5th April 2017

Functional Specification

Department of Computing and Networking

Software Development Degree

Project name:	MaaP (Message as a Platform)
Student:	Chihabeddine Ahmed
Student Number:	C00210496
Supervisor:	Joseph Kehoe

Table of Contents

1. Introduction

Specifying the requirements for a completely new product to be developed is not an easy task. It often contains unnecessary features which at first seem to be useful, while lacking simple things that users would appreciate. The purpose of this document is to give an overview of the expected functionality of this application at the beginning of each iteration. It will also detail any non-functional requirements that must be implemented in conjunction with this functionality and to what extent the functionality specified was altered during implementation.
The implementation of this functionality will be described in greater technical detail in the Design Document of this project, the technologies used will be discussed in detail in the Research Document and the results of this project will be discussed in detail in the Final Report.

This document is divided into nine sections, the first section will provide an introduction to the project and that includes the description of the project and will give an overview of the proposed system that includes, the overview of architecture design, technologies the system in which to give the reader enough information to understand the rest of this document. The third and the fourth section will deal will give use cases as in this section I will give an overview of the system use case and then break that down into smaller uses cases for each service individually and explain each of the uses cases. In the Fifth and the Sixth section, I will explain the targeted user of the system also explain the design of the User Interface. The last three sections will explain the supplementary specifications, the plan for the project and give an overview of the User Interface prototype.

1.1 Project Description

MaaP (Messaging as a platform) is a self-hosted messaging application built for the android platform. The front-end of the application build using Java and the backend is built purely on Node.js. The application will allow the user to:
Authenticate, i.e. Login, Register, and Logout of the system.
Allow the user to send and receive messages to a particular user.
Allow a user to search for another user then send him an invitation and upon the invitation acceptance, they will be able to exchange messages instantly.
The goal of this project to build the application for many types of users such Collages, companies etc. where these users can host the application on their own servers and own the data. Each of the functionality is supposed to be built in a micro-services style, where each service is dependent.

2. Overview

2.1 Technology Overview

As discussed in detail in the Research Document of this project, the principal technologies used in this project are:

- **Java & XML** (Extensible Markup Language): Are used for frontend development.
- **Node**: a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine) Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices
- **Mongo DB:** A non-relational database technology used for storing MaaP information.
- **FCM**: Firebase cloud messaging, owned by google and its technology that allow the application to send notification between devices.

2.2. Architecture Overview (Context Diagram)

The architecture of this project is summarized in the context diagram (figure 1) below with an Android used as the main client side. The diagram below represents a high level of how the system will work. The system will consist of a mobile app (Android) as a frontend for interacting with the backend (Node services). Mongo dB, a non-relational database technology used for storing MaaP information. FCM (Firebase), allowing notification to be sent from device to device.

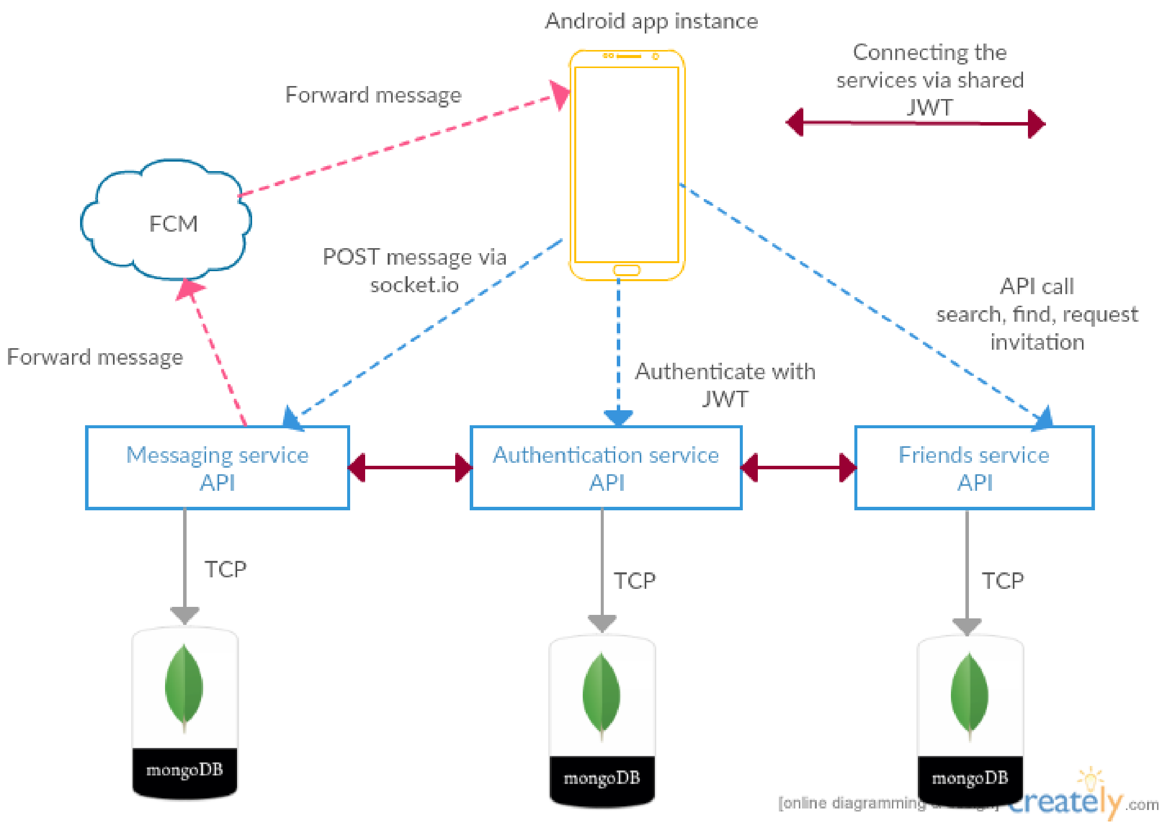

Android app instance
Connecting the services via shared JWT
Forward message
FCM
POST message via socket.io
API call search, find, request invitation
Forward message
Authenticate with JWT
Messaging service API
Authentication service API
Friends service API
TCP
TCP
TCP
mongoDB
mongoDB
mongoDB
[online diagramming & design] creately.com

Figure 1: system architecture overvie

2.3. Use case overview

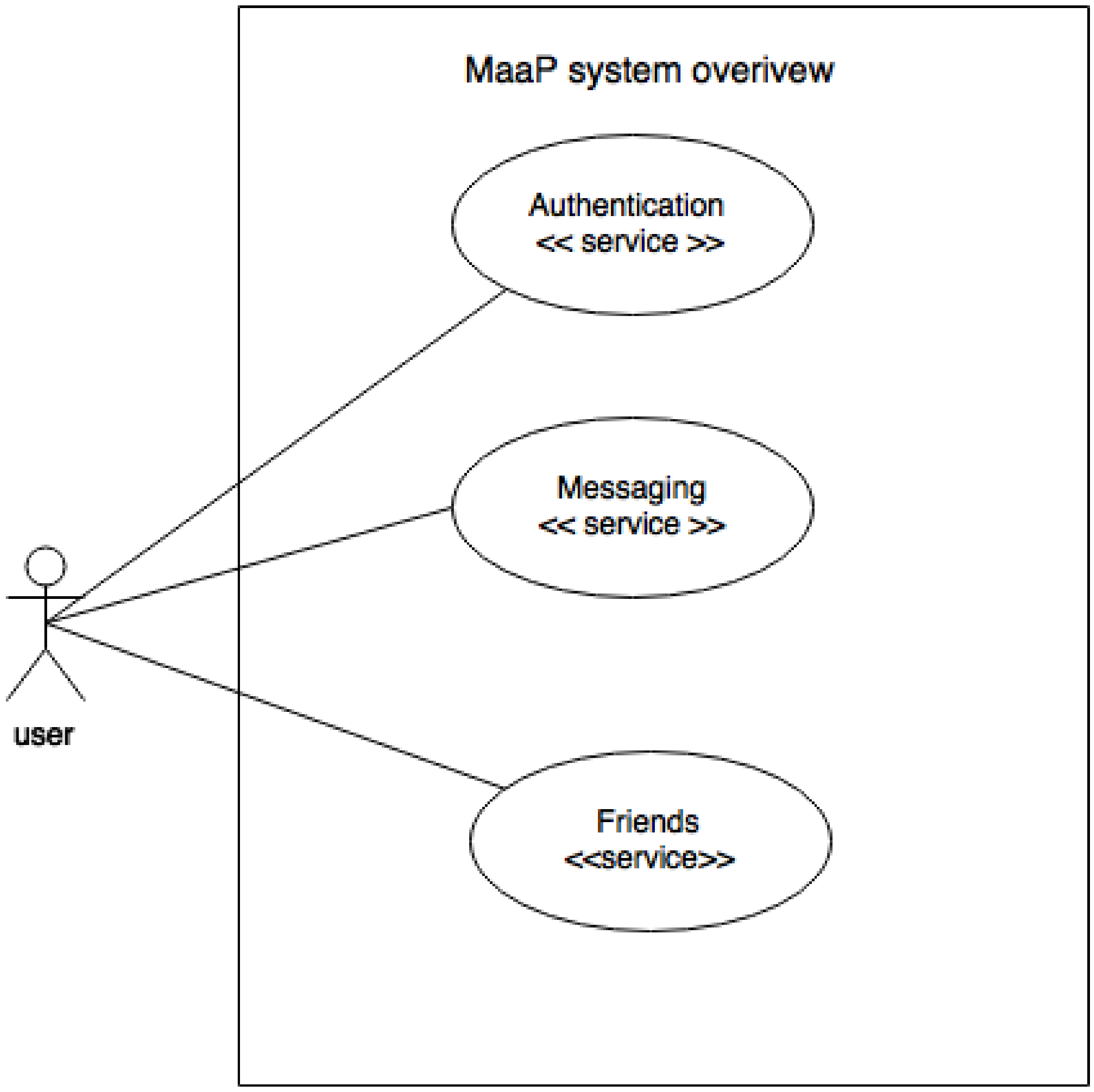


Figure: 2

3. Brief Use Cases

3.1.1 Authentication

A user authenticates against an identity provider to establish a certificate that can subsequently be used to identify the user. The authentication process will allow the user to Signup, Login, and logout from the system. New users, they have to register to the system. This can be done by email/password. Old users can login to the system without the need of registering. To register, the system will display an input mechanism that will allow you to enter username/email phone number and password. Then old user uses these credentials [phone number, password] to login to the system. The user then can logout of the system upon pressing the logout button.

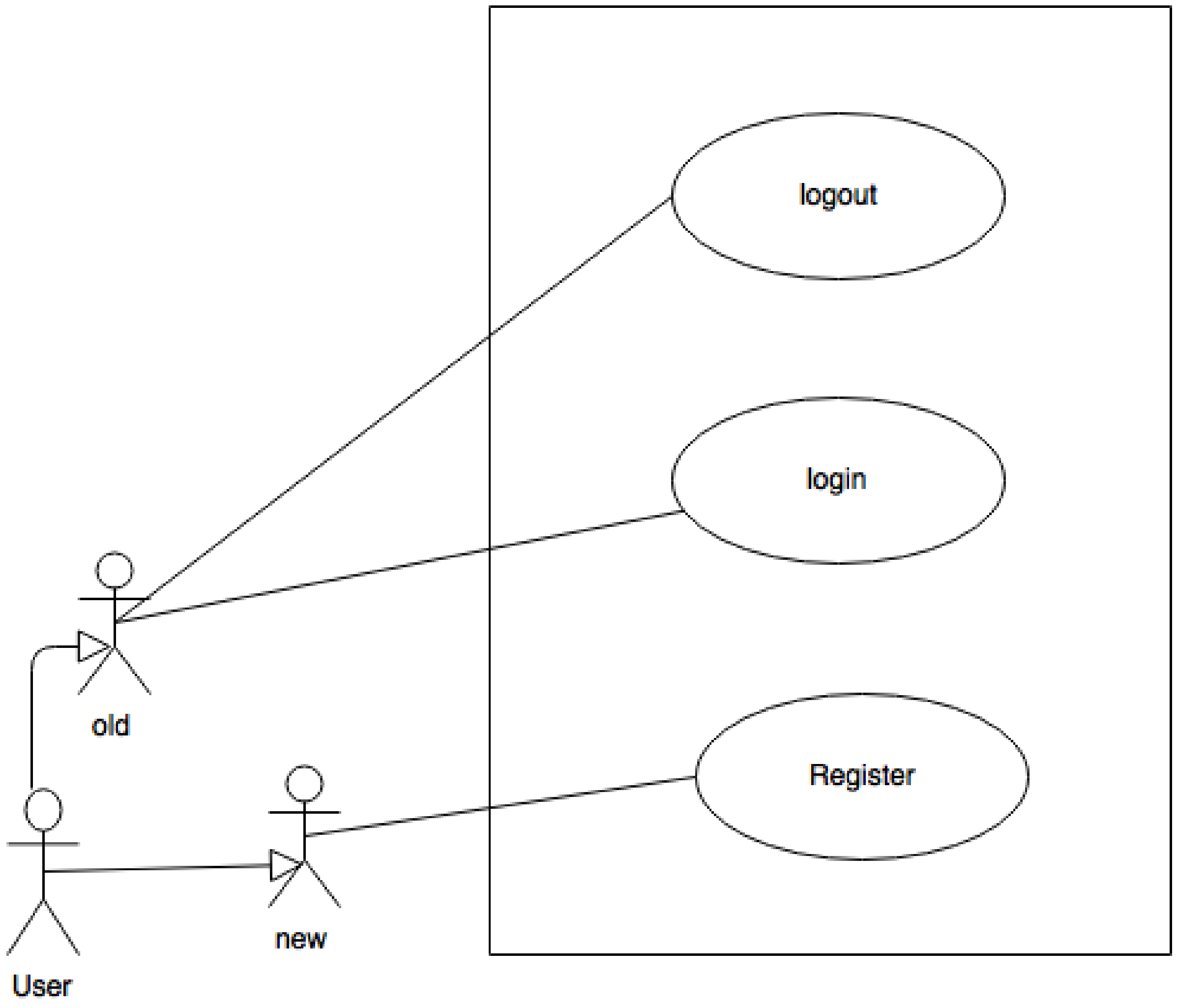


Figure: 3

3.1.1.1 Register

Informal Description

Once the app has been downloaded by the user, they will be able to register their credentials to gain access to the app. The credentials which must be provided are (phone number, username, email address, and password). The user account will be created if the provided credentials are valid and not already in use by another user.

Casual Use Case

Actors: New User

Main Success Scenario: This use case begins when a user wishes to create an account. The user will select the "Register" functionality. The system will prompt the user for a username, phone number, email and a password to be entered twice. The user will enter all details just once and a password twice. The system will verify this information with Mongo database. The system will create a new user in the database. The system will notify "registration successful ", the user that their account has been created. The system will return the user to the "login" screen.

Alternatives: Alternatives: The system will display an error message if the username or password to not match specific format criteria if the two passwords entered do not match or if the username entered is already the name of a registered user in the database.

Non-Functional Requirements

1. This use case should take no longer than 20 seconds for a user with elementary experience, assuming a stable internet connection.
2. This use case should be successful upon the user's first attempt 90% of the time, assuming a stable internet connection.

3.1.1.2 Login

Informal Description:

Informal Description When a user opens the application for the first time on a new device they will be presented with the "login" screen. Assuming they have an existing account they will need to enter their username and password to continue to the main application.

Casual Use Case

Actors: Old User

Main Success Scenario: This use case begins when a user wishes to log in. The system will prompt the user for their phone number and password. The user will enter their username and phone number. The system will verify this information against the database. The system will prompt success message. The system will display the user's main screen.

Alternatives: If the phone and/or password entered were invalid the system will notify the user and not move from the login screen.

Non-Functional Requirements

1. This use case should take no longer than 10 seconds for a user with elementary experience, assuming a stable internet connection.
2. This use case should be successful upon the user's first attempt 95% of the time, assuming a stable internet connection.

3.1.1.3 Logout

Informal Description

The user will be able to logout from the "Main interface" menu. This will clear all the application's locally stored data for that user and will be directed to the login screen.

Brief Use Case

Actors: Old User

Description: This use case begins when a user wishes to log out. The user will select the "Logout" functionality. The system will log the user out and display the login screen.

Non-Functional Requirements

1. This use case should take no longer than 5 seconds for a user with elementary experience.
2. This use case should be successful upon the user’s first attempt 95% of the time.

3.1.2 Messaging

The messaging service is one of the core components of the system. The user will have the ability to send different messages such, text, video and images to different users. The user the selects a message type they wish to send to another user (the recipient). The user should the ability to view the message sent from a user via the push notification.

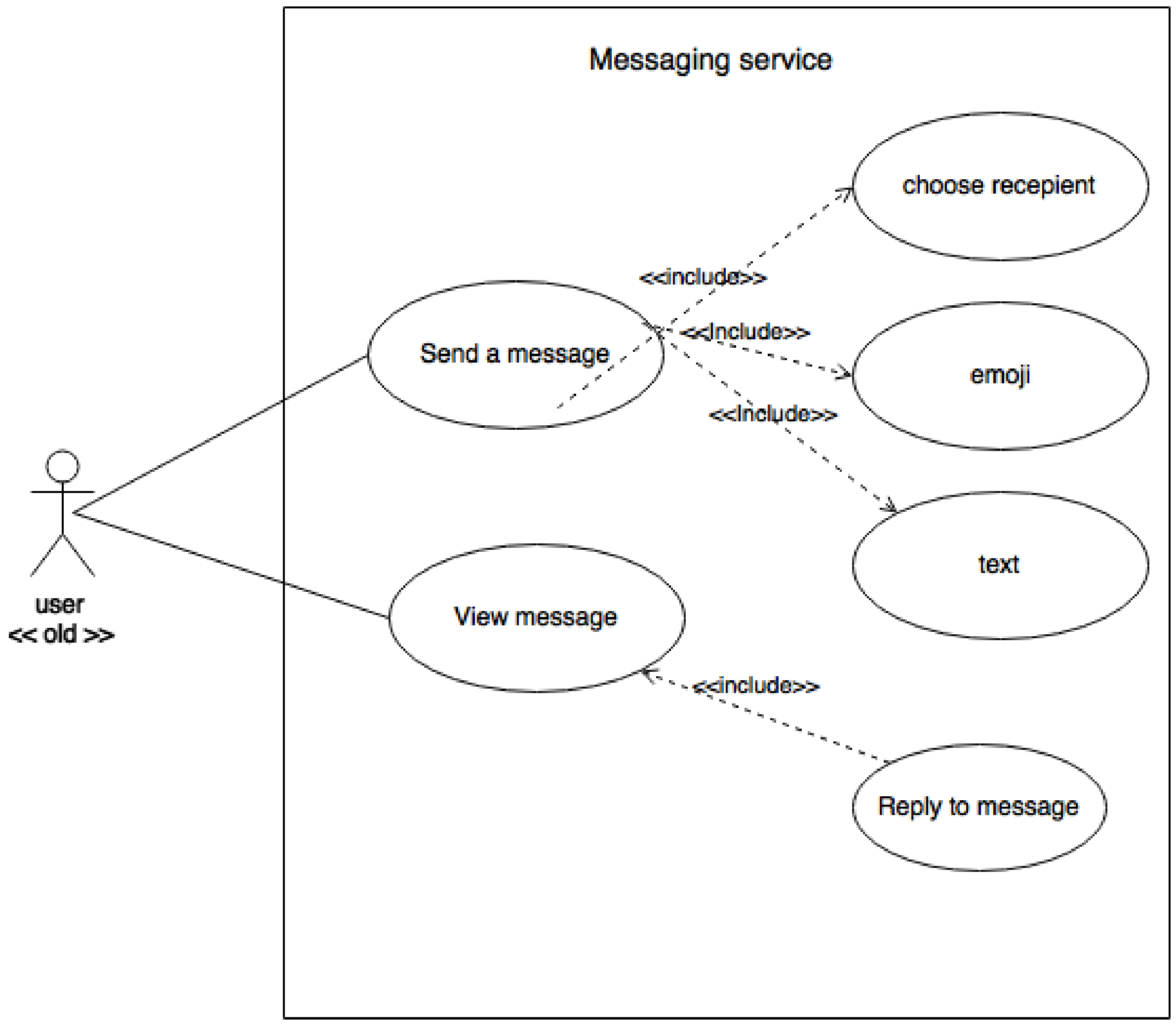


Figure: 4

1.2.1 Send message

Informal Description

The user will wish to send a message to a user from the contact list. All messages are located in an activity called "Messages". In this project, the user can send text message and Emoji but sounds and videos are excluded in the development of this application but in the future I will consider video and audio and image messaging.

Brief Use Case

Actors: User

Main Success Scenario: This use case begins when a user wishes to send a message type to a user. The user selects a contact they wish to send a message to from the contact list. The user will be directed to a new activity called (messaging activity). Messaging processing will take place in the messaging activity.
The system displays a list of options that are available for the message type (Text, emoji). The user selects the desired message type. The user presses the "send" function indicated by send icon.
Messages are sent successfully.
Use case terminates here.

Alternatives: If messages are not sent the system display an error "message not sent ".

Non-Functional Requirements

- Messages should be sent instantly.

3.1.2.2 Receive message

Informal Description

Users can receive messages in two ways. First if the user a logged in and if they are in the messaging activity, they can see all messages. Second, if they are not logged in they will receive messages via push notification.

Brief Use Case

Actors: User

Main success scenario the use case begins when the user wishes to view the sent message. If the user is logged in they will see the message sent instantly. If the user is not logged in they will receive the message via FCM (Firebase cloud messaging) push notification.

Alternatives: None

3.1.2.3 View message

Informal Description

Users can view a message in two ways:

1. If the user is logged in they can view messages from the "Messages" activity"
2. If they are not logged in they can view the messages from the notification drawer on their phone.

Brief Use Case

Actors: User

Main success scenario the use case begins when the user wishes to view the sent message. The user can navigate to the "Friend "activity. The user can select a friend. The user will be navigated to the "messages" activity where there they can view messages from the selected user.

Alternatives: None

3.3.1 Friending

This service allows the user to invite, add, and reject a user request and search for a user.

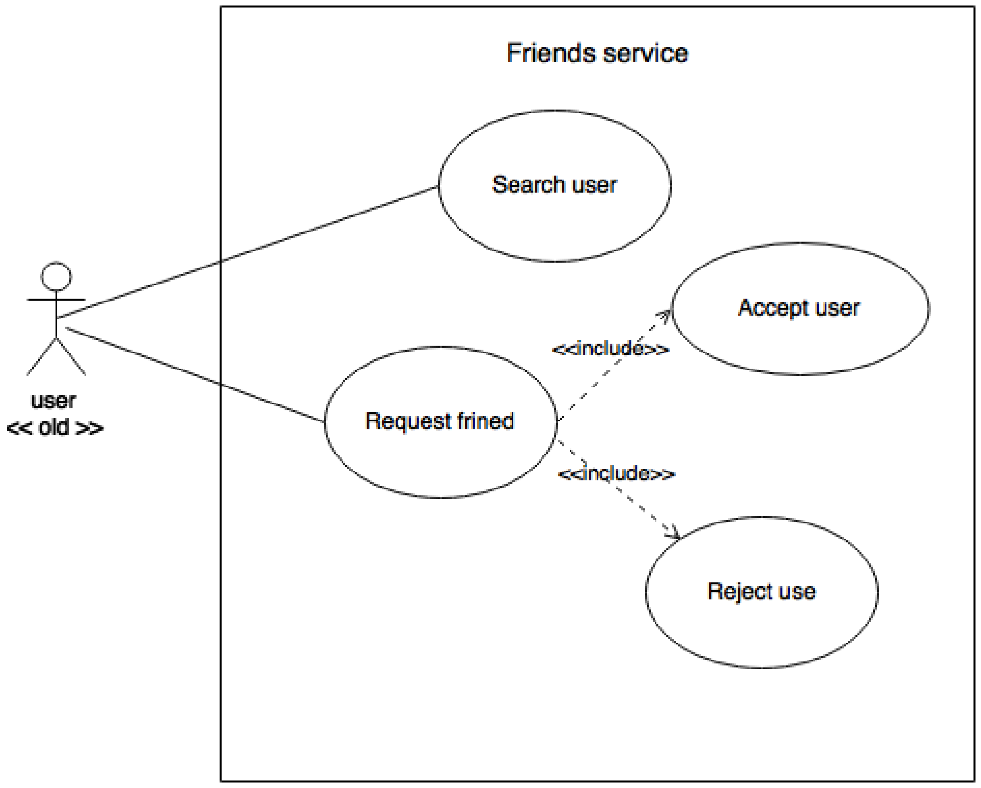


Figure: 5

3.1.3.1 Search friend

Informal Description

Users will be able to search for a user from the database using the search box on the top of friend activity. Users can type a phone number of the person that they wish to search for. Once the user types the phone and hit the search icon a search result will be displayed upon on the availability of the user.

Brief Use Case

Actors: User

Main success scenario the use case begins when the user wishes to search for a user. The type the phone number of that particular user located on the top of the "friend "activity. The user presses the search icon. The system will display the name of the searched user.

Alternatives: the system will display “user not found” error if the user is not found in the database.

3.1.3.2 Friend request

Informal Description

From the search result, the user can send a friend to request to a particular user. That particular user can then accept or reject the user request.

Brief Use Case

Actors: User

<u>Main success scenario</u> the use case begins when the user wishes to send a friend request. The select a user from the search result that they wish to send a request to. The user press sends an invitation. The server sends friendship invitation. Friend user accepts friendship invitation. Friend user does not accept friendship invitation. The server receives the accepting message and connects the two users Server receives the not accept a message and the use case ends.
The server sends a message to the user telling that the friend has accepted the friendship invitation.

<u>Alternatives:</u> User reject invitation and the use case ends.

4. Backend (Node.js)

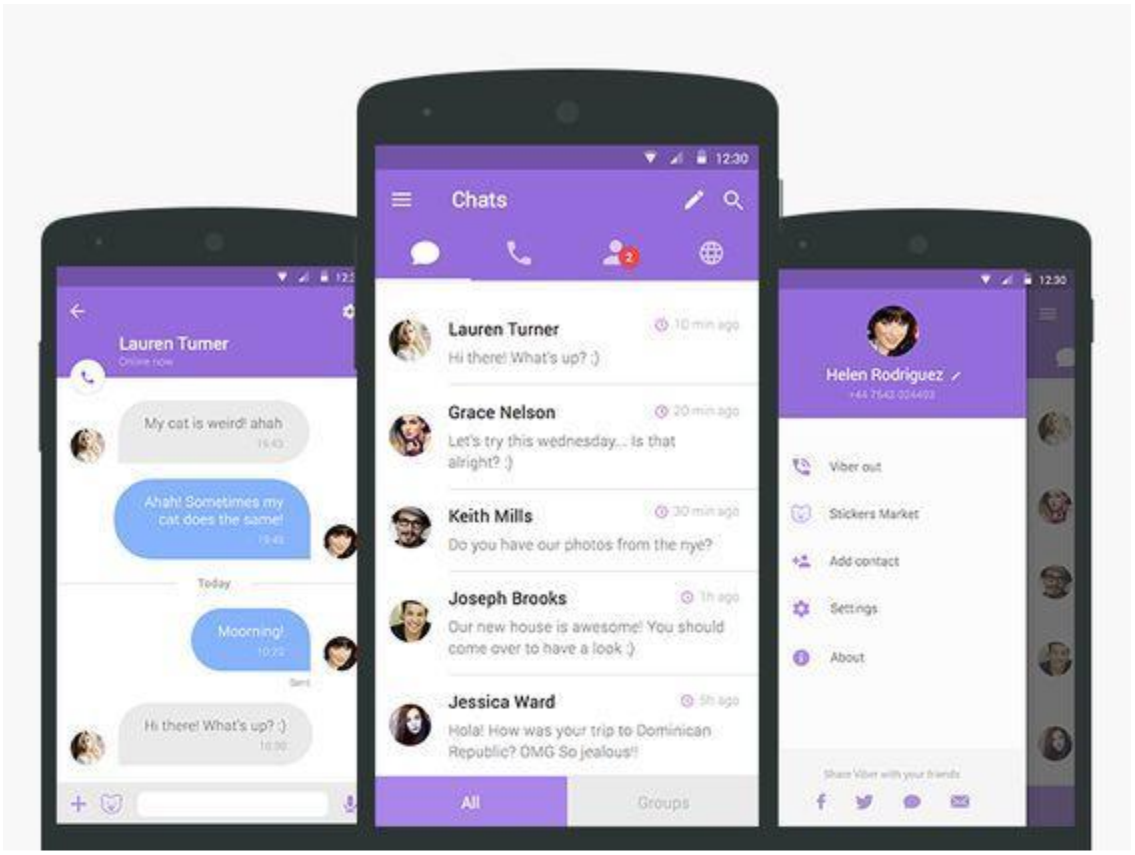
The mobile application will need to be able to communicate with the backend services that are implemented in node.js. Since the data will be unstructured data, a Mongo database will be used to store the user and item information. The passing of data from the mobile application to the backend will be done in a JSON and HTTP methods over the networks. The JSON objects can then be parsed to extract the information. FCM (Freebase cloud messaging) is used with the backend to allow the user to send notifications from device to another.

5. Target Users

The application will appeal to any android phone user interested in send and receive instant messaging. The application will open sourced, companies will able to get the source code from Github and they will able to modify it and run the application on their own infrastructure, allowing them to have a complete control on the data of the application. Companies can integrate other services for example. A College that may have an idea of a service that allows teacher and students to send and receive messages but at the same time, the college wants to own the data of the application. Well, this application is here at the right place at the right time.

6. User Interface Design

The mobile application's user interface will be designed with the aim of it being simple and as concise as possible to provide a good user experience. If the user interface is too difficult to navigate or understand, the user will not want to use the application due to bad user experience. It is important that the user will be able to operate the application immediately without the need for a user guide. The color scheme for the application will vary from activity but as it has been tested by many users the result was impressive. The user interface must also be responsive and dynamic. The user interface design was adored by Viber. (https://www.viber.com/en/). To see the User Interface design please refer to section (7) of this document.



Lauren Tumer
Online now
My cat is weird! ahah 19:43
Ahah! Sometimes my cat does the same! 19:45
Today
Moorning! 10:22
Sent
Hi there! What's up? :) 10:30

Chats
Lauren Turner — 10 min ago — Hi there! What's up? :)
Grace Nelson — 20 min ago — Let's try this wednesday... Is that alright? :)
Keith Mills — 30 min ago — Do you have our photos from the nye?
Joseph Brooks — 1h ago — Our new house is awesome! You should come over to have a look :)
Jessica Ward — 5h ago — Hola! How was your trip to Dominican Republic? OMG So jealous!!
All Groups

Helen Rodriguez
+44 7542 024403
Share Viber with your friends

Figure: 6
https://www.viber.com/en/

7. User Interface Prototype

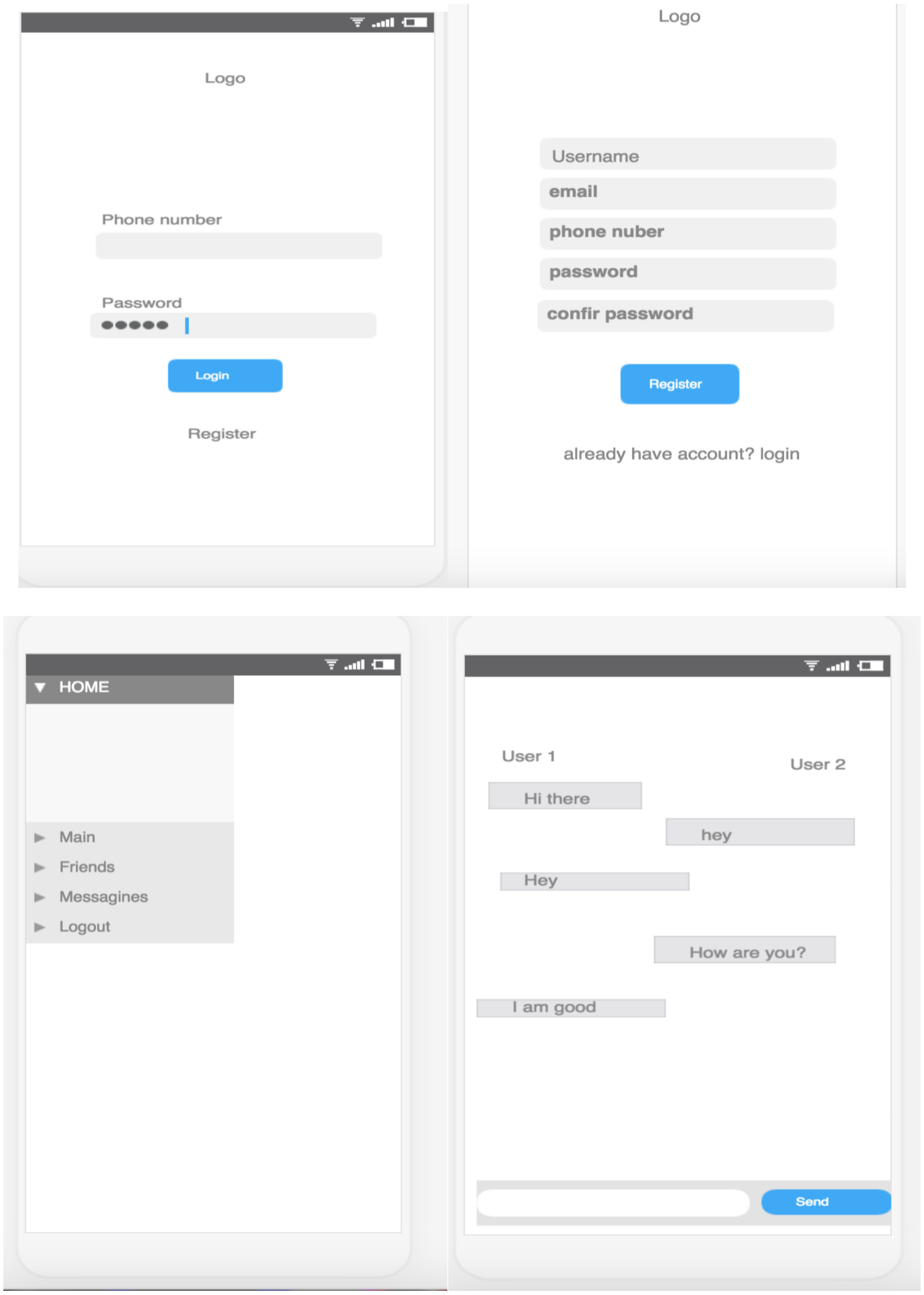


Figure: 7

8. Supplementary Specification

8.1 Functionality

- Users must be registered and logged in to use the application.
- Data can only be sent and received to and from the server if there is an active internet connection.
- Users should be able to send different type of messages to different services.
- User should own all data.

8.2 Usability

- The application must be easy to navigate through the menus.
- User friendly to users of all ages, simple options, no ambiguity.
- Consistent look and feel to the mobile application.
- Informative feedback to the user in the case of events or errors.

8.3 Reliability

- The user is responsible for managing and backing up his own the data.

8.4 Performance

-
- Activities transitions must take less than two seconds 80% of the time.
- Data retrieval must take less than 3 seconds 70% of the time.

8.5 Supportability

- The initial application will be design for Android platform only.
- User credentials must be kept private and secure.

9. Project Plan

The duration of the project from commencement date to submission date is 29 weeks. In this time frame, there will be three Iterations, with each iteration lasting approximately 9 weeks more or less. For each of this iteration, I plan on dividing my time up in segments as 1.5 weeks for research. 2.5 weeks for design 4 weeks for coding 1 week for testing.

9.1 Iteration 1

This iteration began on the 5th of October 2016 and concluded on the 15th of December 2016. Prior to this iteration and halfway to this iteration I have completed my (research document) so that I know what technology that I am going to use.

Scheduled work

- **Design the authentication API**
  Authentication plays a big part in this project, I will design the authentication API for my system. The authentication will cover user (registration, login, and logout).

- **Implement authentication interface**
  To test the authentication API, I need to create a login, register, logout interface with welcome screen upon successful registration. This user interface will be replaced with a well-designed one at a later stage and therefore it should not be treated as a high priority task for this iteration.

Changes in requirements

At the end of this iteration, I found myself spending more time developing a complex authentication API. So the following were completed in this iteration.
I have completed the authentication API and I kept it as simple as possible and I have also created a simple user interface to test my API, at the end of this iteration I found myself up to date with scheduled work.

9.2 Iteration 2

This iteration began on the 20th of October 2016 and concluded on the 13th of Feb 2017.

Scheduled work

- **Design the messaging Service**

  As a messaging application the messaging function is a must. In this part, I will design and plan a way of developing messaging API or a function that I can add to the authentication API and let them work together and provide the messaging functionality.

- **Implement messaging interface**

  To test the messaging function, I need to create a simple messaging interface that would allow two users to send and receive messages using the backend API.

Changes in requirements

At the end of the iteration, I managed to get some part of the messaging function backend and front end to work such, connecting users using socket.io and store that data in the database. The only problem with this there is no restriction on who should I send a message to. For instance, if I send a message everyone who is logged in can see it. I needed some kind of mechanism that would allow me to restrict message. So I came up with the idea of adding the Friend list feature. I did my planning for this and I tackled this in the last iteration. So at the end of the project, I found myself a bit short in term of timing but in the third iteration I will finish the rest of the project in terms of coding.

9.3 Iteration 3

This iteration began on the 13th of October 2016 and concluded on the 5th of March 2017.

Scheduled work

- **Complete the rest of the project**
  From iteration 1 and 2 I have completed most parts of the project such authentication, messaging and I did not finish the messaging function of the app. So part of the plan in this iteration is to finish the messaging function of the app.
  The rest of messaging function is to create friends that a user can add to his friend list and from there they can exchange messages with them.
- **Update project Documentation**
  The final changes need be added to the project documentation and the final report must be completed.
- **Testing**
  As a part of the software, development testing is one of the most important factors that determines the outcome of your application. I am hoping to do more testing if I get more time.

Changes in requirements

At the end of this iteration, I found myself under pressure in term of finishing the rest of the project and updating the documentations. But in term of implementation, I thought I got most the core features of the system that I have planned is completed. But extra features like the Integration Service is not tackled due to the time.