SOFTWARE REQUIREMENT SPECIFICATION
for
Location Tracker Application: “BEPPY”

(**In accordance with IEEE 830-1998**) v1.0

Developer:

Ashiqul Hasan Shaki l(2018831072)
Md Fahim Mia (2018831064)

Course Code: SWE226.
Date : 20-08-2021

Content

Project Title: “BEPPY”

Category: Mobile Application.

1. INTRODUCTION

This document is a Software Requirement Specification for the Android Mobile Application named “BEPPY”. This document is prepared by the following IEEE conventions for software requirement specification. This document includes all the functions and specifications with their explanations to solve related problems .

1.1 PROBLEM DEFINITION

In the last two decade, Internet and mobile phones have increased rapidly. Nowadays, almost all people has a mobile phone and different kind of mobile applications. This leads many simplicities on people’s life in terms of communication. There are many beneficial mobile applications and web applications for the humankind which ease their lives. However, there are also some concepts that could not be resolved yet. There are many social platform application on mobile phone and most of them do not socialize people in real life. There are not enough social media platforms gather people in a place and do not enable users meet new people in terms of their interests. Furthermore, there are not enough location based messaging which provide users to see the location of user and message with that user simultaneously. Let us explain those needs in a scenario. First need in a mobile application is location based messaging. For example, two users message with each other and try to find each other in a certain area. Using whatsapp, they can send each other their locations to find themselves. However, in order to open those locations, whatsapp use Google Maps application and this is the hard way for finding each other in real life. The easiest solution for finding each other in real life while messaging is to create a system which provides messaging and seeing locations at same time together. Second need is gathering people in a certain area in terms of interests. For example, let us think a person who wants study a lesson with a person who perfectly know that lesson. However, he may study that lesson with a person who he does not know. There are not enough applications for that purpose. In our project, considering those needs and problems, we are trying to develop a mobile Android application who will meet those needs.

1.2 PURPOSE

Purpose of this software requirement specification document is providing complete description of the features which will be implemented in this location-based social platform. Furthermore, this document includes user scenarios, UML diagrams, working principles of the system, internal and external interfaces. This document is going to serve as a guideline for both the development team and users of the application.

1.3 SCOPE

As it is described in the Problem Definition section, there is no social platform which has location based messaging and gather different people with their interests and provide an opportunity for users to meet new people with same interests.

Our project named “BEPPY” has a purpose for solving those problems as mentioned above and described in the Problem Definition section. Our project has mainly two major features. First major feature of our Android Mobile Application is location based messaging system. In our location based messaging system, user can see location of friend who is messaging with him simultaneously. In other words, a user can see location of the friend and message with that friend on the same interface .This is the main aspect of location based messaging system. Second major feature is creating events and sharing post with the locations, photos and contents. Event Types are private, public, privileged events and each event has some features inside. Firstly ,a private event or private post of a user can only be seen by the friends of the user within a certain radius. User can share post or create events such as home party and in this way , user can easily contact with his friends and keep in touch with his friends. This event or post sharing is like Facebook, Twitter. Secondly, public event or public post can be seen by everybody within a certain radius. This event or sharing post is like private event

creating but this event type makes people a social people in real life because this event provide opportunities for people to meet new people. Lastly, privileged events created by employees or boss of hangout places and can be seen by everybody within a certain radius. This event type creates a new advertisement area for hangout places. For example, a hangout place like a pub organize a live music program for a night and create a privileged event on BEPPY mobile android application and reach as much as possible people in that area. These features are the main part of our project BEPPY. We will try to explain our project in the following section in detail.

1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

SRS	Software Requirement Specification
GCM	Google Cloud Messaging
GUI	Graphical User Interface
OS	Operating Systems
GPS	Global Positioning System
User	Anybody who uses the application BEPPY
IEEE	Institute of Electrical and Electronic Engineers
External Usage	Android Disk Usage
API	Application Programming Interface
SDK	Software Development Kit

1.5 REFERENCES

IEEE.IEEE Std. 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society,1998 .

1.6 OVERVIEW

This document includes six major topics for the following chapters of software requirements specification. Firstly, overall description of the product perspective, functionality of the product, dependencies and constraints are explained. In the following chapter, specific requirements like interface, functional and non-functional requirements are mentioned. In the following chapters, behavioral and data model are discussed. Finally, software requirements specification document end with conclusion.

2. OVERALL DESCRIPTION

This chapter of the SRS is about general factors that influence the product. Moreover, this chapter contains interfaces of product. Also, this part provides a background for requirements which will be defined in Chapter 3.

2.1 PRODUCT PERSPECTIVE

In this project, we will use different cloud system such as Google Cloud Messaging and different Google APIS such as Google Maps API and Google Places API. We will use Google Maps API and Google Places API for events and location based part of the project and Google Cloud Messaging for location based messaging system. For instance, we will use Google Cloud Messaging system for the connection between server, database and Android device. Application sends a message request to the server and server connecting with the Cloud and server sends the message to the target user.

2.1.1 SYSTEM INTERFACES

This Android Mobile application will be implemented in programming language Java and development environment is Android Studio developed by Intelij. In this section, we will mention APIs, SDKs, databases that we will use.

SDKs : We will implement our mobile application with only using Android SDK which is SDK version 21.The reason behind using this version is because 22 or highest SDK version is not stable yet.

APIs: We will use Google Places API and Google Maps API for implementing our mobile application because our project is based on locations. The services and functions provided by Google have very essential role in our application.

Cloud System: We will only use Google Cloud Messaging cloud system provided by Google in our application for realizing messaging system. This cloud will be used for the connection between Android devices and Firebase. Let us explain this connection with a scenario. For example, in messaging system, one Android device sends a message to the target device from the Firebase database and sends registration Id of the target device to the Google Cloud Messaging. After then, Google Cloud Messaging sends message to the target device using sender id and push notifications to the target device .This is how messaging system works in briefly.

Databases: We will use mainly Firebase Database for our application.

2.1.2 USER INTERFACES

Since we will have a mobile application and we desire to be used by as much as possible people, we will aim that user interfaces of this application will be comprehensible and easy to use. In this section ,we will try to give best explanations about our interfaces.

2.1.2.1 Welcome Page Interface

This is the first interface that user will see when user uses the application for the first time. In this interface, there will be a logo of Splash Screen and Show Alert Dialog to allow location permission on/off. If you will allow location permission on .then you will go next step. Otherwise you will not go in this app.

2.1.2.2 Register Interface

In this interface, BEPPY will enable users to register into the BEPPY system. After entering necessary information, user can register to the application. Furthermore, in this interface, user can also upload their profile photo to the Firebase Storage.

2.1.2.3 Main Interface

This interface is the main interface after registration and welcome page/home page. This interface is mainly about created all friends list. In this interface,you will share your location to your friends. There are some navigation button such as Emergency helper, My location, Notification button and right to top 3 dots button. There are some options such that Send Message, Find Friends, My friends, User Profile And Log Out Option.

2.1.2.4 User Profile Interface

In this interface, a user can see his information, profile photo, lastly created events etc. Also,in this interface he may change his information or profile photo.

2.1.2.5 My Friends Profile Interface

In this interface, a user can see a profile of friend .A user can also send message to a person in this interface.

2.1.2.6 All Users' Profile Interface

In this interface, a user can see a profile of friend or profile of all users A user can also send a friend request to a person in this interface.

2.1.2.7 Send Message Interface

In this interface, a user can send message to his friend. A user can change her message Interface background.

2.1.2.8 Emergency Helper Interface

In this interface, A user can show who has given permission to view the location.

2.1.2.9 My Location Interface

In this interface, user can see his current location as well as search any places and see nearby hospital, school and Restaurant.

2.1.3.1 Notification Interface

In this interface, notifications will be shown. These notifications are something like friends request, created event by a friend or created event nearby a user.

2.1.3.2All Event Interface

This interface will split up three sub-interfaces named "private event interface"," public event interface" and "privileged event interface". These three sub interfaces are mentioned in the following sections.

2.1.3.3 Private Event Interface

We will show the private events nearby the user in this interface. A private event is the event created by friends of the user. When a request comes from friend of user, user can accept the invitation for joining event.

2.1.3.4 Public Event Interface

We will show the public events nearby the user in this interface. A public event is the event created by anybody even if user does not know the creator of the event. If the user want to participate the event ,he can send the request to the creator of the creator of the particular event.

2.1.3.5 Privileged Event Interfac

We will show the privileged events nearby the user in this interface.A privileged event is the event created by employees or boss of hangout places such as pubs,restaurants or

a fitness club.User can join privileged events without sending request.

2.1.3.6 Particular Event Interface

This interface will be used for showing the informations of an event such as description of the event, location of the event or the creator of the event.If it is private or privileged event ,location of the event can be seen by the user.On the other hand,if this is a public event ,location of the event can not be seen by the user if creator of the event did not accept the joining request coming from the user.After the creator accepts the joining request,that user can see the location of the event.This property inside BEPPY provides security of users

2.1.3.7 Creating Event Interface

This interface provides opportunity to the user to create new event.A user can either use his current location or places already on the map.If the user want to create a private event , he can invite his friends to the event.Furthermore, a user should describe the enough informations of the event.If the user want to choose a location already on the Google map,this interface will direct to the Google Places Interface.This interface will be mentioned in the following section.

2.1.3.8 Google Places Interface

This interface enables a user to choose a location for creating event different than present location of user.User can travel around the map and choose location or he/she can search a location for creating event using search button.

2.1.3.9Seeing Nearby Friend Interface

This interface will be created by using Google Map Api and in this interface, user can see the nearby friends on the map with their locations.This interface has many features.For example,user can see the locations of the friends within a certain radius determined by the user.

2.1.3.10 Friends Location Interface

This interface will be used for showing locations of two friends on the map and scales the map in terms of markers of the users. Photos of the friends are shown instead of locations. The locations are updated in every 5 seconds(may be changed later) and also on the Map.

2.1.3) HARDWARE INTERFACES

This mobile application, BEPPY,will work on Android Device. Since the application must run connected to internet and need a location data, Android devices must have a GPS unit and internet connection in order to run this application.

2.1.4)SOFTWARE INTERFACES

Since our project is an Android application,in order to be runned by an Android Device, an Android device must have minimum SDK 22.0 or higher SDKs. It means that %91 of all Android devices can run this application.

2.1.5)MEMORY CONSTRAINT

We are expected that BEPPY will occupy at most 20 MB in the external storage and will have 30 MB Ram usage. We will develop BEPPY with trying to occupy less RAM usage. External storage usage is not in our hands. It is determined by Android SDK.

2.1.6) SITE ADAPTATION REQUIREMENTS

After developing BEPPY, the application will be available on Google Play Store. Android device must have SDK 22.0 or higher SDKs in order to run this application.

2.2 PRODUCT FUNCTIONS

2.2.1 PRIVILEGED USER USE CASE

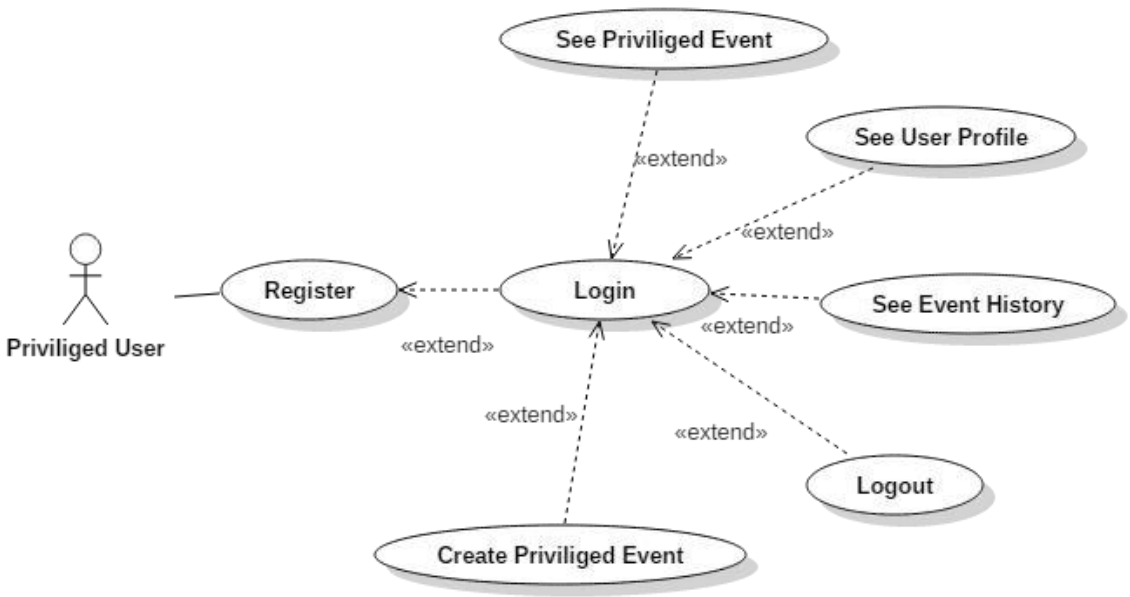


Fig - 2.2.1

Use Case	Description	Figure
Registration	Privileged user should identify them to be privileged and register the system.	Fig-2.2.3
Login	Privileged user can log in to the system in order to use the system.	Fig-2.2.7
Logout	Privileged user can logout from the system.	Fig-2.2.6
Delete account	Privileged user can delete their account.	Fig-2.2.8
See Privileged Events	Privileged user can see all privileged events	Fig-2.2.5
See Privileged User Profile	Privileged user can see their and other users' profile who participated their event(s).	Fig-2.2.10
See Participants of Events	Privileged user can see the users who participate the events.	Fig-2.2.9
See Event History	Privileged user can see event history of them.	Fig-2.2.4
Create Privileged Event	Privileged user can create privileged event.	Fig-2.2.2

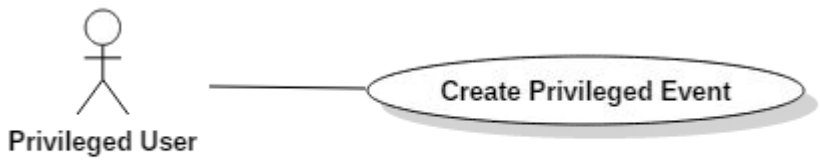


Fig – 2.2.2

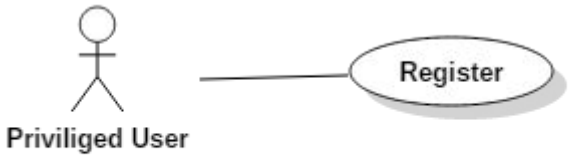


Fig - 2.2.3

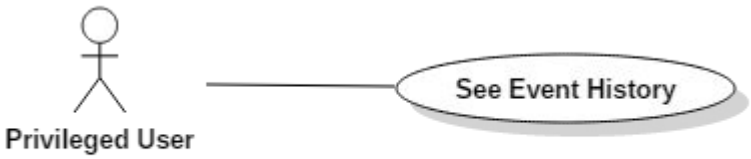


Fig - 2.2.4

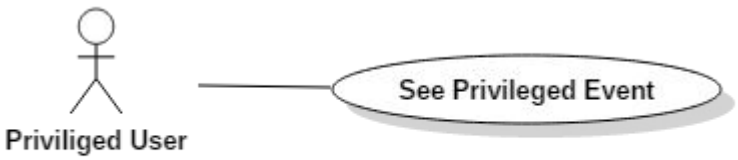


Fig - 2.2.5

Fig-2.2.6

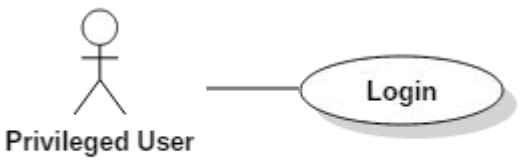


Fig - 2.2.7

Fig - 2.2.8

Fig - 2.2.9

Fig - 2.2.10

2.2.2 USER USE CASE

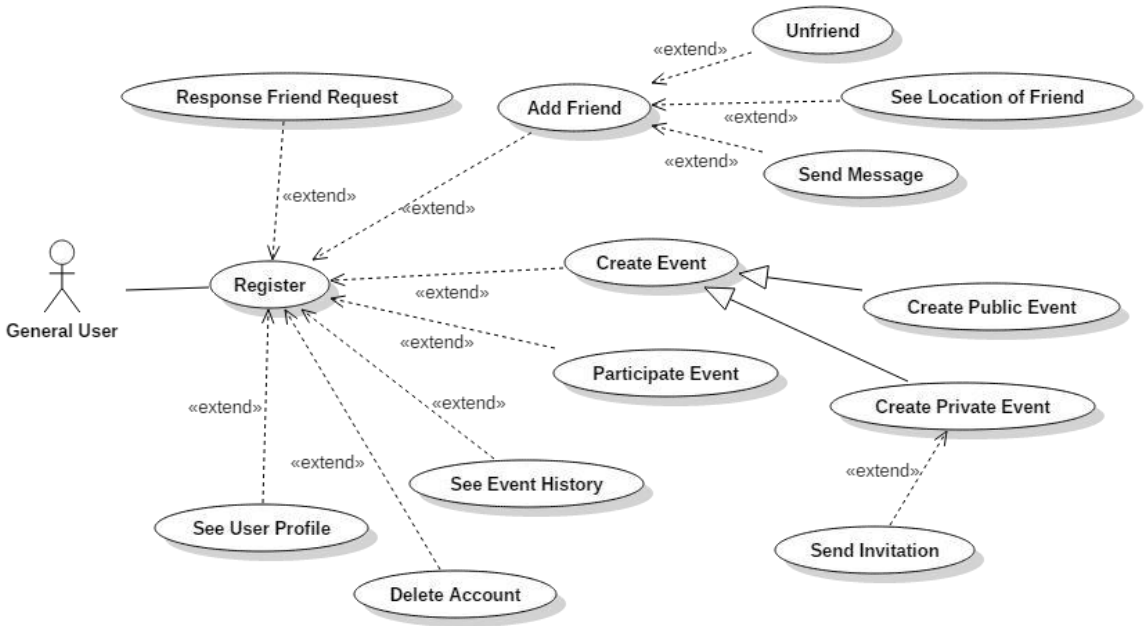


Fig - 2.2.11

Use Case	Description	Figure
Register	User can register to BEPPY without verification.	Fig-2.2.17

Delete account	User can delete their account.	Fig-2.2.25
Respond Friend Request	User can respond friend requests	Fig-2.2.18
Add Friend	User can add people as friend	Fig-2.2.12
Create Event	User can be able to create public/private event	Fig-2.2.13
Participate Event	User can be able to participate all kinds of events by sending requests. All request will be responded by the creator of event.	Fig-2.2.14
See Event History	User can see the event history(participated/created)	Fig-2.2.19
See User Profile	User can see user profiles	Fig-2.2.21
Unfriend	User can delete people from their friend list	Fig-2.2.24
See Location of Friends	User can see the location of their friends without any permission	Fig-2.2.20
Send Message	User can send message to their friends	Fig-2.2.23
Create Public Event	User can create public event both based on a privileged event or not	Fig-2.2.16
Create Private Event	User can create private event both based on a privileged event or not	Fig-2.2.15
See Events	User shall be able to see all events that they have permission on map.	Fig-2.2.26
Send Invitation	People can send invitation to their friends while they are creating private event	Fig-2.2.22

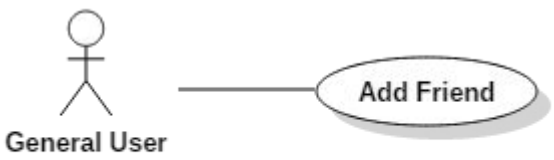


Fig - 2.2.12

Fig - 2.2.13

Fig - 2.2.14

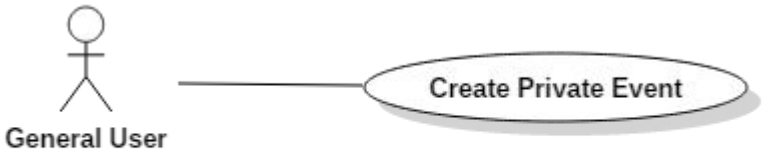


Fig - 2.2.15

Fig - 2.2.16

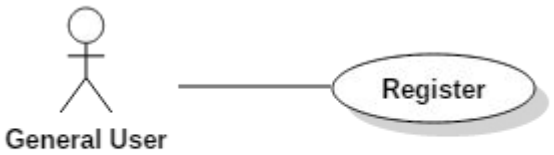


Fig - 2.2.17

Fig - 2.2.18

Fig - 2.2.19

Fig - 2.2.20

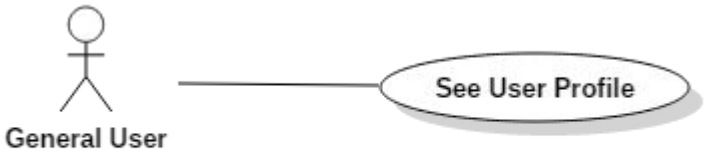


Fig - 2.2.21

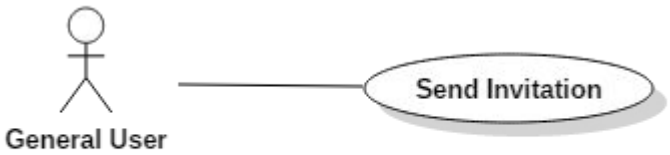


Fig - 2.2.22

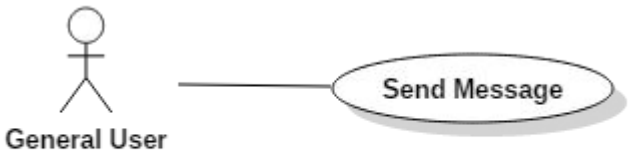


Fig - 2.2.23

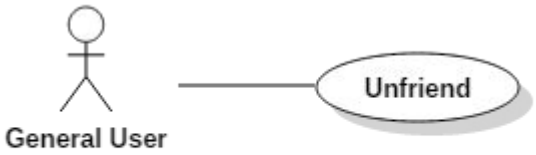


Fig - 2.2.24

Fig - 2.2.25

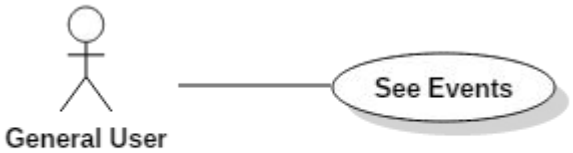


Fig - 2.2.26

2.3 CONSTRAINTS

- BEPPY shall operate on Android 5.0 or higher operating systems.

There should be at least 20 MB free external storage space.CPU speed and Ram capacity is not big concern.

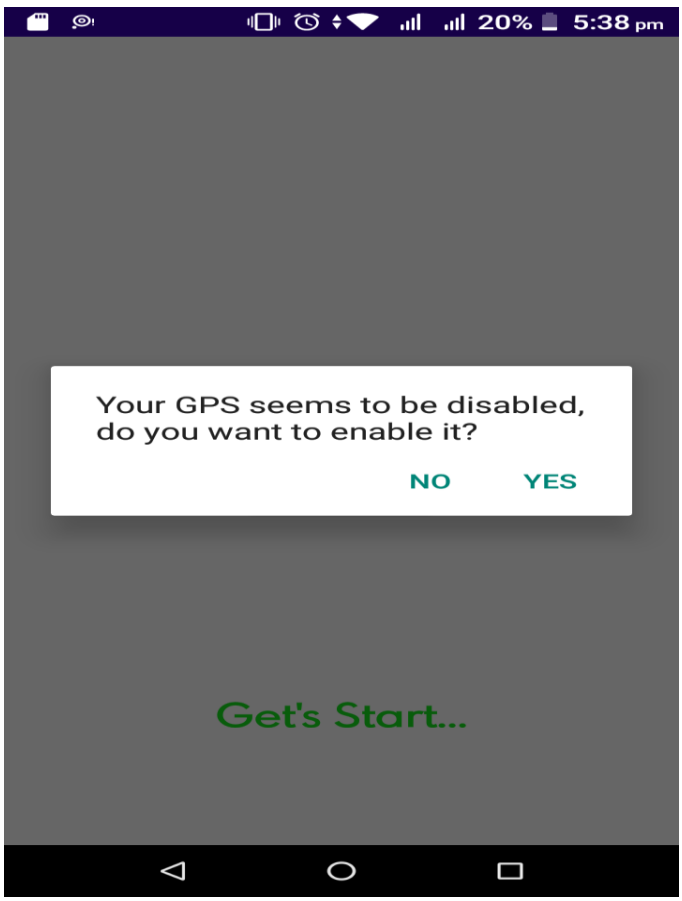
- Java shall be the implementation language with Android and Google Apis.

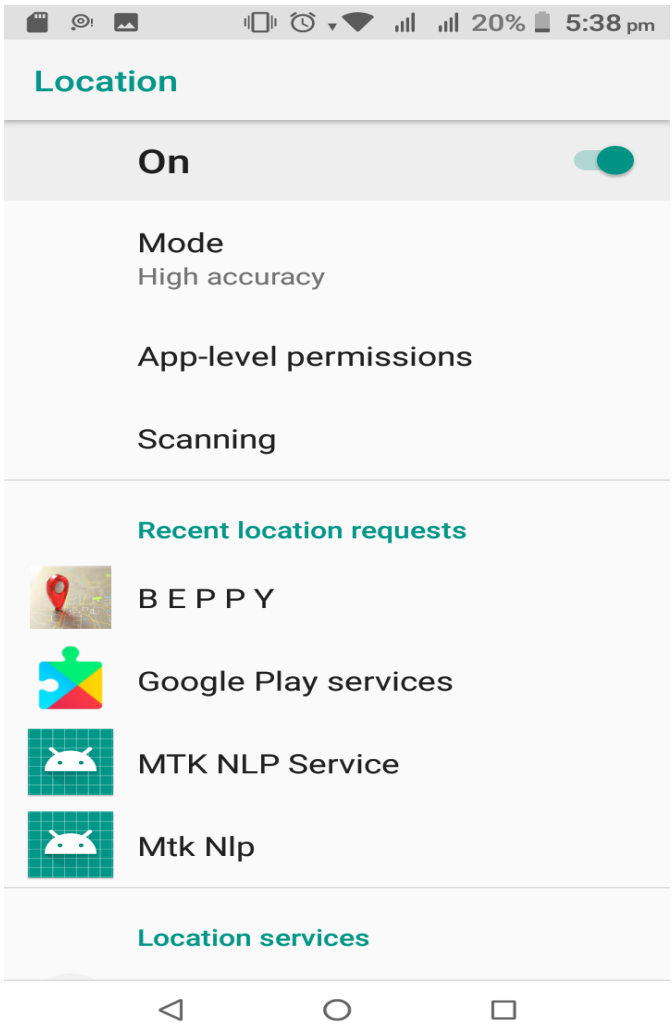
2.4 ASSUMPTIONS AND DEPENDENCIES

- We assume that finding exact locations of the users and events ,GPS and Internet services will give the exact longitudes and latitudes.
- We assume that Google Places API provides contemporary places.
- We assume that Google Maps API provides contemporary roads ,streets etc.
- Since we will develop our application with Agile methods, user interfaces and functionalities may change in the future .It depends on the cycle of the project and the team Code Whisperers.
- We assume that BEPPY application will be used by people frequently in the future.We will expect that because this application needs people and events that people create. Otherwise, it will not be useful application for people.

3.SPECIFIC REQUIREMENTS

3.1) INTERFACE REQUIREMENTS




20% 5:38 pm
Location
On
Mode
High accuracy
App-level permissions
Scanning
Recent location requests
B E P P Y
Google Play services
MTK NLP Service
Mtk Nlp
Location services


66% 9:00 am
এখানে চাপুন
Get's Start...

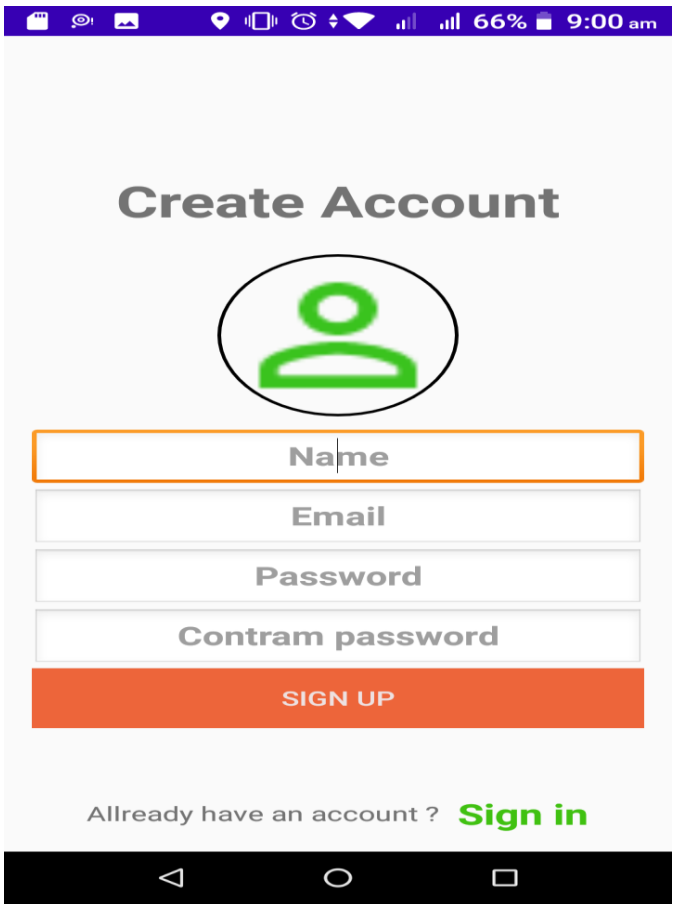

66% 9:00 am
Create Account
Name
Email
Password
Contram password
SIGN UP
Allready have an account ? Sign in

Fig - 3.1.1 Registration Interfaces for Application

User Registration

After click এখানে চাপুন Button, User has to fill out the required attributes like name, password, e-mail and confirm password in order to register the application.

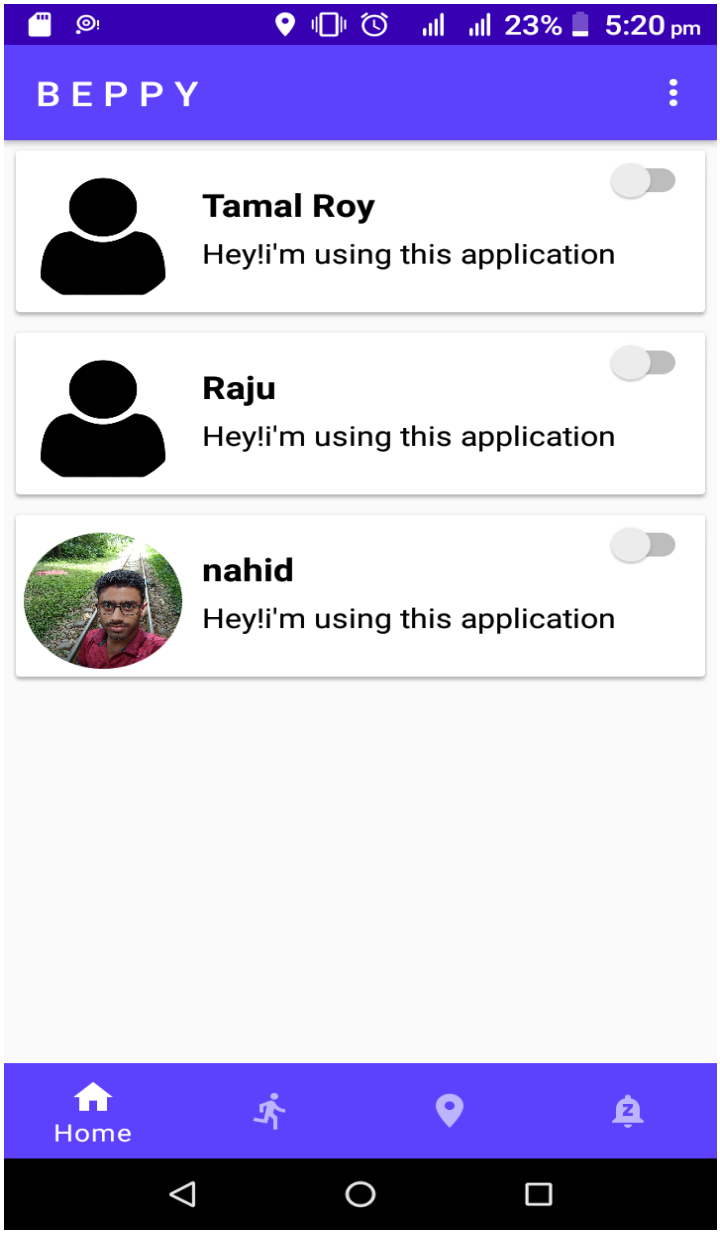
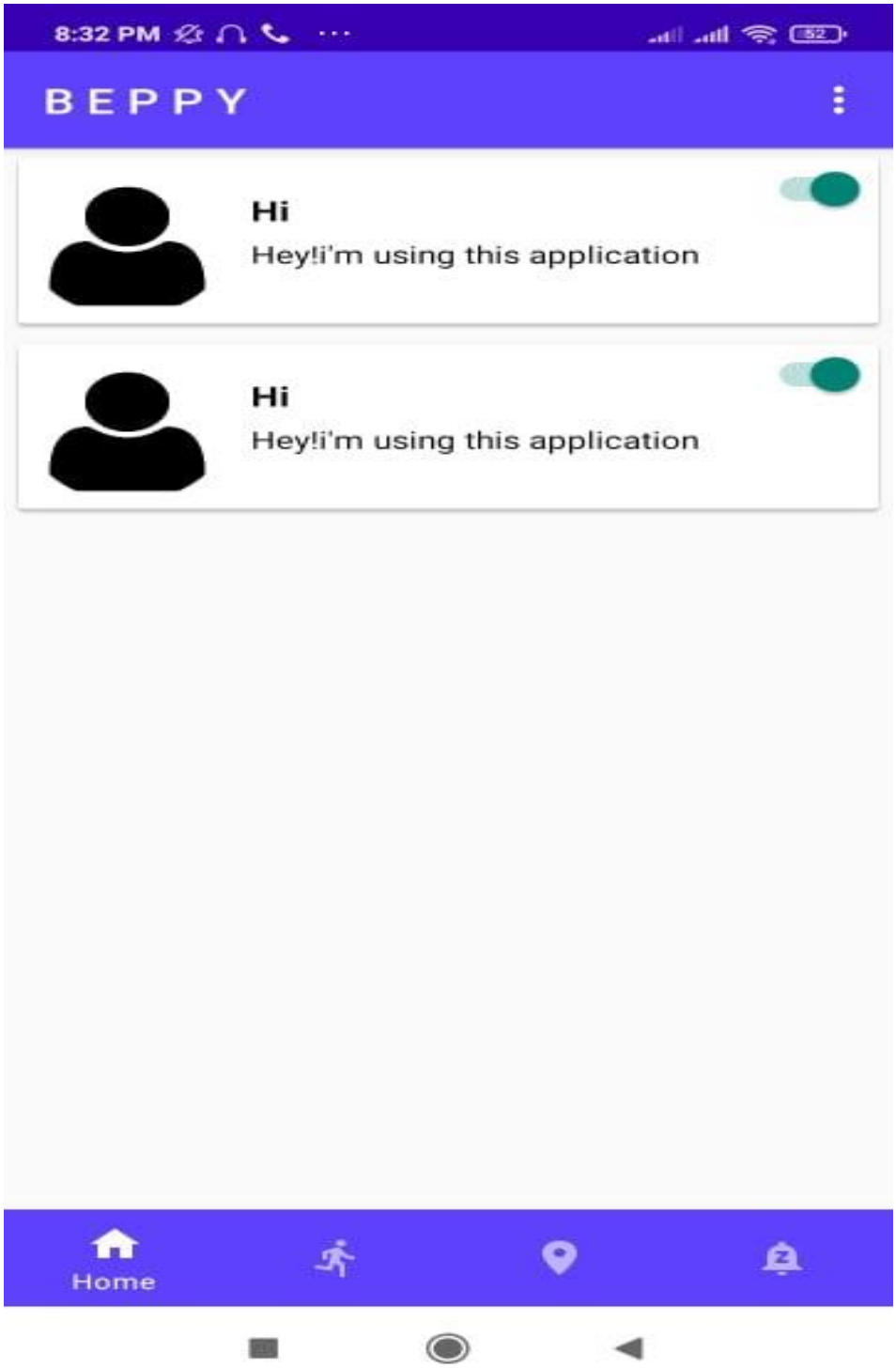


Fig - Main Interfaces for Application


8:32 PM
BEPPY
Hi
Hey!i'm using this application
Hi
Hey!i'm using this application
Home

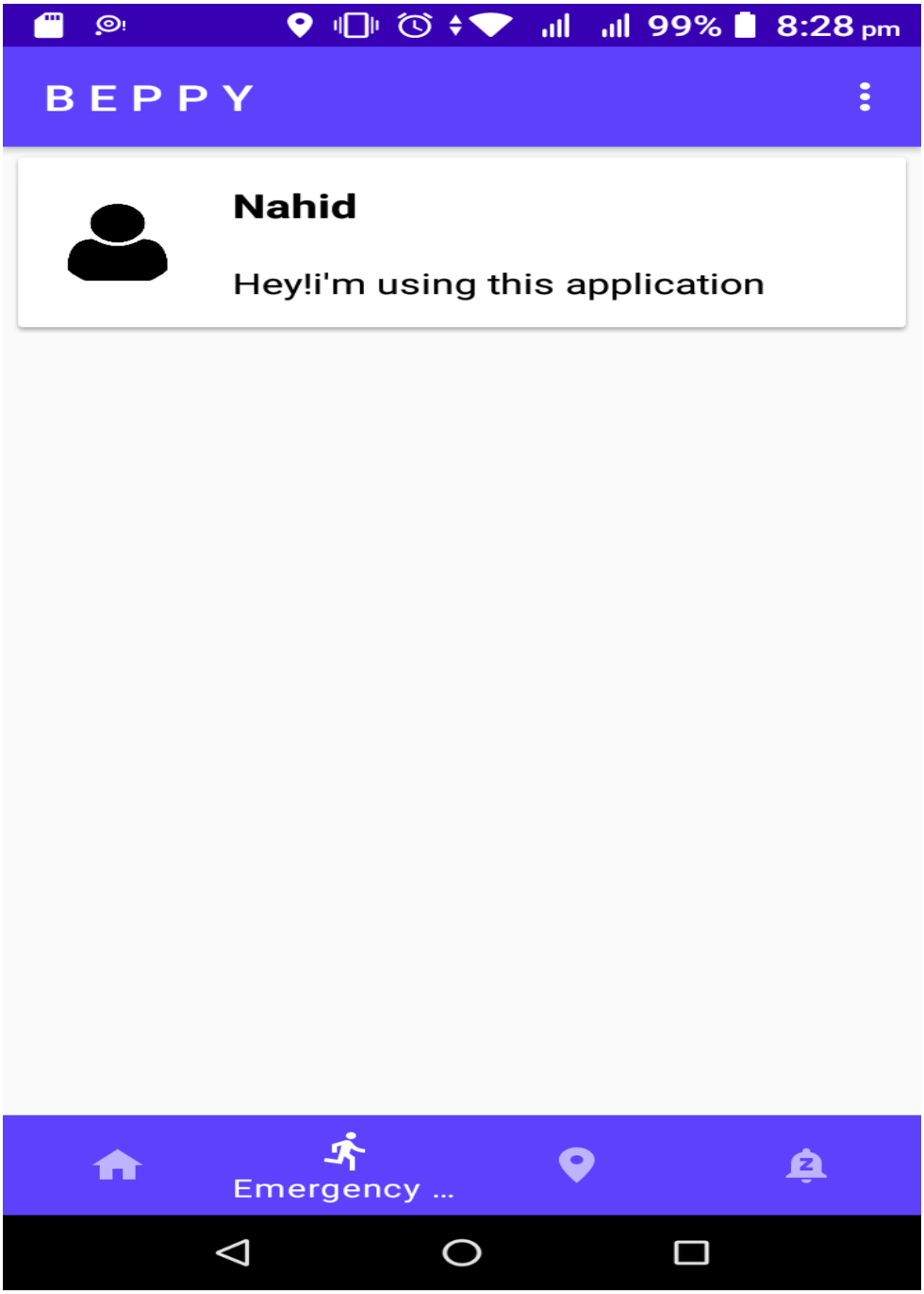


Fig : Emergency Helper Interface

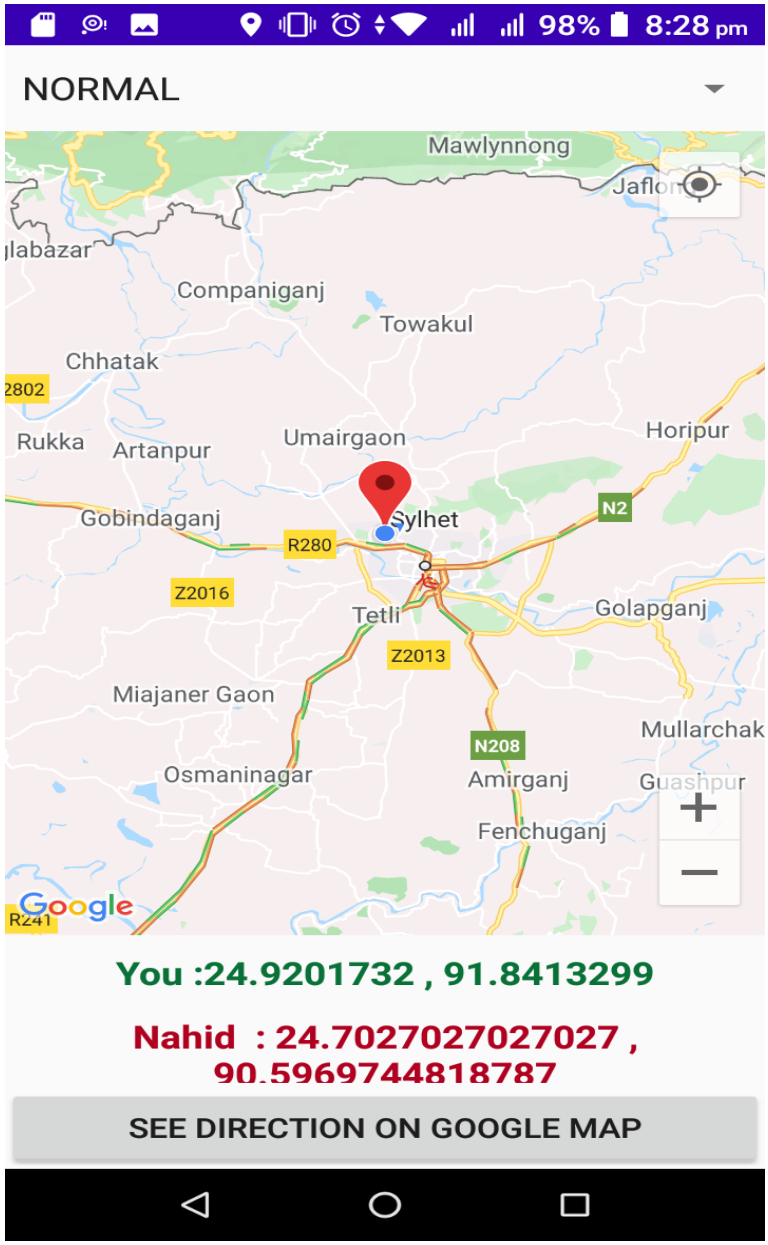


Fig :- Friends Location Interface

After Click see direction Button,

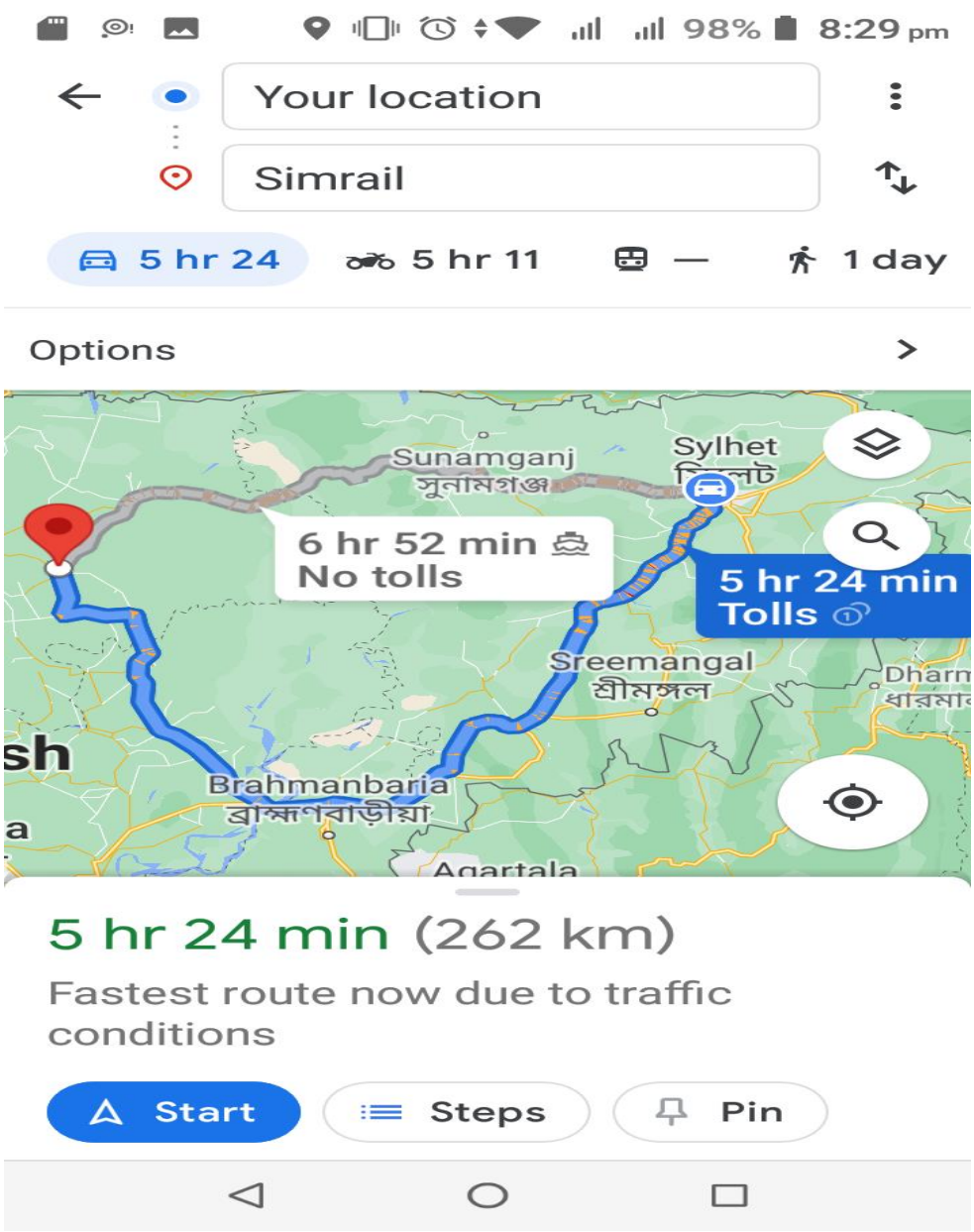


Fig :- Friends Location Interface in google map

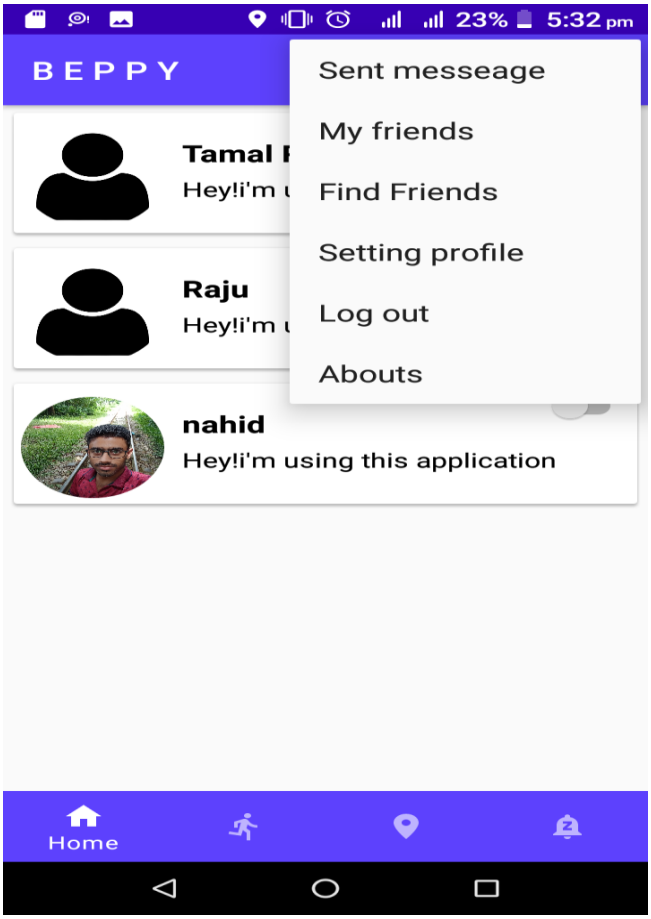


Fig – 3 dots menu Interface

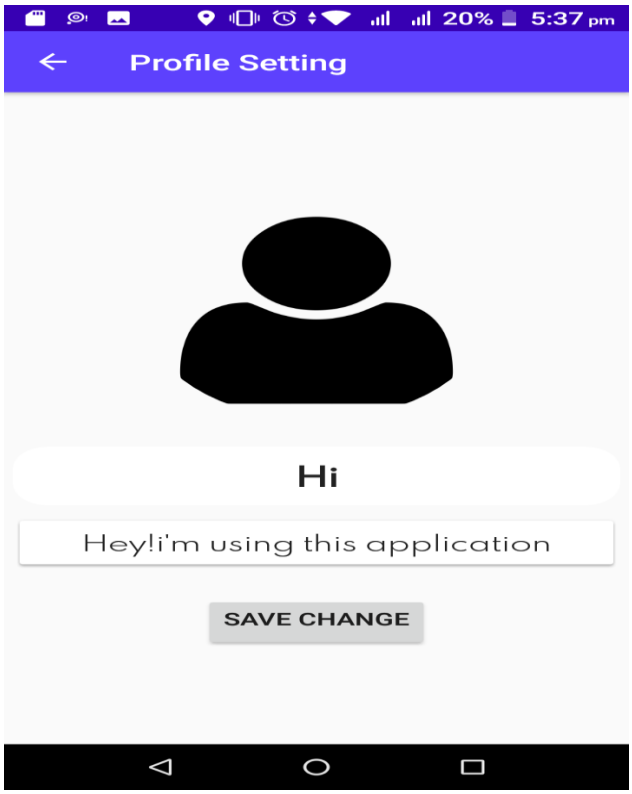


Fig – User Profile Interface .

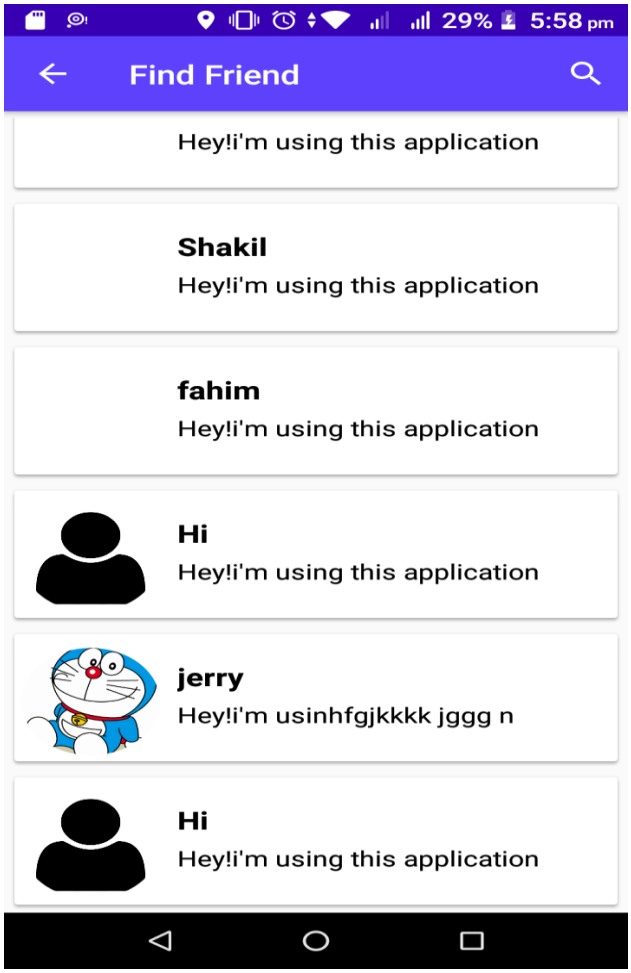


Fig - Find friends Interface

From Here, a User finds and adds his friend by click any user item.

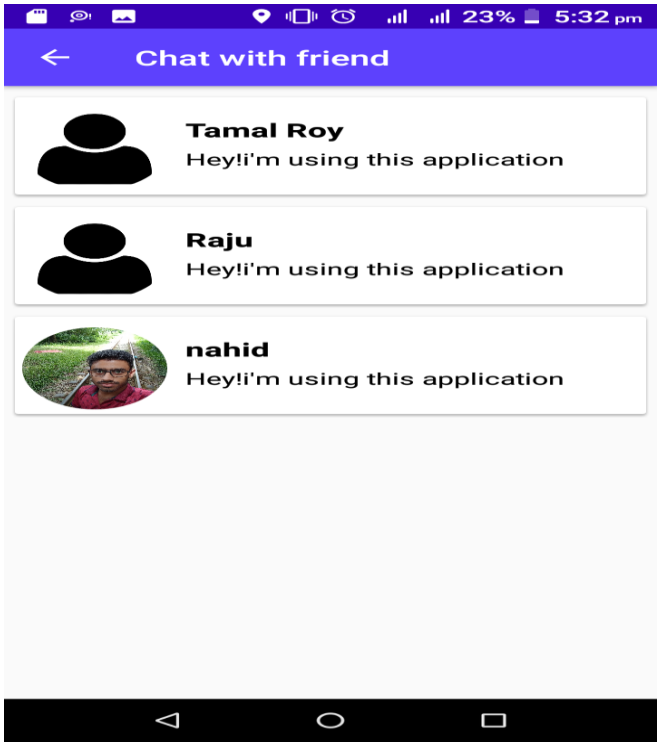


Fig :- Friends interface for this Application

Messaging

When user click the friend on friends interface, he/she are directed to this interface. In this interface, user can send message to his/her friend, can receive message from his/her friend.

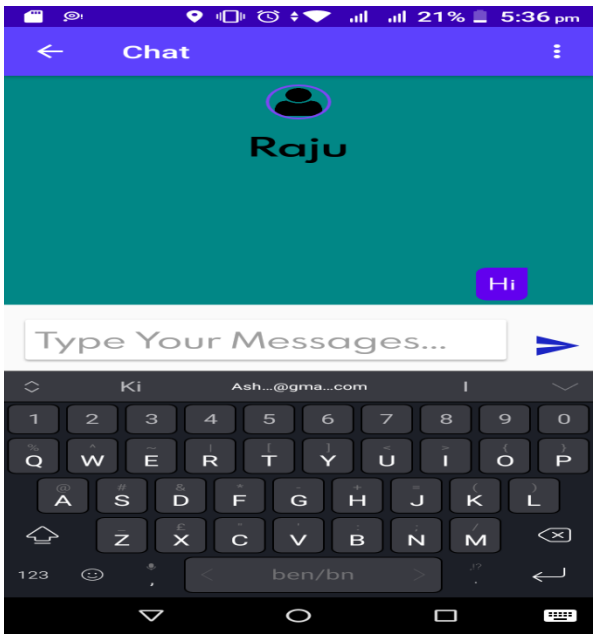


Fig:-Message Interface for this Application

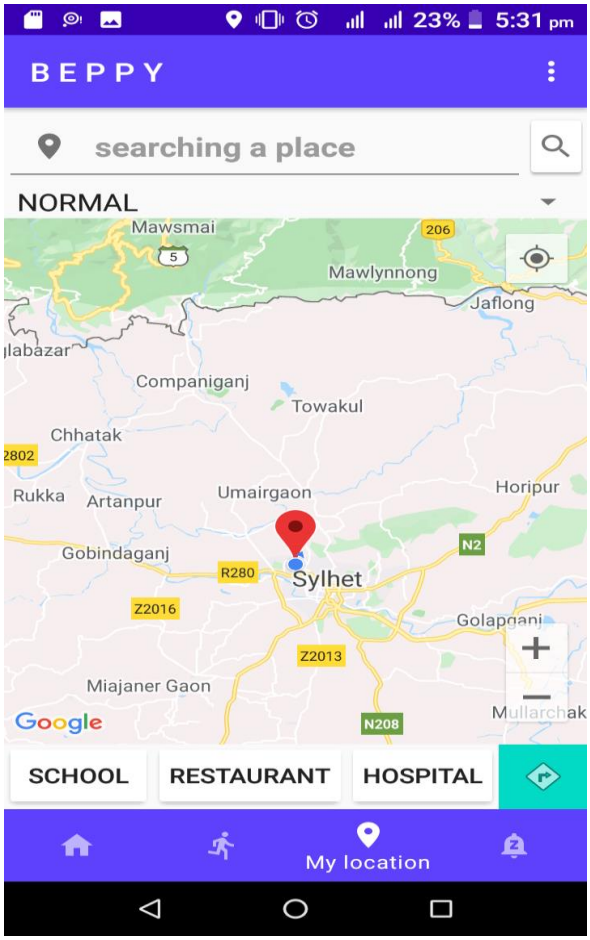


Fig - My Location Interface For this Application

User can see his current location in this interface. And see near school, resturent and Hospital as well as sortest path from user by click those button .

Fig : All notification Interface

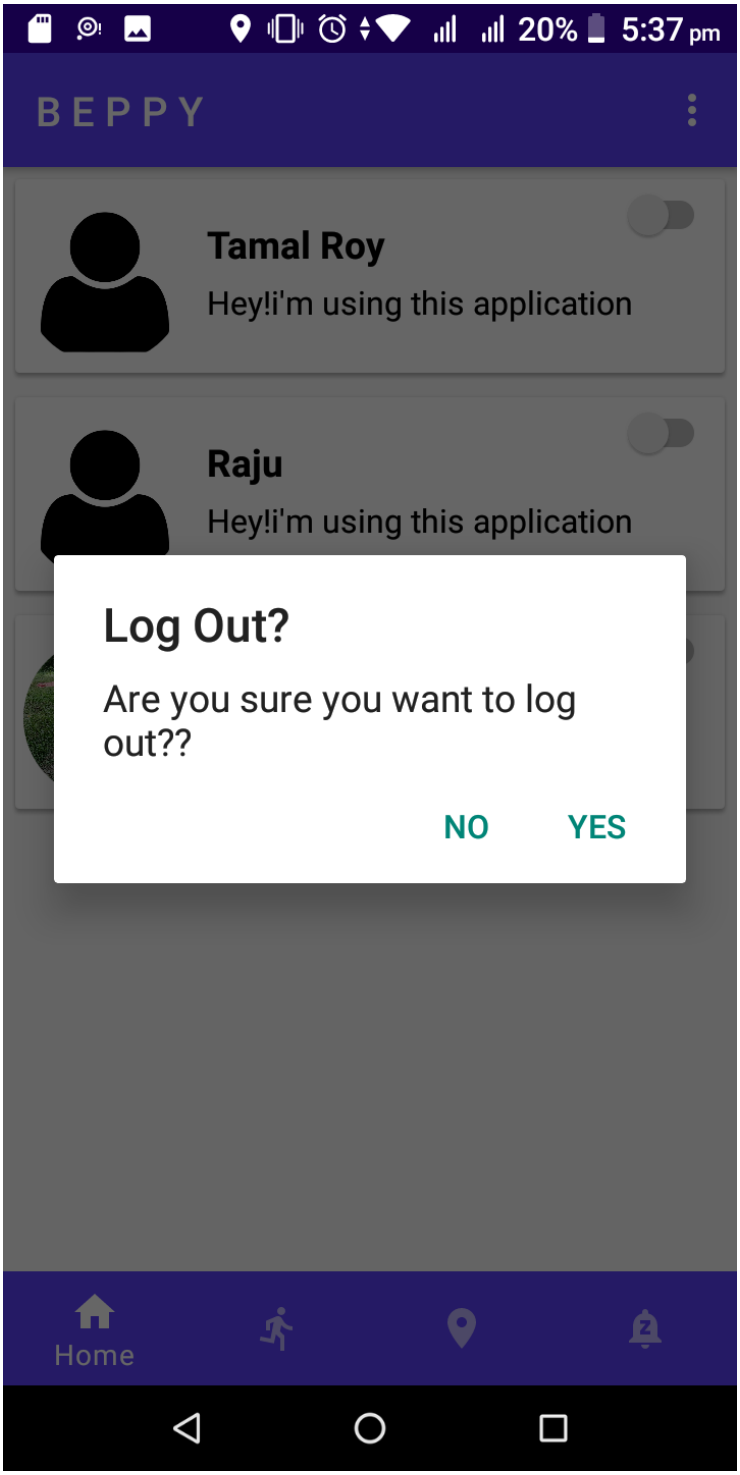


Fig : Log out Dialog .

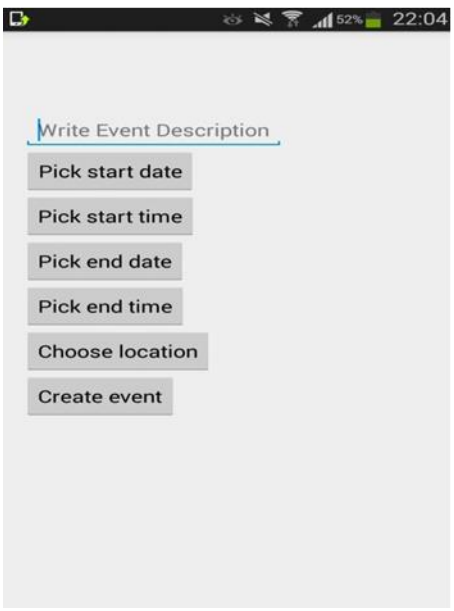


Fig - Creating private event interface

Fig - Selecting start and end date of event interface

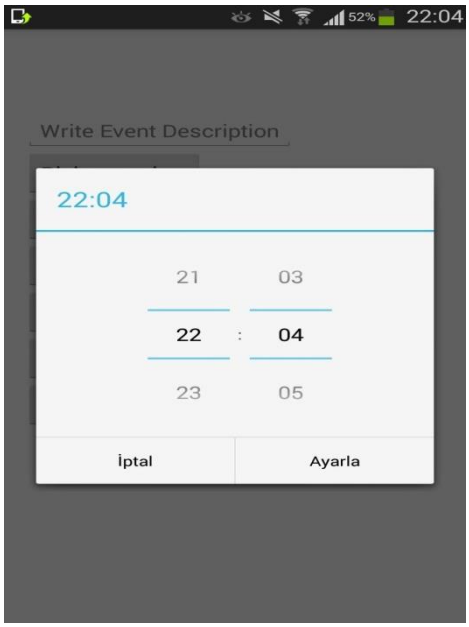


Fig - 3.1.10
Selecting start and
end time of event interface

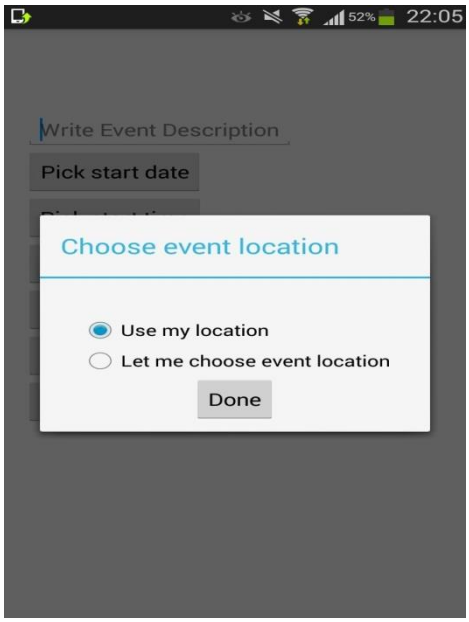


Fig - 3.1.11
Choosing event location interface

Component	Actors
Creating private event	When user click private event on menu interface, he/she is directed to this interface.
Specifying the description of event	User can write the description of event in order to tell the topic of event to other users.
Selecting start date and end date of event	User can adjust the start date and end date of event using this dialog.
Selecting start time and end time of event	User can adjust the start time and end time ıf event using this dialog
Choosing location of event	User can use his/her present location for creating event either user can select other location for creating event with clicking let me choose event location button.
Choosing event location different than present location of user	User can select event location with travelling around map or user can search a location for creating event.
Recording event	User can record event that he/she create with clicking create event button.

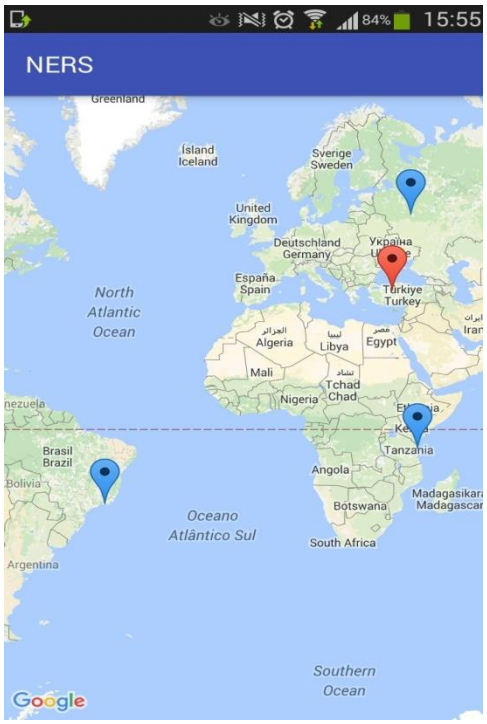


Fig - 3.1.13

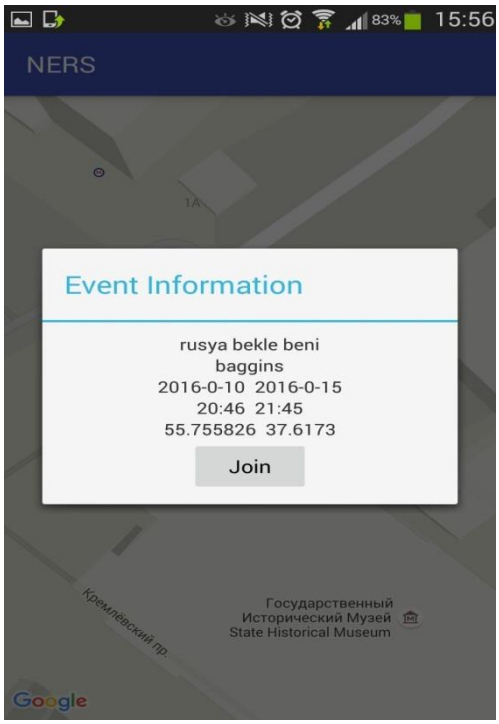


Fig - 3.1.14

Showing events that friends of user create interface

Seeing information of event interface

Component	**Actors**
Showing events that friends of user create	User can see events that friends of his/her with clicking show events button in menu.
Seeing information of event	User can see information of event when he/she click long marker of event in map.
Joining event	User can join event with clicking join button using information dialog.

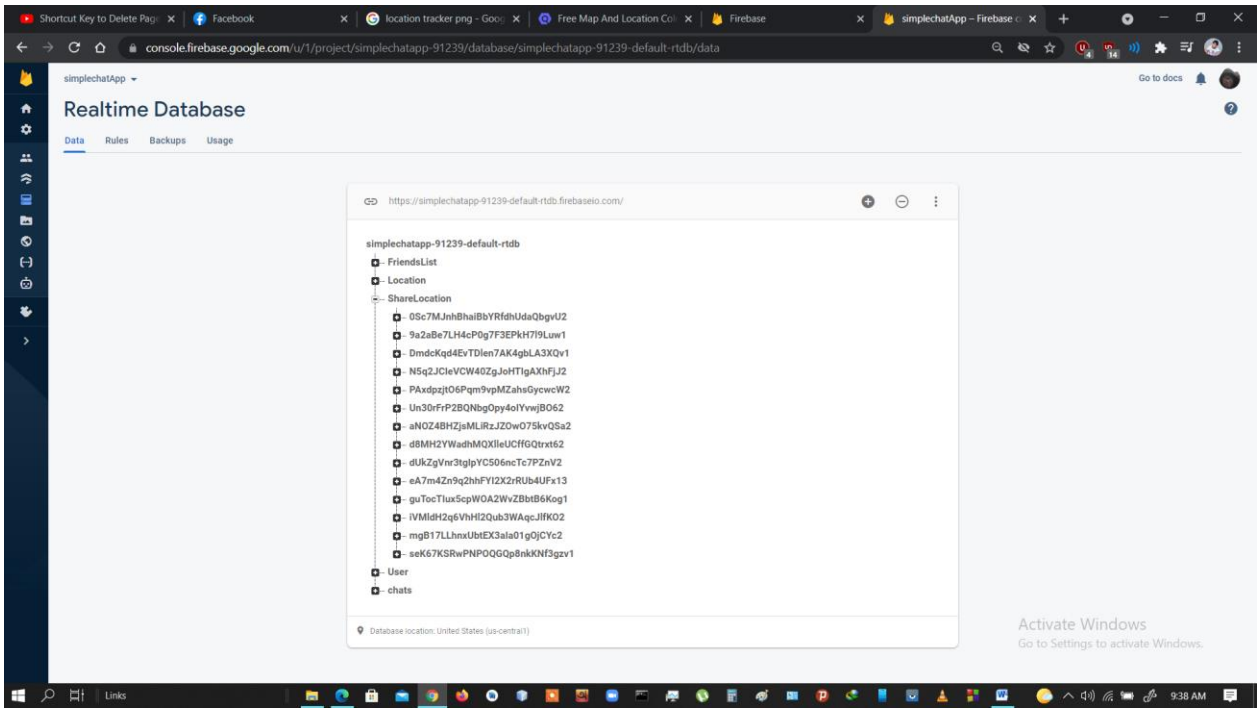


Fig : Firebase Real time Database .

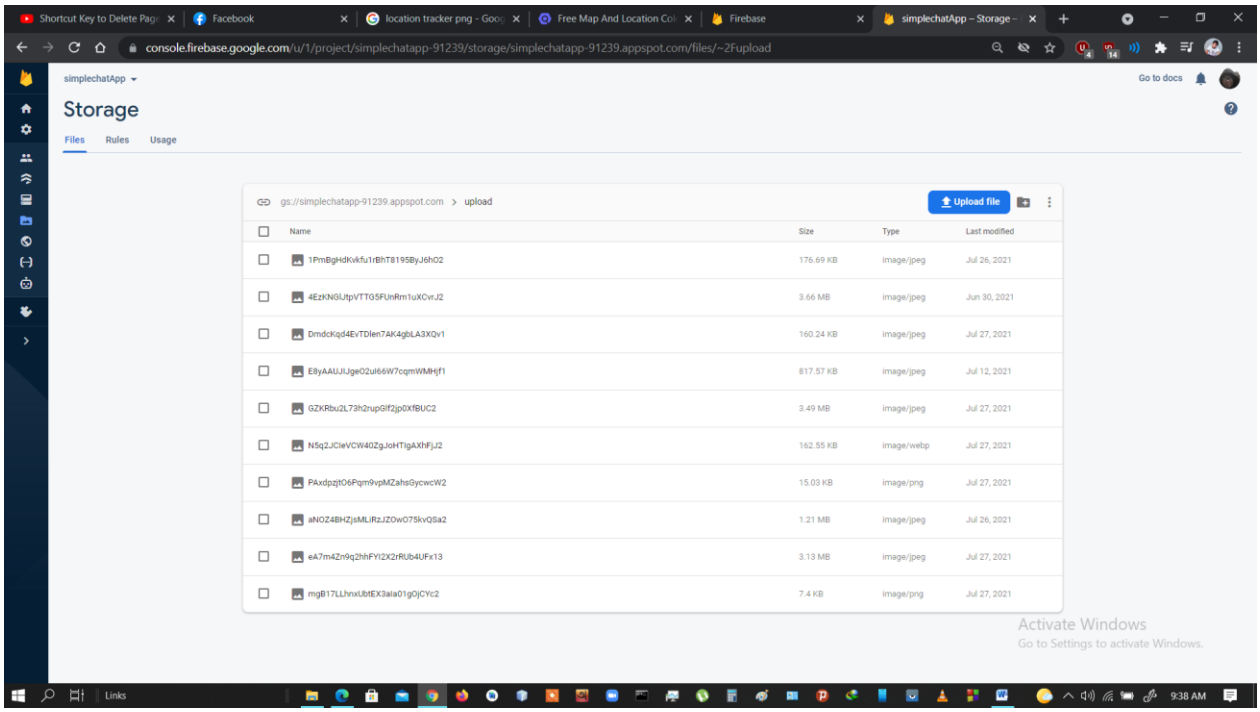


Fig : Firebase Storage database

3.2 FUNCTIONAL REQUIREMENTS

3.2.1) MOBILE APPLICATION COMPONENT

3.2.1.1)FUNCTIONAL REQUIREMENT 1

User shall be able to allow location permission before actual registration process without verification.

3.2.1.2) FUNCTIONAL REQUIREMENT 2

After the allow location permission, user shall be able to register to the application filling out required attributes in register interface.

3.2.1.3)FUNCTIONAL REQUIREMENT 3

User shall be able to delete his/her account from system of application.

3.2.1.4) FUNCTIONAL REQUIREMENT 4

User shall be able to add friend in order to see the location of his/her friend.

3.2.1.5) FUNCTIONAL REQUIREMENT 5

User shall be able to delete friend in his/her friends.

3.2.1.6) FUNCTIONAL REQUIREMENT 6

User shall be able to see location of friends on main map.

3.2.1.7) FUNCTIONAL REQUIREMENT 7

User shall be able to scale the main map according the location of friends.

3.2.1.8) FUNCTIONAL REQUIREMENT 8
User shall be able to see his/her friends.

3.2.1.9)FUNCTIONAL REQUIREMENT 9
User shall be able to chat with one friend while user is seeing his/her location and his/her friend's location

3.2.1.10)FUNCTIONAL REQUIREMENT 10
User shall be able to create private event that friends of user only can see.

3.2.1.11)FUNCTIONAL REQUIREMENT 11
While user is creating private event, user shall be able to select friends in order to share event only with them.

3.2.1.12)FUNCTIONAL REQUIREMENT 12
User shall be able to create public event that all users of application can see.(Other users can send a request to user for joining event.It's user's choice whether he/she accept the request or not.)

3.2.1.13) FUNCTIONAL REQUIREMENT 13
User shall be able to see private events that shared with him/her on map.

3.2.1.14) FUNCTIONAL REQUIREMENT 14
User shall be able to see all public events on map.

1.1.1.15) FUNCTIONAL REQUIREMENT 15
User shall be able to accept request from creator of private event for joining event.

1.1.1.16) FUNCTIONAL REQUIREMENT 16
User shall be able to send a request to creator of public event for joining event.

1.1.1.17)FUNCTIONAL REQUIREMENT 17
User shall be able to join privileged event without sending request to the creator of privileged event.

1.1.1.18)FUNCTIONAL REQUIREMENT 18
User shall be able to receive notifications when friend request comes and event that invited by other users is created.

1.1.1.19)FUNCTIONAL REQUIREMENT 19
User shall be able to see his/her profile.

1.1.1.20)FUNCTIONAL REQUIREMENT 20
User shall be able to see profile of other users.

1.1.1.21)FUNCTIONAL REQUIREMENT 21
User shall be able to edit his/her profile.

1.1.1.22)FUNCTIONAL REQUIREMENT 22
User see events that he/she created and events that he/she joined in events history.

3.2.2 WEB APPLICATION COMPONENT

3.2.1.1 FUNCTIONAL REQUIREMENT 1
Privileged user shall be able to register to the application filling out required attributes in register interface before he/she log in to the application.

3.2.1.2 FUNCTIONAL REQUIREMENT 2
Privileged user shall be able to log in to the system in order to use the application.

3.2.1.3 FUNCTIONAL REQUIREMENT 3
Privileged user shall be able to delete his/her account from system of application.

3.2.1.4 FUNCTIONAL REQUIREMENT 4
Privileged user shall be able to create privileged event that all users of application can see.

3.2.1.5 FUNCTIONAL REQUIREMENT 5
Privileged user shall be able to see events that other privileged users create.

3.2.1.6 FUNCTIONAL REQUIREMENT 6
Privileged user shall be see his/her profile.

3.2.1.7 FUNCTIONAL REQUIREMENT 7
Privileged user shall be see profile of other privileged users.

3.2.1.8 FUNCTIONAL REQUIREMENT 8
Privileged user shall be able to see participations of events that he/she create.

3.2.1.9 FUNCTIONAL REQUIREMENT 9
Privileged user shall be able to see profile of other privileged users.

3.2.1.10 FUNCTIONAL REQUIREMENT 10
Privileged user shall be able to see the events history of him/her.

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 PERFORMANCE REQUIREMENTS

* BEPPY shall be able to support at least 100.000 simultaneous users.The capacity can be extended in the future if needed.
*There will be large amount of information to be processed in database like user profile informations, friendship conditions, event informations, events joining conditions, messages etc. This is why, server will be enough space to overcome this occupation.
*All of the functions such as registration, sending messages, receiving messages, seeing locations of friends on map, seeing events on map shall perform in less than 3 seconds.

3.3.3 SOFTWARE SYSTEM ATTRIBUTES

3.3.3.1 SECURITY

Security issues on application side

*Other applications on user's phone should not reach and manipulate stored data in phone of user.
*Stored data in mobile device and data to send database should be encrypted.
*Sent and received data should be transferred via HTTP connection.
*Password characters should be displayed as black dots in registration interface.
*Password shall be at least 6 characters.
*If user enters his/her password 5 times while he/she is deleting account, informative mail should be sent to the user.

Security issues on database side

*Users should not reach and manipulate stored data of other users in database.
*Data that come from application should be decrypted in database side.

3.3.3.2 AVAILABILITY

*The system should be available for 7 days and 24 hours.
*In the application side, application should be tested against unexpected errors and failures before releasing the first version and updated versions of application. End-product should be error free.

3.3.3.3 MAINTAINABILITY

*While system is developing, version control system such as gitlab should be used in order to reduce complexity, make the system traceable. Also, thanks to git lab, while more than one developer is developing system, if the system has an unwanted crash, code can be recovered.

*When new interface or component is wanted to add the system, any problem

should not occur. System should be implemented in this way.

3.3.3.4 RELIABILITY

*If any interface or component of system does not work properly, informative message about error should be displayed to the users.
*There should be a backup system for holding all stored data of system such as users, events or friendships in case of failure of the system.
*This system should keep the database updated.

3.3.3.5 PORTABILITY

*Since the application is Android application, the system will run on any platform that has Android OS.
*The version of Android Operating system on device should bigger than 3.0.

3.3.4 DESIGN CONSTRAINTS

There is no specific design constraints in this project.

4 DATA MODEL AND DESCRIPTION

4.1 DATA DESCRIPTION

4.1.1 DATA OBJECTS

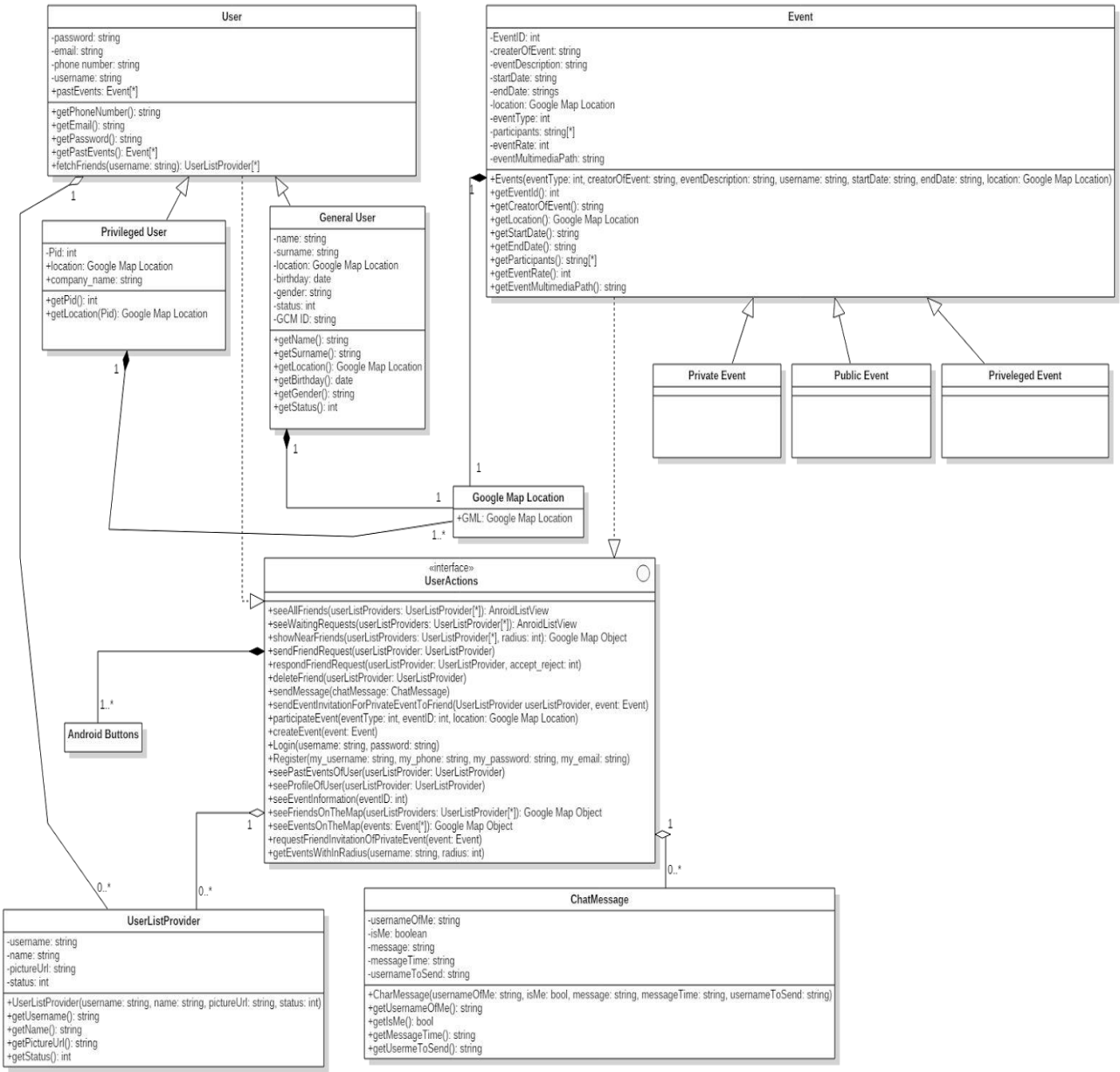


Figure 4.1.1 Class Diagram

In our data model, there are two main classes which are User and Event. In addition, private event, public event and privileged event classes are generalized by Event. Also, user generalizes privileged and general user classes. Moreover, there are two helper classes which are UserListProvider class and ChatMessage

class. To implement each functional requirements, we use user actions interface, in the implementation of the interface all classes are used to handle user requests. UserListProvider is used almost in every user action. ChatMessage class is used for messaging activity.

4.1.2 DATA DICTIONARY

Objects	**Description**
User	User is a major class which is used from user actions interface.User class is a singleton class. **Attributes:** username: holds username of user, username is unique. password: holds passwords of user. email: holds email of user. pastEvents: holds past events of user. **Functions:** getEmail(): return email of user. getPassword(): return passwords of user. getPastEvents():return past events of user. fetchFriends(): returns friends of a user.

Privileged User	Privileged User is generalized by User class. This class represents the private entities like hangout places. **Attributes:** Pid: hold privileged user id, it is also unique. location: holds the location of the place represented in Google Map Location companyName: holds the name of the company name. **Functions:** getPid(): returns Pid of the privileged user. getLocation(): returns location of the place setPid(params): sets the Pid of the privileged user. **Attributes:** name: holds name of the user surname: holds surname of the user location: holds the location of the user, this is represented by Google Map Location birthday: holds the birthday of the user gender: holds the gender of the user status: holds the online or offline status of the user. **Functions:** getName(): return name of the user getSurname(): return surname of the user getLocation(): return location of the user getBirthday(): return birthday of user getGender(): return gender of the user getStatus(): return status of the user setName(params): sets the name of the user setSurname(params): sets the surname of the user setBirthday(params): sets the birthday of the user setGender(params): sets the gender of the user
General User	
General User	General User is generalized by User class. This class represent the normal users of the application.

Event	Event is a major class which is used in user actions interface. Event is singleton class. **Attributes:** eventID: holds eventId of the event, this is unique creatorOfEvent: holds username of the creater of the event eventDescription: holds the description of the event like, purpose of the event, how will be the event. eventType: holds the type of the event like private, public or privileged. startDate: holds the startdate of the event endDate: holds the enddate of the ever location: holds the location of the user which is represented as Google Map Location. participants: holds usernames of participants of the event eventRate: holds eventRate of the event which is from 0 to 10 eventMultimediaPath: holds directory path of the representative picture of the event **Functions:** getEventID(): return identifier of the event getCreatorOfEvent(): return username of the creator of the event getLocation(): return location of the event getStartDate(): return startdate of the event getEndDate(): return enddate of the event getParticipants(): return usernames of participants of the event getEventRate(): return event rate of the event. getEventMultimediaPath(): return directory path of the multimedia of event. getEventType(): return the type of the event
Private Event	Private event class is generalized by event class. When event type is 0 the event is private event. The participants of the event are determined by creator.
Public Event	Public event is generalized by event class. When event type is 1 the event is public. Anyone can participate this event.
Privileged	Privileged Event is generalized by event class. When event

Event	type is 2 the event is privileged. This type of event belongs to privileged user. Only privileged users can create this type of event.
User ListProvider	UserListProvider is a helper class for user actions interface. This class is widely used by interface and other classes. It consists most widely used attributes of the User class. Also it is a singleton class. **Attributes:** username: holds username of the user name: holds the name of the user pictureurl: holds the directory path of the user picture status: holds the offline or online status of the user. **Functions:** UserListProvider(params): constructor for the userlistprovider class getUsername(): return username of the user getName(): return name of the user getPictureUrl(): return the directory path of the user picture getStatus(): return the status of the user.
Chat Message	ChatMessage is a helper class for actions interface. This class is used in sending and fetching messages. ChatMessage is a singleton class. **Attributes:** usernameOfMe: holds username of the sender side isMe: holds if the message is sended by me or get by opponent side message: holds message text of the message messageTime: holds the sending time of the message. usernameToSend: holds the username of the the receiver **Functions:** ChatMessage(params): constructor for the ChatMessage class getUsernameOfMe(): return username of the sender side getIsMe: return if the is sended by me or not getMessageTime(): return message time getUsernameToSend(): return username of the receiver side.

Interfaces	**Descriptions**
UserActions	**Functions:** Register(params): takes action when user click on register button. After click, if the information is appropriate the register information is recorded on database. sendFriendRequest(params): takes action when user clicks on add friend button. After click, waiting request is recorded on database. fetchWaitingRequests(paramas): takes action when user click on waiting requests tab, after click, the waiting requests are fetched from database. seeWaitingRequests(): works with fetchWaitingRequests function, after fetching this function show these datas to the user. respondFriendRequest(params): after user call of seeWaitingRequests user is able to see waiting requests, from here user can click on accept or reject friend request. After that click, this function takes on and depending accept or reject it organizes database. fetchFriends(params): this function is called by some other funtions. it fetches friends of a user from database. seeAllFriends(): works with fetchFriends function, takes action when user clicks on Friends, after fetching process this function shows fetched data to user. deleteFriend(params): takes action when user clicks on delete friend button. Afer click, the friend relation on the database is deleted. sendMessage(params): takes action when user clicks on send button on the chat screen. after click, the message is sended to receiver side. seeFriendsOnTheMap(params): this function automatically works when user go to the map activity and it also updates the locations of the friends periodically. showNearFriends(params): this function works with seeFriendsOnTheMap and takes action when user choose a radius for seeing near friends. after choose this function shows the friends who are inside the radius only.

	seeEventsOnTheMap(params): this function automatically works when user wants to see the events. this function also updates the state of the events. getEventsWithInRadius(params): this function works with seeEventsOnTheMap and takes action when user choose a radius for events. After choose, this function shows the events who are inside the radius. participateEvent(params): when a user want to participate a public or privileged event he/she can click on participate after click the participants relation on the database is updated. seeEventInformation(params): when user clicks on an event marker this function takes action and show the information of event. seeProfileOfUser(params): when a user clicks on another profile of user this function takes action and show the profile of user. seePastEventsOfUser(params): this function takes action when user clicks on past events of an user. this function fetches information from database and show them to user. CreateEvent(params): this function takes action when user clicks on create event button. after the necessary information for the event is given from user user clicks on the create button and the event is recorded into the database. sendEventInvitationForPrivateEvent(params): when user creates a private event the user can decide who can participate the event. after deciding user can send invitation to his/her friends. this function takes action after clicking invite button. the the participants relation on the database is updated with this function requestFriendInvitationOfPrivateEvent(paramas): this function takes action after clicking on accept or reject button. when user gets invitation from his/her friends the user can click on accept or reject button. after clicking button the database is updated with respect to decision.

PLANNING

4.2 TEAM STRUCTURE

We meet our advisor Mr Fazle Mohammed Tawsif , Assistant Professor Dept. of Software Engineering IICT,SUST. We show our application to them and take advice from them what and how will we do in our project. In addition to these, we meet and work on project five or six a month with teammate Members of our group and their focus' are explained below:

Md Ashiqul Hasan Shakil - Developer, Massaging side Development
MD Fahim Mia - Developer, MAP side Development

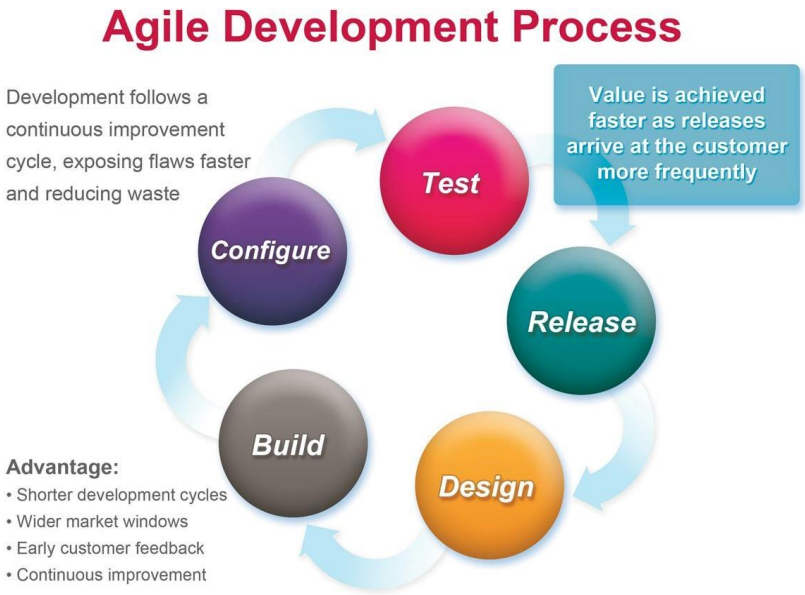
4.3 ESTIMATED SCHEDULE

At the end of this semester, we are planning to finish a prototype of application. With this prototype, user can see locations of all friends, create private events and see private events that his/her friends create.
At the end of this academic year, we are planning to finish the application. Much more developed interface will be added to the application. After that, we will put our application to Google Play Store. In this end product, user can also create public event and see public events. In addition to these, privileged users can create privileged events and users of application see these privileged events. Finally, we will work on advertisement techniques for marketing our product.

4.4 PROCESS MODEL

We will apply agile model for our location based BEPPY application because our software is developed in incremental, rapid cycles. This is why, system can respond quickly to changing requirements. Since agile model is based on iterative approach, each iteration involves planning, requirement analysis, design, implementation and testing. Each iteration takes 4 weeks. After we successfully build prototype of our application, our project's development will

be continued according to agile model in the second semester.



5 CONCLUSION

This Software Requirements Specification Document(SRS) is a guideline to understand the project for both customers and developers. The customer or developer who read this document can have reach details of project. Interfaces of project, detailed functional and non-functional requirements, data and behavioral model of project are stated in this document. This document will be useful while designing and developing the project.