GROUP 2
Software Engineering & Management

ME Software Systems
Department of CSIS
BITS Pilani

WHEREABOUTS DESIGN SPECIFICATION DOCUMENT V1.0

NOVEMBER 7, 2016

This document describes the overall solution of the Whereabouts Software described in the Whereabouts Software Requirements Specification

TABLE OF CONTENTS

1. Introduction

1.1. Purpose of this document

The purpose of this document is to describe the implementation of the Whereabouts Software described in the Whereabouts Software Requirements Specification. Whereabouts is an android application which provides a location based messaging service to users.

1.2. Scope of the development project

The android application Whereabouts will allow the students at BITS, Pilani to locate each other on campus and provide them instant messaging service.

1.3. Definitions, acronyms, and abbreviations

App Android application

BITS Birla Institute of Technology and Science

GPS Global positioning system

Pdf Portable document format

Peer When a user accepts the connection request from another user, both users are said to peers of each other.

PU Placement Unit, BITS, Pilani

User Anyone using the application is a User.

1.4. References

[1] A. Rezi and M. Allam, "Techniques in array processing by means of transformations, " in Communications and Multimedia, International Conference on Telecommunications and Multimedia (TEMU), IEEE, 2016

[2] Penghui Li and Yan Chen, “Implementation of Cloud Messaging System Based on GCM Service” in Proceedings, ICCIS '13 Proceedings of the 2013 International Conference on Computational and Information Sciences, 24 October 2013, pp 1509-1512

[3] Yavuz Selim Yilmaz, “Google Cloud Messaging (GCM): An Evaluation” in GLOBECOM, 12 February 2015, Semantics Scholar[4] Xiufeng Liu, ”Location-based mobile instant messaging system”, , 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), IEEE, 20122. General Description

1.5. Overview of document

The DS for this system contains the description of the system including its detailed design, various modules and other components, constraints and other features in section 2. In section 3 various components of the system are explained in a detailed manner. Section 7 contains the development plan associated with the project, & instead of specifying Abstract API in Section 8, it is attached as a separate document . Comments on design decisions are specified in section 9.

2. System design description

2.1. Introduction

We will be applying *Structured Design* approach to develop our Android application. The various tools are specified below:

Android Studio Android Studio is the official integrated development environment (IDE) for Android platform development.

Draw.io To draw various diagrams such as DFDs, Structure Charts & State diagrams.

2.2. Overview of modules / components

The brief descriptions of all components are given below:

Authentication Component: This module is used for signup and login.

Map Rendering Component: Locates the user and their peers on the map.

Instant Messaging Component: This module provides the user to send instant messages to his/her peer.

Peer Management Component: This module is designed to provide users, features of adding peers, view peer list and managing peer requests.

Drop Message Component: This module is designed to provide a message delivery service to the users based on a predefined location and time- a user can schedule a message to be delivered to a peer when the peer is near a specified location during a specified time interval provided by the user.

Track me Component: This module is used for sending location details of the user to a peer.

2.3. Constraints and Assumptions

The design constraints are given below :

User Interface Constraints:

UI-1. The app shall provide a map view of BITS, Pilani campus and location of users using markers on the map.
UI-2. The app shall provide a screen for sending message to a particular peer

DUI-1. The application has been designed to display the map view of BITS, Pilani by using Google Map API.

DUI-2. A chat screen has be designed for instant messaging functionality.

Software Interfaces Constraints:

- Use Firebase as a cloud services provider and backend as a service that will provide the application with database & authentication support, cloud based messaging and usage statistics.

➔ Google Map API is used to fetch the location and Google sign in is used for login the application.

➔ Firebase platform is used to store the messages and its APIs are used to implement chat functionality.

Communication Interfaces Constraints:

- SSE i.e. Server Sent Event an option for the REST API for real time updates to be streamed back to application.
- Application will use websockets with long polling fallback to send messages to Firebase server.

➔ The communication interfaces are handled by the Firebase platform internally.

Design and Implementation Constraints:

CO-1. Mobile device should be compatible with Android 4.4.4(Kitkat) and above.
CO-2. All code will be written in JAVA.

DCO-1. Design of application is based on usage of FCM for messaging

DCO-2. The design takes care that the application will be able to run on all device with Android version 19(Kitkat) or above by mentioning it in the minimum supported Android version in the .apk file.

Performance Constraints:

PE-1. Map shall take no longer than 5 seconds to load onto the screen after the user logs in.
PE-2. The system shall display messages to peer within 4 seconds after the user submits message to be sent the system over a 3G network.
PE-3. The location markers should get updated within 5 seconds.

DPE-1. The display map functionality has been designed to refresh the contents of map displayed on user screen in every 5 seconds.

DPE-2. Firebase real time database and Firebase event handler APIs are used to instantly deliver messages to peers.

Security Constraints:

SE-1. Users shall be required to log in using Google account to the application usage.
SE-2. Only peers shall be allowed to track and message each other.
SE-3. The messaging history of users will be achieved only on user device and will be stored on the server for buffering purpose only and then after message delivery will be deleted from the server.

DSE-1. The peer requests in the peer management module, are used to add peers to a user. A separate table is created for storing the peers for each user and peer operations(tracking and chat) can only be performed for peers by a user.

DSE-2. Adding and removing of messages is designed to be implemented by using the event handler APIs of Firebase database.

Safety Constraints:

SA-1. The app shall provide the provision of retrieval of only those messages which are not yet received on the user device in case of abrupt power/system failure of mobile device.

DSA-1. The delivery of messages which are not yet received on the user device in case of abrupt power/system failure of mobile device is handled by Firebase event handlers.

2.4. Structure and Relationships

The context diagrams, level-1, level-2 DFDs and Structure charts depicting the relationships between different components and their sub-modules are given below:

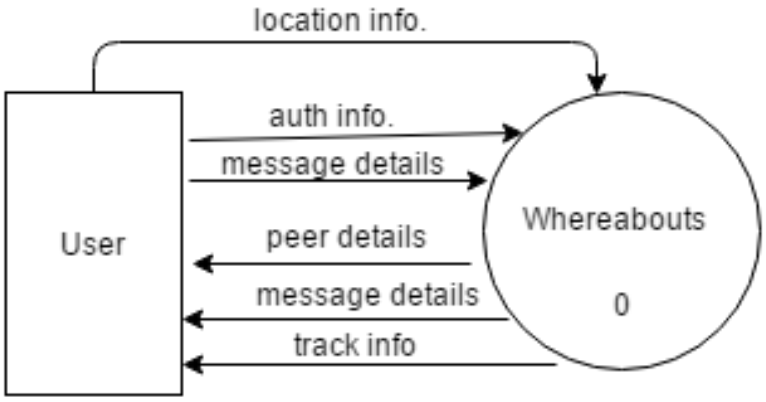


fig. Level 0 DFD

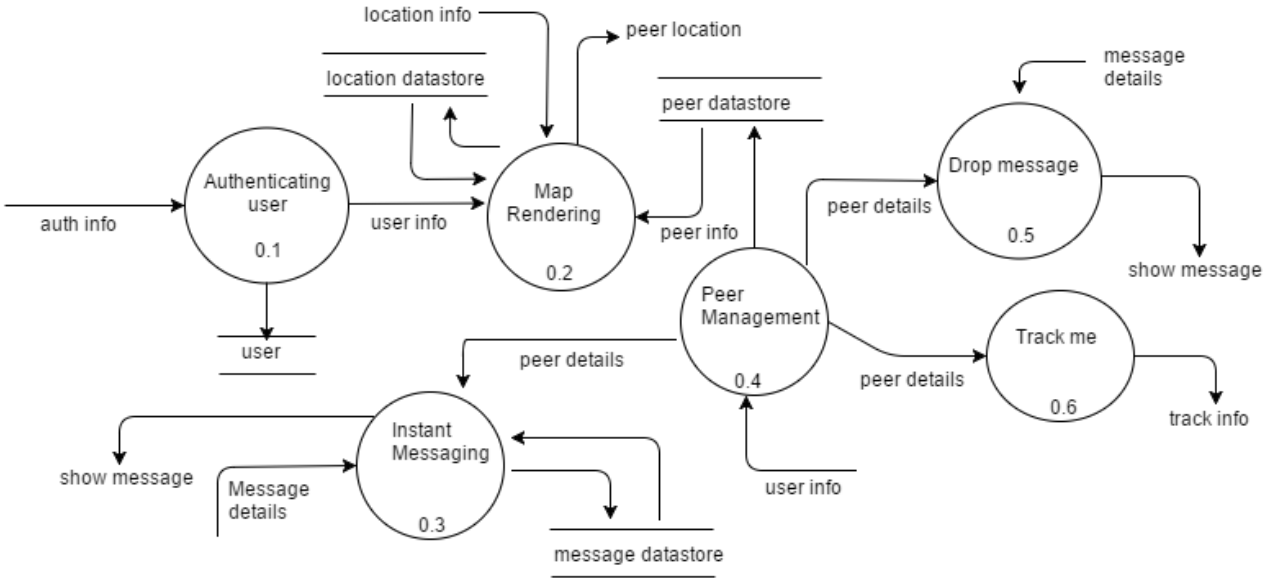


fig. Level 1 DFD

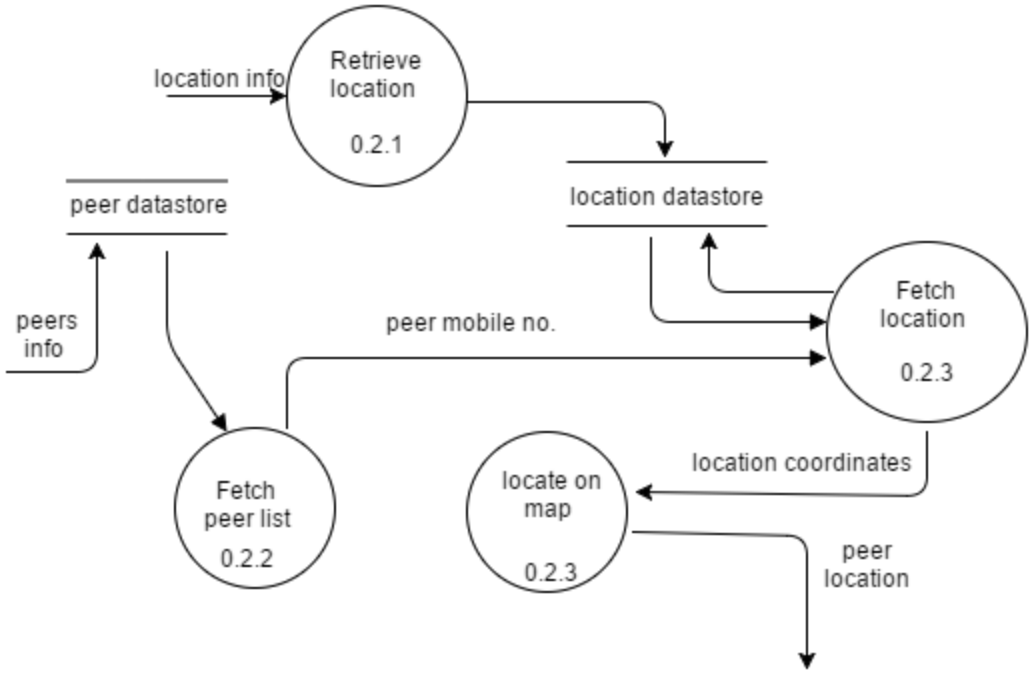


fig 2.1 Level 2 DFD of Map rendering component

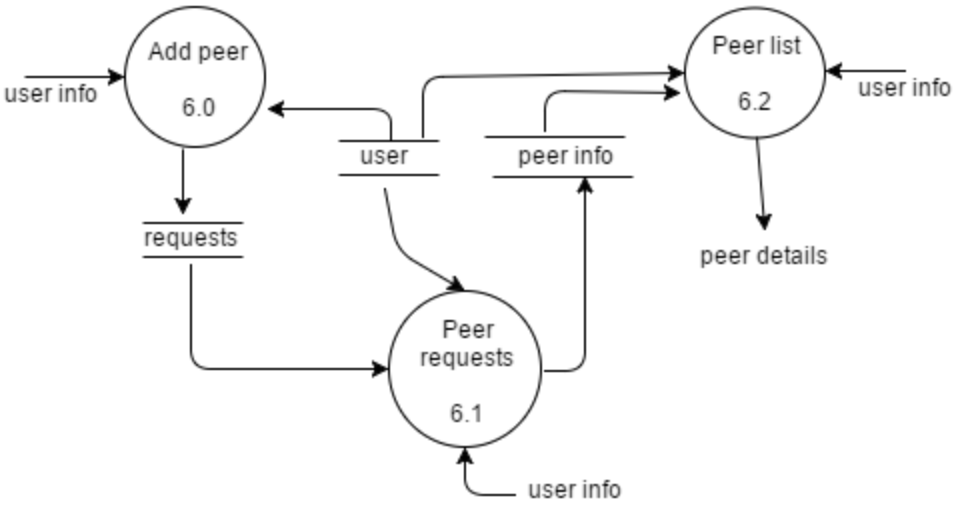


fig 2.3 Level 2 DFD of Peer management component

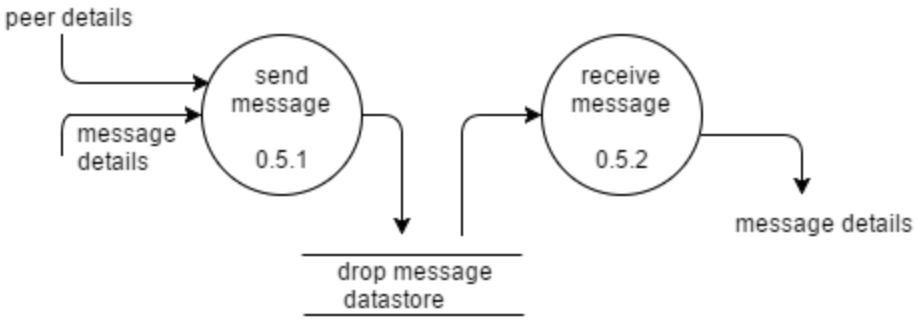


fig 2.2 Level 2 DFD of Drop message component

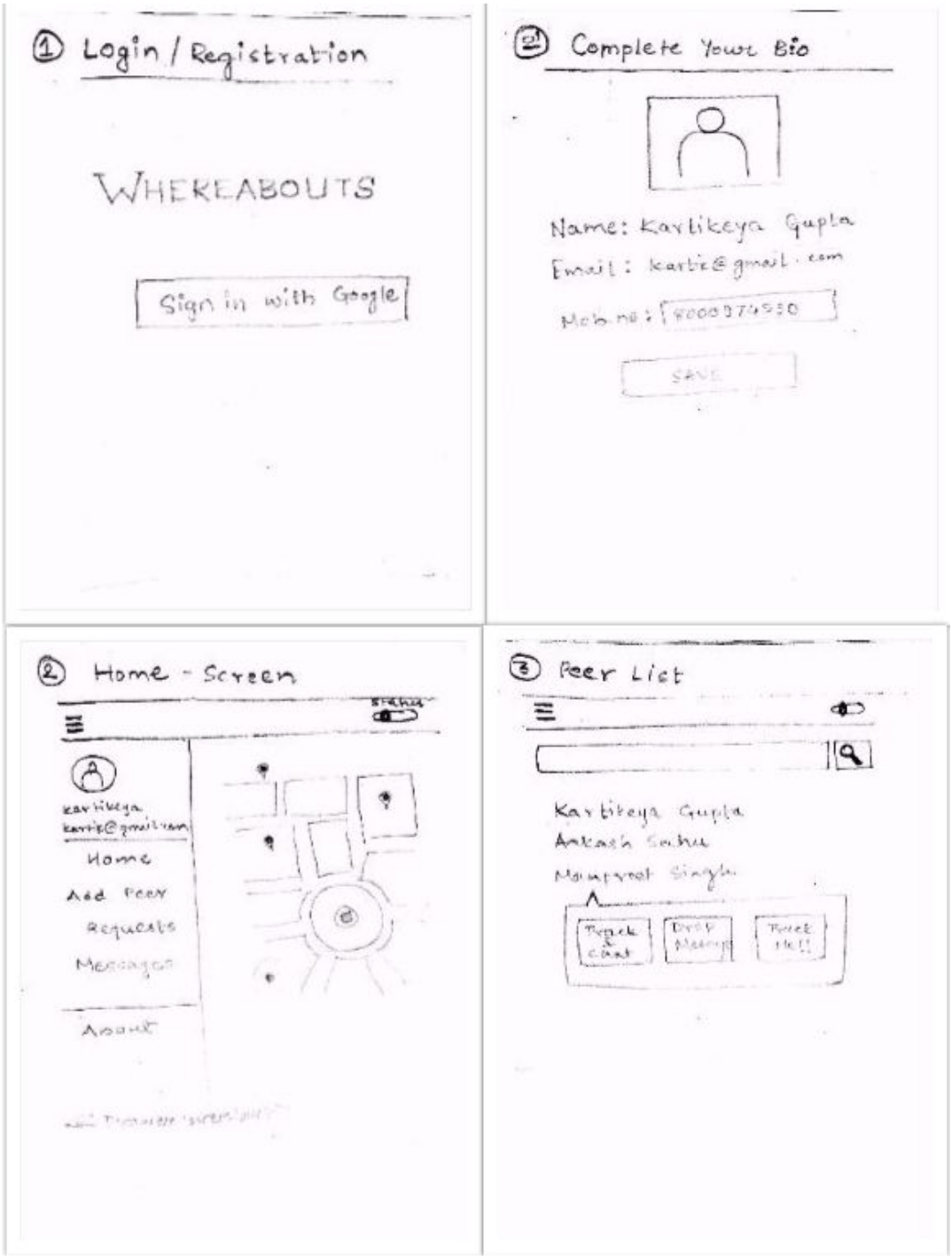
Data Dictionary :-

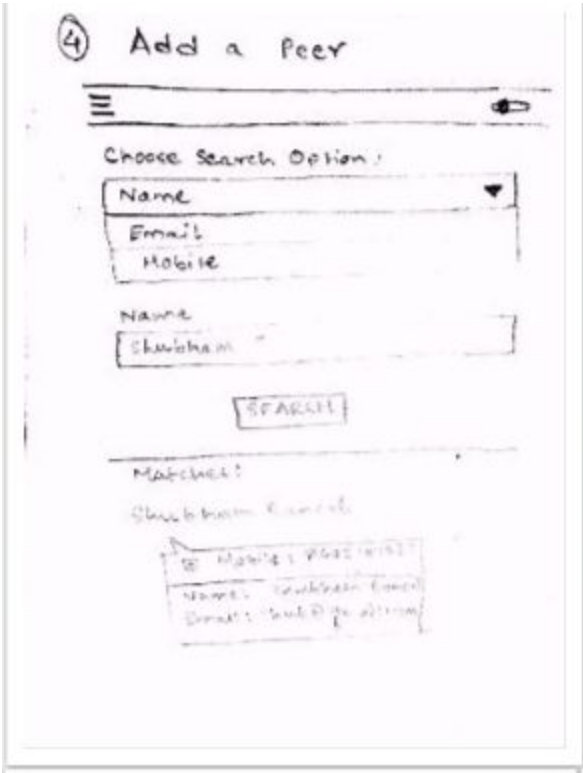
#	DATA	MEANING
1	auth info	Gmail login info and phone number of app user
2	location coordinates	Current latitude and longitude of peers
3	location info	Current latitude and longitude of app user
4	message details	source/destination phone number, message text, time stamp and [location]
5	peer details	Phone number, email and name of peer
6	peer location	Location Marker of all peers

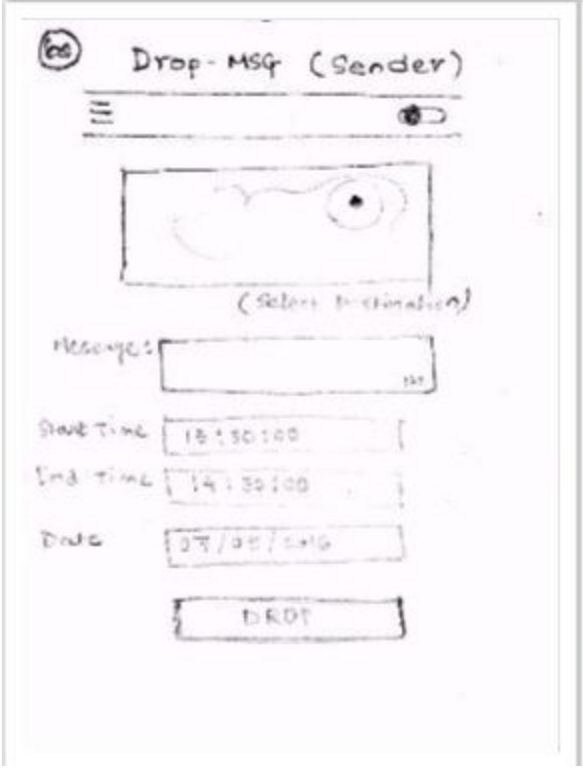
7	peer mobile no	Phone number of peer
8	track info	Phone number, latitude and longitude of this phone number
9	user info	Phone number, email and name of app user

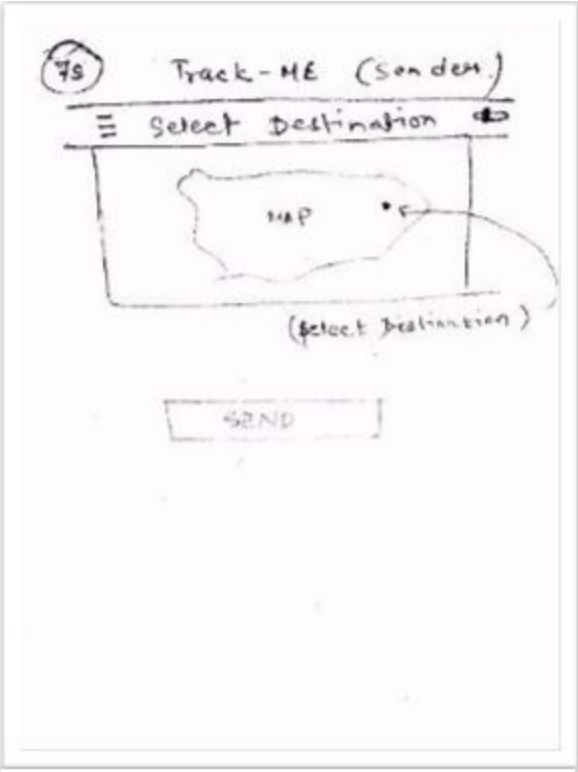
2.5. User interface issues

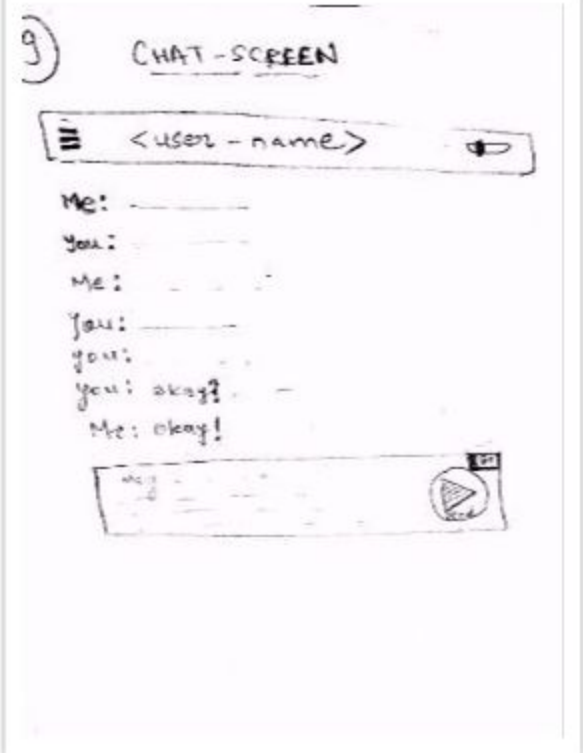
The main screen designs are depicted below in the form of sketches which are according to the user interface constraints given in the Software Requirements Specifications :











3. Detailed description of components

3.1. Authentication Component

The user is provided with “login with Google” functionality on application startup for the first time. He/she then logs in with the application with the credentials used in his/her device for Google account. These details are stored in the application environment and the user need not sign in every time he/she starts the application.

On first time login, the user is prompted to provide his/her phone number. After getting the phone number , home screen of application is presented to the user. On subsequent application startup, the user sees the application home screen.

3.2. Map Rendering Component

This modules displays the current location of the users and their peers on the Google map API. The module fetches the current location of the user and stores/updates it into the firebase database. The stored location of the peers are retrieve and update its location on the map. A chat screen will be prompt if the pointer of any peer is tapped.

3.3. Instant Messaging Component

The user is displayed his chat screen with the peer, with previous conversation being displayed .The user types in the new message he/she wants to send to the peer and on clicking on send button the message would be received by the peer. In similar way, the messages sent by the peer will be displayed on the user chat screen.

3.4. Peer Management Component

This module provides the user the functionality of adding peer through sending peer request. This request details will be stored in a database and the other peer receives notification in his peer request section . If user accepts the request then name entry of each other will appear in peer list section.

3.5. Drop Message Component

This module provides the user with the functionality to schedule a message that will be delivered to a peer only if that peer happens to be around a particular place in a particular time duration- both the place and duration is provided by the user. These details will be stored in a database table and the message will be delivered to the peer in form a notification if he/she is present at the user specified location within the time interval mentioned by the user. If the selected peer is not present at the mentioned location within specified time interval, the message will not be delivered to the peer and its lifetime will be timed out-such messages will be removed from the database and will never be received by the peer.

3.6. Track me Component

The user selects a peer and a destination to which the user intends to visit. On clicking the track me option, a map will be displayed on the selected peer's device and the trace of user location will be displayed on the map along with the destination the user intends to visit.

4. Reuse and relationships to other products

Not Applicable to this application.

5. Design decisions and tradeoffs

Decision D01: Use of Firebase platform for application development

Issue	In existing platform transfer of messages requires use of separate server side application development, separate authentication tools and data storage.
Decision	Use Firebase platform for development of android application.
Status	Approved.
Grouping	Platform selection.
Assumptions	We must use existing Firebase APIs for authentication, message management and storage.
Constraints	Android SDK version 19 or above will be supported.

Positions	Use of existing GCM for message management.
Argument	Firebase platform will help in application development under an umbrella structure providing all the necessary APIs for required functionalities that will continued to be used for a considerable long time. On other hand using GCM platform will render the application to be migrated to FCM very soon as the GCM support has been depreciated.
Implications	The team will need to develop a real-time interface between Firebase platform available online and client applications. Firebase will become a mission-critical platform as application availability will solely depend Firebase availability.
Related decisions	NA
Related requirements	See Table 1.
Related artifacts	None.
Related principles	Reuse existing infrastructure, buy before build. Use proven technologies.
Notes	None.

S.No	**Selection criteria**	**Firebase**	**GCM**
1	User authentication service provided?	Yes	Yes
2	Real time database storage provided?	Yes	No
3	APIs for services provided?	Yes	Yes
4	Will continued to be used in near future?	Yes	No

Table 1: Alternatives for platform selection for application development

6. Pseudo code for components

6.1. Authentication Component

```
authenticate(email, password)
BEGIN
      //authenticate using google sign-in
      client := FirebaseGoogleSignInClient()
      flagSuccess := client.authenticate(email, password)
      IF flagSuccess = True THEN
            return True
      else THEN
            return False
END

void completeRegisteration()
BEGIN
      //ask user to enter phone number
      //without phone number, registeration is not possible
      auth := FirebaseAuthenticaion()
      name := auth.getName()
      email := auth.getEmail()
      //get phone number from user
      //phone number is the primary key node for userTable
      phoneNumber := inputPhone.getValue()

      //add this new phonenumber in userTable node
      database := FirebaseDatabase()
      userTableReference := database.getNode("userTable")
      userTableReference.addUser(phoneNumber,email,name)
END
```

6.2. Map Rendering Component

```
displayMap()
BEGIN
      //render map
END
```

```
setLocation()
BEGIN
      //gets current location coordinates
      // and update it to database
      currLocation := LocationAPI.getCordii()

      //get firebase database instance
database := FirebaseDatabase()
locationTable  := database.getNode("locationTable")
locationTable.addLocation(myPhoneNumber,currLocation)
END
```

```
getPeerDetails()
BEGIN
      list := empty

      //fetch data from database
      database := FirebaseDatabase()
      peersTable  := database.getNode("peersTable")
      peerNodes := peersTable.allChildNodes(myPhoneNumber)
      REPEAT for each node in peerNodes
      BEGIN
            list.add(node.data)
      END

      Return list
END
```

```
locationDisplay( list )
BEGIN
      REPEAT for each element of list
      BEGIN
            locatePeer(node)      //locate the peers on map
      END
END
```

```
getLocation(phoneNumber)
BEGIN
//fetch data from database

      database := FirebaseDatabase()
locationTable  := database.getNode("locationTable")

return locationTable.getLocation(phoneNumbers)
END
```

6.3. Instant Messaging Component

```
sendMessage(sourceNumber, destinationNumber)
BEGIN
      message := inputMessage.getValue()
      timestamp := System.currentDate()

      database := FirebaseDatabase()
//add message into messageSentTable
      messageSentTable := database.getNode("messageSent")

messageSentTable.addMessage(sourceNumber,destinationNumber,message,ti
mestamp)

      //add message in messageRecieved table
     messageRecievedTable := database.getNode("messageRecieved")

messageRecievedTable.addMessage(sourceNumber,destinationNumber,messag
e,timestamp)
  END

receiveMessage(sourceNumber, destinationNumber)
BEGIN
      database := FirebaseDatabase()
      //fetch messages from messageReceived table
      messageRecievedTable := database.getNode("messageRecieved")

      sourceNodes := messageRecievedTable.getAllSourceNodes()
      REPEAT for each source in sourceNodes
```

```
      BEGIN
            chatWidget.display(Source.getMessage())
      END
END
```

6.4. Peer Management Component

Sub-module: ADD PEER

```
sendRequest(phoneNumber)
BEGIN
      //get firebase database instance
      //update peerRequests table
      database := FirebaseDatabase()
        peerRequestsTable := database.getNode("peerRequests")
        peerRequestsTable.addPeer(phoneNumber,myPhoneNumber)

END

searchPeer()
BEGIN
      //get search criteria
      searchType := dropDownSearchType.getValue()

      searchResultList := empty

      SWITCH searchType
      BEGIN
            case Phone:
                  searchResultList  :=
searchViaPhone(inputSearch.getValue())
            case Email:
                  searchResultList  :=
searchViaEmail(inputSearch.getValue())
            case Name:
                  searchResultList  :=
searchViaName(inputSearch.getValue())
      END

END
```

```
searchViaPhone(searchString)
BEGIN
      database := FirebaseDatabase()
      userTableReference := database.getNode("userTable")
      phoneNode := userTableReference.findNode(searchString)

      IF phoneNode.exists() = True THEN
            list.add(phoneNode.data)

      return list

END


searchViaEmail(searchString)
BEGIN
        database := FirebaseDatabase()
      userTableReference := database.getNode("userTable")

      //iterate through all nodes of userTable and check email match
      REPEAT for each node in userTableReference
      BEGIN
            IF node.matchEmail(searchString) = True THEN
                  list.add(node.data)

      END
END


searchViaName(searchString)
BEGIN
      database := FirebaseDatabase()
      userTableReference := database.getNode("userTable")

      REPEAT for each node in userTableReference
      BEGIN
            IF node.matchName(searchString) = True THEN
                  list.add(node.data)

        END
END
```

```
Sub-module: PEER LIST
getPeerList()
BEGIN
      //get firebase database instance
      //fetch all peers of myphoneNumber in peersTable
      database := FirebaseDatabase()
      peersTableReference := database.getNode(“peersTable”)
      myPhoneNodeReference :=
peerTableReference.getNode(myPhoneNumber)
      REPEAT for each node in myPhoneNodeReference
      BEGIN
            List.add(node.data)
      END
END
```

Sub-module: PEER REQUEST

```
getPeerRequestList()
BEGIN
      //get firebase database instance
      //fetch all peer requests from peerRequest Table
database := FirebaseDatabase()
      peerRequestTableReference :=
database.getNode(“peerRequestTable”)
      myPhoneNodeReference :=
peerRequestTableReference.getNode(myPhoneNumber)
      REPEAT for each node in myPhoneNodeReference
      BEGIN
            List.add(node.data)
      END
END
```

```
acceptRequest(phoneNumber)
BEGIN
      //add new entry in peer table
      database := FirebaseDatabase()
      peerTableReference := database.getNode(“PeerTable”)
      peerTableReference.addPeer(myPhoneNumber,phoneNumber)
END
```

```
rejectRequest(phoneNumber)
```

```
BEGIN
      //delete entry from peerRequest table
      database := FirebaseDatabase()
      peerRequestTable := database.getNode(“PeerRequestTable”)
      myPhoneNodeReference := peerRequestTable.getNode(myPhoneNumber)
myPhoneNodeReference.removeNode(phoneNumber)
END
```

6.5. Drop Message Component

```
createDropMessage()
BEGIN
    peerNumber := dropDownPeerList.getValue()
    message := inputMessage.getValue()
    date := datePickerInputDate.getValue()
    starttime := timepickerInputStartTime.getValue()
    ENDtime := timepickerInputENDTime.getValue()
    location := locationPickerInputLocation.getValue()

   // add new entry into dropMessage table
    database := FirebaseDatabase()
    dropMessageNode := database.getNode(“dropMessage”)

dropMessageNode.addMessage(peerNumber,myPhoneNumber,message,StartTime
,ENDTime)
END
```

```
retrieveAllDropMessages()
BEGIN
     database := FirebaseDatabase()
     dropMessageNode := database.getNode(“dropMessage”)
     myphoneNumberNode := dropMessageNode.getNode(myPhoneNumber)
     REPEAT for each sourceNode in myphoneNumberNode
     BEGIN
          Message := sourceNode.getLeafNodeValue(“Message”)
             Date := sourceNode.getLeafNodeValue(“date”)
             Starttime := sourceNode.getLeafNodeValue
getValue(“starttime”)
             ENDtime :=
sourceNode.getLeafNodeValue.getValue(“ENDtime”)
```

```
            Lati :=
getLati(sourceNode.getLeafNodeValue.getValue("location"))
            Longi :=
getLongi(sourceNode.getLeafNodeValue.getValue("location"))

         //set geo-fencing parameters
         // THEN start geo-fencing background service
         setGeofence(Message,Lati,Longi)
      END
END

showNotification(geofence,date,starttime,ENDtime,message,sourceNumber)
BEGIN
IF peer.location in geofence and starttime<=system.time<=ENDtime THEN
     displayNotIFication(message)
     removeGeofence(instanceGeofence)

      //now remove message from database
     database := FirebaseDatabase()
     dropMessageNode := database.getNode("dropMessage")
     myphoneNumberNode := dropMessageNode.getNode(myPhoneNumber)
     //remove node
    myphoneNumberNode.removeNode(sourceNumber)
END
```

6.6. Track me Component

```
sendLocation(peerPhoneNumber)
BEGIN
REPEAT every 5 seconds
BEGIN
            currLocation := getlocation(myPhoneNumber)
            database := Firebasedatabase()
            Trackmetable := database.getNode("TrackmeTable")

         //update location of user in table for peer
           IF Trackmetable.getNode(peerPhoneNumber) is NULL THEN
               Trackmetable.addNode(peerPhoneNumber)
               peerNodeReference :=
Trackmetable.getNode(peerPhoneNumber)
               peerNodeReference.add(myPhoneNumber,currlocation)
       END
END
```

7. Development plan

	A 1	A 2	A 3	A 4	A 5	A 6	A 7	A 8	A 9	A 10	A 11	A 12	A 13	A 14	A 15	A 16	A 17	A 18	A 19	A 20	A 21	A 22
Aakash	*				*	*				*						*	*	*		*		
Kartikeya	*	*	*					*	*				*						*	*		*
Manpreet	*				*		*				*			*	*		*			*	*	
Shubham	*	*		*				*			*	*		*						*	*	*

Activities:

A1 - UI Discussion & Design

A2 - Design & Documentation of Authentication Component

A3 - Development of Authentication Component

A4 - Testing of Authentication Component

A5 - Design & Documentation of Map Rendering Component

A6 - Development of Map Rendering Component

A7 - Testing of Map Rendering Component

A8 - Design & Documentation of Instant Messaging Component

A9 - Development of Instant Messaging Component

A10 - Testing of Instant Messaging Component

A11 - Design & Documentation of Peer Management Component

A12 - Development of Peer Management Component

A13 - Testing of Peer Management Component

A14 - Design & Documentation of Drop Message Component

A15 - Development of Drop Message Component

A16 - Testing of Drop Message Component

A17 - Design & Documentation of Track me Component

A18 - Development of Track me Component

A19 - Testing of Track me Component

A20 - Integration Testing

A21 - System testing

A22 - Product Documentation

8. Documentation of abstract APIs

Attached as a separate document.

9. Comments on design decisions

Major design decision taken so far has been to use Firebase platform for application development. This decision was taken considering diverse range of functionalities provided by the Firebase. One of the most important feature of the Firebase platform is its real time database that served as a motivational factor for using it for our chat application. The authentication service provided by the platform is used for login in the application. The decision was backed up by rising popularity of Firebase platform and ease of usage of its functionalities

10. Discussion

Not required for this document.

Attachments:-

S.No.	Document Type	Attached Document
1	Abstract API document for Whereabouts by MASK Inc.	SSG562_G02_AbstractAPI.PDF