(R.id.addressText);
    NGOdomain = (TextView) findViewById(R.id.domainText);
    addNGObtn = (Button) findViewById(R.id.addBTN);
    NGOlat = (TextView) findViewById(R.id.latText);
    NGOlon = (TextView) findViewById(R.id.lonText);
    addNGOlocbtn = (Button) findViewById(R.id.updateBTN);
    finishButton = (Button) findViewById(R.id.finishBTN);
    addNGObtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String ngo_name=NGOname.getText().toString();
            String ngo_id=NGOid.getText().toString();
            String ngo_location=NGOlocation.getText().toString();
```

```
        String ngo_domain=NGOdomain.getText().toString();
        System.out.println(ngo_id+">>>>>>>>>>>>>>>>>>>>>>");
       //FirebaseUser current_user = FirebaseAuth.getInstance().getCurrentUser();
      // String uid=current_user.getUid();
         mDatabase=
    FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id);
    HashMap<String,String> VolunteersMap2= new HashMap<>();
        VolunteersMap2.put("name",ngo_name);
        VolunteersMap2.put("location",ngo_location);
       // VolunteersMap2.put("food",rest_food);
        System.out.println(ngo_id+"<<<<<<<<<<<<<");
        VolunteersMap2.put("id",ngo_id);
        VolunteersMap2.put("vacancy","");
        VolunteersMap2.put("domain",ngo_domain);
        VolunteersMap2.put("date","");
        VolunteersMap2.put("lat","");
        VolunteersMap2.put("lon","");
        mDatabase.setValue(VolunteersMap2);
        Intent login_intent=new Intent(AddNGOActivity.this,MapsActivity3.class);
        startActivity(login_intent);
        System.out.println(ngo_id+"><><><><><><><><><><>");
    }
});
```

```
addNGOlocbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           String ngo_id=NGOid.getText().toString();
           mUserDatabase=
        FirebaseDatabase.getInstance().getReference().child("admin").child("Temp_loc1");
           mUserDatabase.addValueEventListener(new ValueEventListener() {
              @Override
              public void onDataChange(DataSnapshot dataSnapshot) {
Toast.makeText(MainMessActivity.this,dataSnapshot.toString(),Toast.LENGTH_LONG).show(
);
                 String lat=dataSnapshot.child("Lat").getValue().toString();
                 String lon=dataSnapshot.child("Lan").getValue().toString();
                 System.out.println(lat+"?????????????");
                 NGOlat.setText(lat);
                 NGOlon.setText(lon);
                 //for image uploading
                 //  Picasso.with(AmbegaoActivity.this).load(image).into(mImage1);
              }
              @Override
              public void onCancelled(DatabaseError databaseError) {
              }
           });
```

```
mRESTDatabase=FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id).child("lat").setValue(NGOlat.getText().toString());

mRESTDatabase=FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id).child("lon").setValue(NGOlon.getText().toString());
            System.out.println(NGOlat.getText().toString()+"?????????????/////////");
        }
    });
    finishButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
        String ngo_id=NGOid.getText().toString();

mRESTDatabase=FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id).child("lat").setValue(NGOlat.getText().toString());

mRESTDatabase=FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id).child("lon").setValue(NGOlon.getText().toString());
            System.out.println(NGOlat.getText().toString()+"?????????????/////////");
        }
    });
  }
}
```

6.2 HOMEPAGE

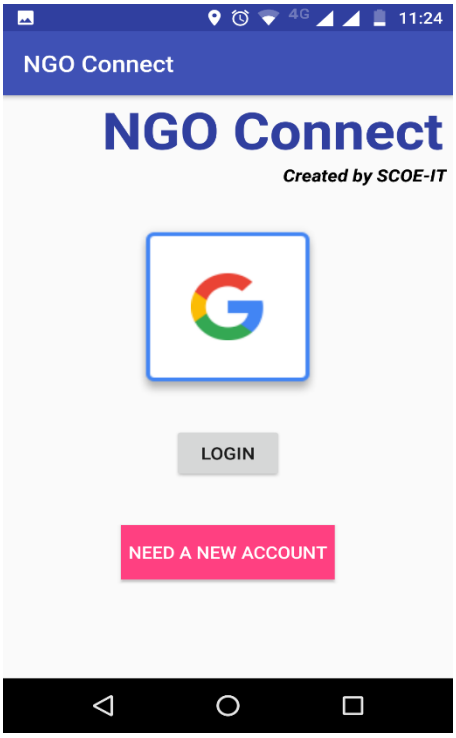


Fig. 6.2.1 Homepage

Form: Homepage

Purpose: For Login and Registration

Code:

```
package com.example.bmrd.stesbuddy;
import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class HomeActivity extends AppCompatActivity {
    private static int SPLASH_TIME_OUT=600;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent homeIntent=new Intent(HomeActivity.this,StartActivity.class);
                startActivity(homeIntent);
                finish();
            }
        },SPLASH_TIME_OUT);
    }
}
```

6.3 USER LOCATION

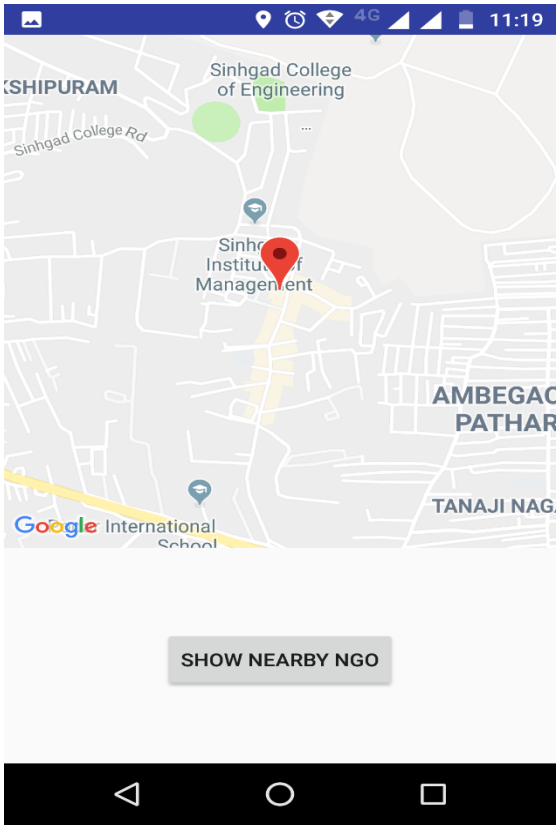


Fig. 6.3.1 User Location

Code Snippets

```
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.HashMap;
```

- **For Nearby Restaurant**

```
CameraUpdate update = CameraUpdateFactory.newLatLngZoom(latLngCurrent, 15);
mMap.animateCamera(update);
```

```
MarkerOptions options = new MarkerOptions();
options.position(latLngCurrent);
options.title("Current Location");
mMap.addMarker(options);
statusOfMap=1;
```

- **Connection to the Cloud**

```
FirebaseUser current_user = FirebaseAuth.getInstance().getCurrentUser();
String uid=current_user.getUid();
mDatabase=
FirebaseDatabase.getInstance().getReference().child("NGO_list").child(ngo_id);
HashMap<String,String> VolunteersMap2= new HashMap<>();
VolunteersMap2.put("name",ngo_name);
VolunteersMap2.put("location",ngo_location);
VolunteersMap2.put("food",rest_food);
System.out.println(ngo_id+"<<<<<<<<<<<<<<");
VolunteersMap2.put("id",ngo_id);
VolunteersMap2.put("vacancy","");
VolunteersMap2.put("domain",ngo_domain);
VolunteersMap2.put("date","");
VolunteersMap2.put("lat","");
VolunteersMap2.put("lon","");
mDatabase.setValue(VolunteersMap2);
```

6.4 Login Page

By using this activity Admin, NGOs, Restaurants and Volunteers can login to the application.

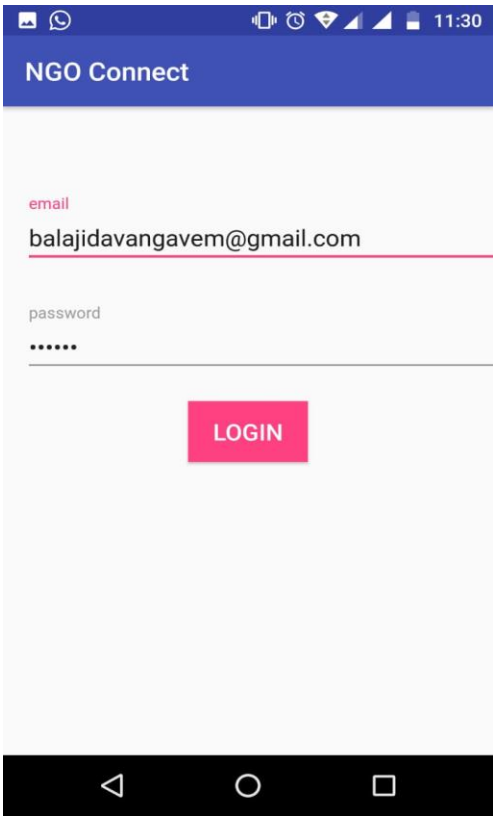


Fig. 6.4.1 Login Page

Code Snippets

```
public void onComplete(@NonNull Task<AuthResult> task) {

if(task.isSuccessful()){

mLoginProgress.dismiss();

Intent mainIntent=new Intent(LoginActivity.this,MainActivity.class);

mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK

Intent.FLAG_ACTIVITY_CLEAR_TASK);
```

```
startActivity(mainIntent);

finish();

}

else

{

 mLoginProgress.hide();

 Toast.makeText(LoginActivity.this,"Oops...error...please try

 again".Toast.LENGTH_LONG).show();

  }

  }
```

6.5 NGOs Entry

It shows the entry of all the NGOs and from here volunteer can select the NGO according to their domain.

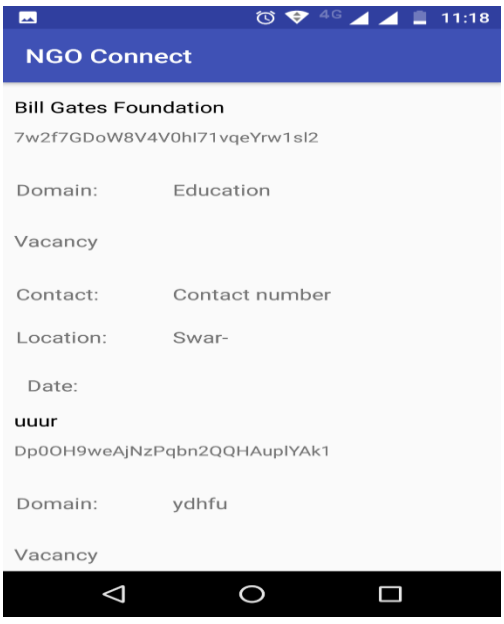


Fig. NGOs List

On the time of registration all the data related to the NGOs will be taken and stored in the cloud.

Registration page will do the work for the entry of the NGOs.

6.6 Restaurant Entry

It shows the entry of all the Restaurants and from here NGO can contact the restaurant according to their need of the food.

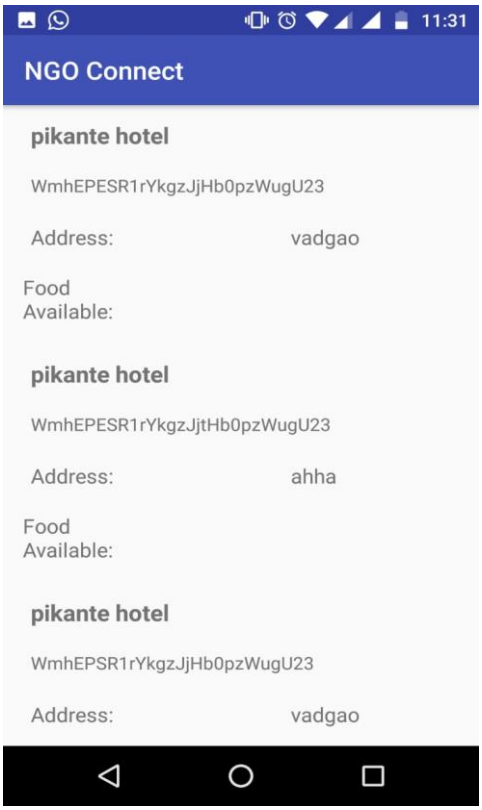


Fig. 6.6.1 Restaurant List

On the time of registration all the data related to the Restaurants will be taken and stored in the cloud.

Registration page will do the work for the entry of the Restaurants.

CHAPTER 7

TESTING

Software Testing:

It is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. Software testing also helps to identify errors, gaps or missing requirements in contrary to actual requirements.

Types of Software Testing:

1. Integration Testing:

Integration testing is a testing in which group of components are combined to produce output. They are subclassified as

A. Black Box Testing:

It is used for validation, in this we ignore internal working mechanism and focus on what is the output?

B. White Box Testing

It is used for verification. In this we focus on internal mechanism i.e. how the output is achieved.

2) System Testing:

In this, software is tested such that it works fine for different operating systems.

3. Performance Testing:

It is designed to test the speed and effectiveness of program.

4. Unit Testing:

It focuses on smallest unit of software design. In this we test an individual unit or group interrelated units. It is often done by programmer by using sample input and observing its corresponding output.

Table 7.1: Test Cases

Test Case ID	Test Case Description	Input Values	Expected Result	Actual Result	Status
1	NGO Sign up	User name, domain, contact number, address, email id, password	Creation of an NGO account and request to admin for acceptance	Creation of an NGO account and request to admin for acceptance	Pass
2	Volunteer Sign up	User name, contact number, age, gender, qualification, email id, password	Creation of a Volunteer account	Creation of a Volunteer account	Pass
3	Restaurant Sign up	User name, FSSAI number, contact number, address, email id, password	Creation of restaurant account and request to admin for acceptance	Creation of restaurant account and request to admin for acceptance	Pass
4	NGO Login	UserID + password	Logged In	Logged In	Pass
5	Volunteer Login	UserID + password	Logged In	Logged In	Pass
6	Restaurant Login	UserID + password	Logged In	Logged In	Pass
7	Administrator Login	UserID + password	Logged In	Logged In	Pass
Test Case ID	Test Case Description	Input Values	Expected Result	Actual Result	Status
8	Getting User's Current location	-	Latitude & longitude of user	Latitude & longitude of user	Pass
9	Update food availability status	Amount of available food in kg	Updated data	Updated data	Pass
10	Creation of volunteering event	1)Date of event 2) Number of vacancies	Create an event	Create an event	Pass

11	View list of applied volunteers	-	List of applied volunteers	List of applied volunteers	Pass
12	Selection of volunteers	-	Selection of volunteers	Selection of volunteers	Pass
13	Acceptance or rejection of a restaurant	-	Accept or reject a restaurant	Accept or reject a restaurant	Pass
14	Acceptance or rejection of an NGO	-	Accept or reject an NGO	Accept or reject an NGO	Pass

CONCLUSION

The system which we have proposed an android application which connecting people, restaurants, NGOs for the purpose to improve waste food management services. This is the first application we are representing, no previous work is done on it.

This application is based on android, cloud, location based service. For location based service we are using Haversine formula. Android app is connected to cloud for data storage purpose. It concludes that using cloud, Haversine formula we are arranging volunteers according to shortest distance allocation to serve for the NGOs.

REFERENCES

[1] E. Winarno, W. Hadikurniawati and R. N. Rosso, "Location based service for presence system using haversine method," *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, Salatiga, 2017, pp. 1-4.

[2] S. K. Tiwari, G. K. Varshney, M. A. Qadeer and M. S. Umar, "JustQuick: Intelligent LBS using Android with GPS and geo-tagging applications," *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Bangalore, 2016, pp. 820-824.

[3] W. Li, C. Yen, Y. Lin, S. Tung and S. Huang, "JustIoT Internet of Things based on the Firebase real-time database," *2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*, Hsinchu, 2018, pp. 43-47.

[4] S. Kumar, M. A. Qadeer and A. Gupta, "Location based services using android (LBSOID)," *2009 IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA)*, Bangalore, 2009, pp. 1-5.

[5] P. Battin and S. D. Markande, "Location based reminder Android application using Google Maps API," *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, Pune, 2016, pp. 649-652.

[6] https://rosettacode.org/wiki/Haversine_formula#Java

[7] https://firebase.google.com/

[8] https://firebase.google.com/docs/android/setup

[9] https://developer.android.com/studio/write/firebase

[10] https://blog.back4app.com/2016/06/15/firebase-parse/

[11] Dawn Griffiths, David Griffiths: Head First Android Development O'REILLY