Software Engineering Lab 2020

Slabber

(Chat Application)

Software Requirements Specification

Version 1.0.0

21-02-2020

Pranjal Somkuwar-180001036
Vinesh Katewa-180001061
Ayush Agrawal-180001011
Harsh Chaurasia-180001018

Prepared for

CS 258 Software Engineering

Spring 2019

Revision History

Date	Version	Comments
21-02-2020	1.0.0	This is the first model of System Requirements.

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Approving Authority	Title	Date

Table of Contents

1. Introduction

This project is intended to create an application that will allow people to create accounts and through their accounts users can have conversations with various other users. This app will allow the users to communicate with each other and start conversations with new users if they exist. These users can use this app at any time of the day and it will be available to the users 24X7.

2. Overview

2.1. Purpose

- The purpose of this project is to implement a real time “chatting” application. The chat app will provide the users the ability to sign up for an account and log in into that account whenever they want to. This app will run on a fully functional NodeJS server in the backend. The logged in users can see their list of ongoing conversation on the app’s homepage and click on any conversation to start “Chatting”.
- The user will have an option to create a chat room and add other users into that chat room to start a group conversation with everyone.
- The user will be able to create an account by providing their email id, the software will authenticate that email id and on successful authentication the software creates the user’s account. The user can then use this account to chat with his/her friends in private conversations or create chat rooms and have group conversations with everyone.
- The development of this project is centered on the development of a message protocol that would allow the users to properly log in users, send messages and perform system maintenance.

2.2. Intended audience and reading suggestions

This project is a prototype of a chat application that allows multicast chatting and it is restricted within the college premises. This has been implemented under the guidance of college professors and teaching assistants. This app is useful for the conversation between the college residents.

2.3. Project Scope

- The purpose of the online chatting app is to create a convenient and easy way to chat with other users. It also simulates multicast chatting. The users will be able to chat with multiple users at the same time in a chat room.
- Every chat or data will be secure with end to end encryption. We will try to provide a bot which you can talk to ,ask questions and much more. Above all , we hope to provide users with a comfortable experience .

2.4. Project Perspective

- The system to be developed here is a Chat facility. It is a centralized system. It is a Client-Server system with centralized database server. All local clients are connected to the centralized server via LAN.
- There is a two way communication between different clients and servers.
- This chat application can be used for group discussion. It allows users to find other logged in users and start private conversation or group conversations.
  The users can find each other with their email ids if they are not connected yet. These connections will then be stored under a ‘My Contacts’ Tab.
- Can work in areas with just a Local Area Network and does not need a proper internet connection. This system is useful when nearby users feel to chat among themselves (eg.in a conference hall). The app will provide a LAN mode where a user can start a server from their mobile phone over the LAN network and every other users connected to that same network can use the app on LAN mode to start a conversation with each other.

2.5. References

Google and TA support

3. Overall Description

3.1. Definitions

- **DB (Database)** : Data architecture describes the way data is collected, stored, accessed, and used in companies and organizations. It can be seen as the roadmap for how data flows across an organization's IT systems and applications. Database provides these functionalities
- **Framework** : It is a pre-written set of code that contains defined open (unimplemented) set of functions that provide generic functionality and can be tailored by additional user-written code, thus providing application-specific software, and also easing the developer from writing all the code from scratch.
- **UI (User-Interface)** : All the parts of a website, app, computer, smartphone, etc. that the user can manipulate and interact with.
- **Front-End** : It is responsible for the interface running and operating. (opposed to the visual looks defined by UI.)
- **UX (User Experience)** : UX describes the emotions, attitudes, and ease-of-use a person has when using a product or service.UX Design is the practice of using design to improve communication between a product and its user in order to enhance the user's overall experience.
- **Back-end** : Back end refers to the "under the hood" part of a website or web service that makes it run (this includes applications, web servers, and databases), and is typically not visible to the user interacting with the site or service.

- **Server** : Web servers are computers used to store websites, online apps, documents, pictures, or other data, and can be accessed through the internet by way of applications like web browsers or file transfer protocol (FTP) clients. When you visit a website with the browser on your computer or smartphone, you are requesting it from a web server.
- **API** : An API or Application Programming Interface is the interface used for building web applications. APIs serve as the communicator between the front-end and back-end (server).
- **Bugs** : Bugs are coding mistakes or unwanted pieces of code that keep a website or program from working properly.
- **Schema -**The database schema of a database is its structure described in a formal language supported by the database management system (DBMS)

3.2. Product features

- This app will provide instant chat service to the Users. The logged in users will be able to use this app over the internet to chat with their friends and family or change to LAN version to chat with other users connected on the same network.
- Users can send text messages to private conversation or group conversation. In addition to text messages users will also be able to send images and set their own image as their profile picture for their friends to see.
- Businesses usually deal with a large number of people and need to keep their conversations private. So instead of handling a large amount of private conversations on their main account, this app provides the users with bots which can be created by business and other users to deal with large amounts of private conversations.

3.3. Operating Environment

- A Centralised Database (hosted at mongodb.com) will be used.
- Operating System : Android, iOS

- Database Type : NoSQL.

3.4. Design and implementation constraints

Works only on the android version greater than 5.
Users must have an email id.

4. System Features

4.1. Functional Requirements

4.1.1. Registration

- As soon as the user installs and opens the application, if the user has not already registered an account, he/she will need to create an account to use this application.
- For registering, the user will need to provide the application with a set of information which are necessary for creating an account. These informations include First name, Last name, email id, a Unique Username, Password, Date of Birth and a set of optional informations like address of the user etc.
- The user's email id will be authenticated by the application and the user's account will be registered successfully. The user will be able to login and use the app now.

4.1.2. Login

- The users will be able to login to their account by the email/username and password that the user has registered

the application with. Email cannot be changed once registered but passwords can be changed on a regular basis.
- Once logged in the user can utilise the features of the application to the fullest.

4.1.3. Starting a conversation

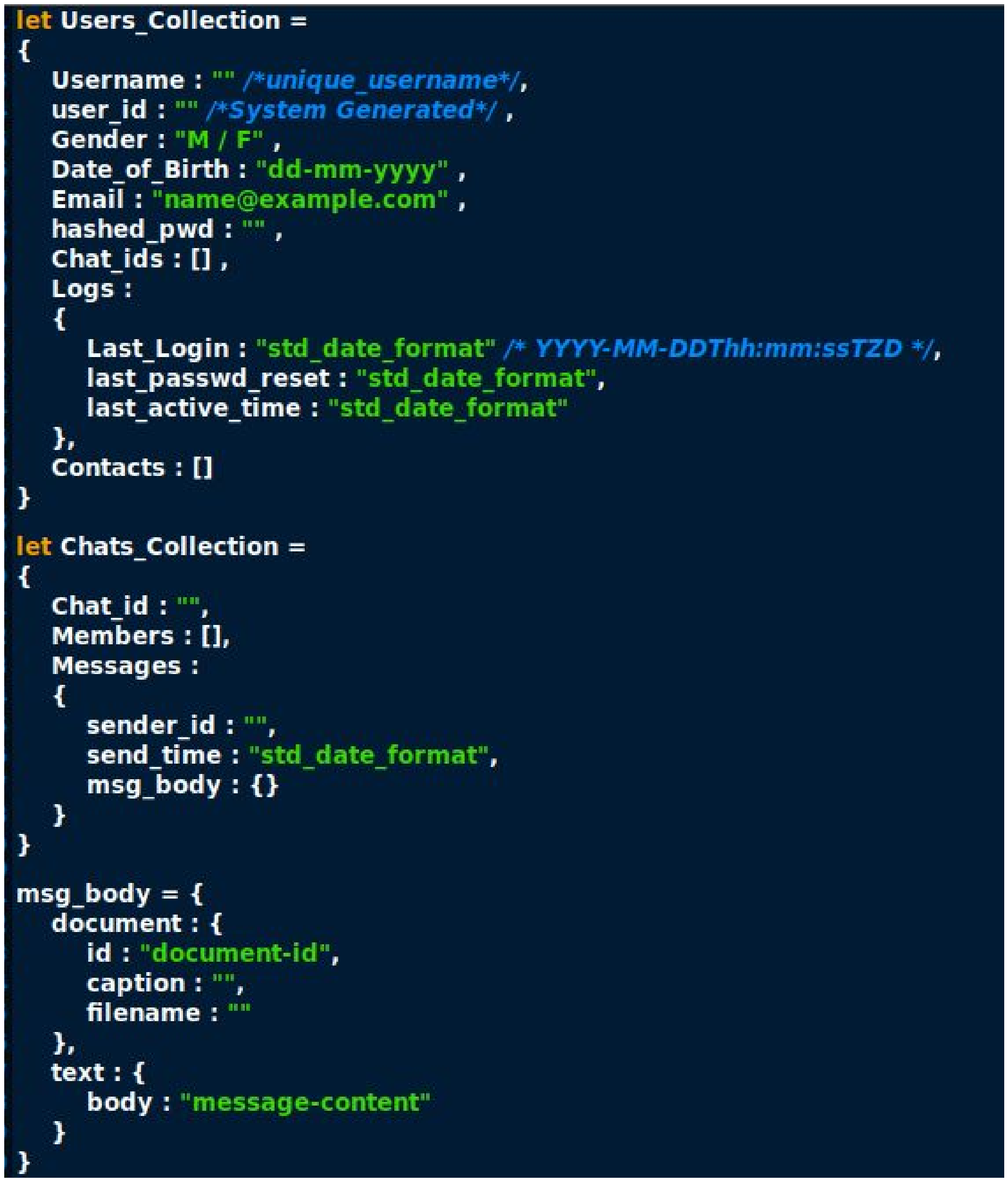
- Any user will have two ways to start a conversation. First, the user can search for his friend by name and the app will provide a list of all users with that name and the user can select the one he/she wants to talk with.
- Another way the user can start a conversation is that each user has a specific unique user ID which can be shown in the user's profile page. The user can ask his friend to share this user ID and they can connect using this and start their conversation.
- Similarly, for creating the chat rooms, the user can add multiple users in the chat room after the user has assigned a name to it. More users can be added to the chat room at any point of time whenever the chat room's creator wishes to do so.

4.1.4. Chatting in an already open conversation

- When the user opens the apps he will be shown a list of already open chat conversations, the user can select any one of them and he will be directed to that particular conversation. Once clicked the user can now send messages in that conversation.

4.2 Non-functional Requirements

4.2.1. Schema :



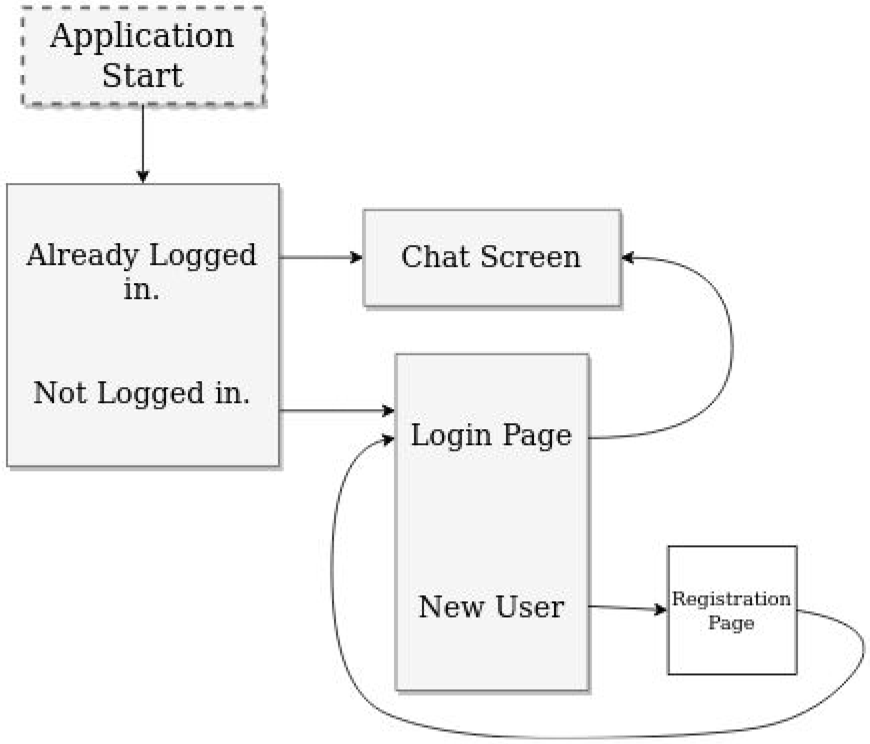
```
let Users_Collection =
{
   Username : "" /*unique_username*/,
   user_id : "" /*System Generated*/ ,
   Gender : "M / F" ,
   Date_of_Birth : "dd-mm-yyyy" ,
   Email : "name@example.com" ,
   hashed_pwd : "" ,
   Chat_ids : [] ,
   Logs :
   {
      Last_Login : "std_date_format" /* YYYY-MM-DDThh:mm:ssTZD */,
      last_passwd_reset : "std_date_format",
      last_active_time : "std_date_format"
   },
   Contacts : []
}

let Chats_Collection =
{
   Chat_id : "",
   Members : [],
   Messages :
   {
      sender_id : "",
      send_time : "std_date_format",
      msg_body : {}
   }
}

msg_body = {
   document : {
      id : "document-id",
      caption : "",
      filename : ""
   },
   text : {
      body : "message-content"
   }
}
```

- Collection of users will contain will contain all personal info of the user, eg. name, date of birth etc. It will be stored in MongoDB database which is a NoSQL database.

4.2.2. Data Flow Diagram



When the user opens the app for the first time, he is redirected to the login screen. Either he can login or register for a new account.

5. External Interface Requirements

5.1. User Interfaces

- **Front-end software**: **React Native** is used to code the front end of the app. React Native is an open-source mobile application framework created by Facebook. It is used to develop applications for Android, iOS, Web and UWP by enabling developers to use React along with native platform capabilities. It has many advantages over other application framework such as :
    1. **Cross Platform Compatibility**: Most of the React Native APIs are cross platforms, and it means you one component will work on both platform iOS and Android as well. You can develop complete, full-blown applications that look, run and feel native without writing a single line of platform-specific code.
    2. **Open Source**: React Native has a very large community of programmers working constantly towards bug fixing, feature improvement and helping people to use it at ease.
    3. **Shorter Development Cycle:** React Native changes the life cycle of the mobile app development apps. A huge number of the community associate with it due to open-source. It has a lot of mechanisms available for use.

- **Back-end software: MongoDB, Express and Nodejs**
  Node.js is popularly being used in web applications because it lets the application run while it is fetching data from the backend server. It is asynchronous, event-driven and helps to build scalable web applications. Even though Node.js works well with MySQL database, the perfect combination is a NoSQL like MongoDB wherein the data in database is stored in JSON format which stands for JavaScript Object Notation. Hence, it provides a better compatibility with NodeJS which runs JavaScript code. MongoDB represents the data as a collection of documents rather than tables related by foreign keys. This makes it possible for the varied types

of data dealt over the internet to be stored decently and accessed in the web applications using Node.js.

5.2. Hardware Interfaces

It will work on all devices which have Android 5.1.1 or above operating system (namely Lollipop, Marshmello, Nougat, Oreo, Pie, Android 10) as well as all the iOS devices.

5.3. Software Interfaces

Following software used for the chat application

Software used	**Description**
Database	MongoDB is chosen for its flexibility and easy scalability. It serves as an online storage for chats and user-details.
Server-side Back-end	ExpressJS framework is used because it is simple and fast and also the most popular node framework for server-side applications.
GUI	React Native, which is used extensively to build cross platform mobile applications with javascript.
Operating System	Android as well as iOS

5.4. Constraints

- The App needs to be connected to a network (either the internet or a local network) in order to communicate.
- The App may not work properly on devices with Android version below 5.1