Software Engineering Lab 2020

Slabber

(Chat Application)

Software Requirements Specification

Version 1.5.0

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.
28-05-2020	1.5.0	This is the updated model of System Requirements with change in UI Framework.

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 signup with their emails and through their accounts users can have conversations with various other users. This app will allow the users to communicate with each other and have conversations with other registered users. These users can use this app at any time of the day and it will be available to the users 24X7. Conversations will include text messages, image sharing, stickers, and other multimedia stuff. Also, on changing devices, any users conversations will be preserved on an online backup Database.

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 conversations on the app’s homepage and click on any conversation to start “Chatting”.
- A user can add any other user to their friend list by sending them a request. A user must know the username of the person he/she wishes to send a friend request.
- 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 log into the app and 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. It is intended for groups of people who want to have productive chats within themselves and do not wish to use platforms with loads of distractions (eg. Facebook). Also people with more than one mobile device can make use of the seamless database service to avail instant messaging on each of their devices, without needing to logout of any device (the flaw in Whatsapp).

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 in communicating with their fellow mates.

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 a centralized database server. 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 registered users and start private conversation or group conversations.

- The users can find each other with their usernames if they are not connected yet. These connections will then be stored in User's database and he can private chat with these users anytime or can add them to a group.

2.5. References

Extensive research has been conducted for excellent study materials, some of the sources are listed -

- NodeJS Documentation, Here.
- ExpressJs Documentation, Here.
- Flutter Documentation, Here.
- Dart Docs, Here.
- Nodemailer, Here.
- Various JS library docs,
  - NodeMailer, Passport, JWT, Bcrypt,
- MongoDB Docs, Here.
- SocketIO Docs, Here.
- StackOverFlow !
- Various Private Blogs at
  - Medium, zKoder,
- Course at YouTube !

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 (Storing Data, Backing up, Handling data queries, etc).
- **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. Eg, React (turns making web apps with javascript a breeze).

- **Protocols** : A communication protocol is a system of rules that allow two or more entities of a communications system to transmit information. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods.
- **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** : A software bug is an error, flaw or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.
- **Debugging** : The process of finding and fixing bugs.
- **Schema** : The database schema of a database is its structure described in a formal language supported by the database management system (DBMS)

- **Network** : A computer network is a group of computers that use a set of common communication rules over digital interconnections for the purpose of sharing resources located on or provided by the network nodes

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.
- Users can send text messages to private conversation or group conversation. In addition to text messages users will also be able to send images, audio and other multimedia formats and set their own image as their profile picture for their friends to see.

3.3. Operating Environment

- A Centralised Database (hosted at mongodb.com) will be used.
- Operating System : Android, iOS
- Database Type : NoSQL.
- UI : Flutter Framework (Dart Programming Language)

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 is necessary for creating an account. These informations include First name, Last name, email id, a Unique Username, Password and a set of optional informations 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.
- Once logged in the user can utilise the features of the application to the fullest.

4.1.3. Starting a conversation

- Any user can start a conversation with the people present in his/her friends list by searching for them by username and the app will provide a list of all users(in friend's list) whose username starts with that name and the user can select the one he/she wants to talk with.
- 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.

4.1.5. Adding a new friend

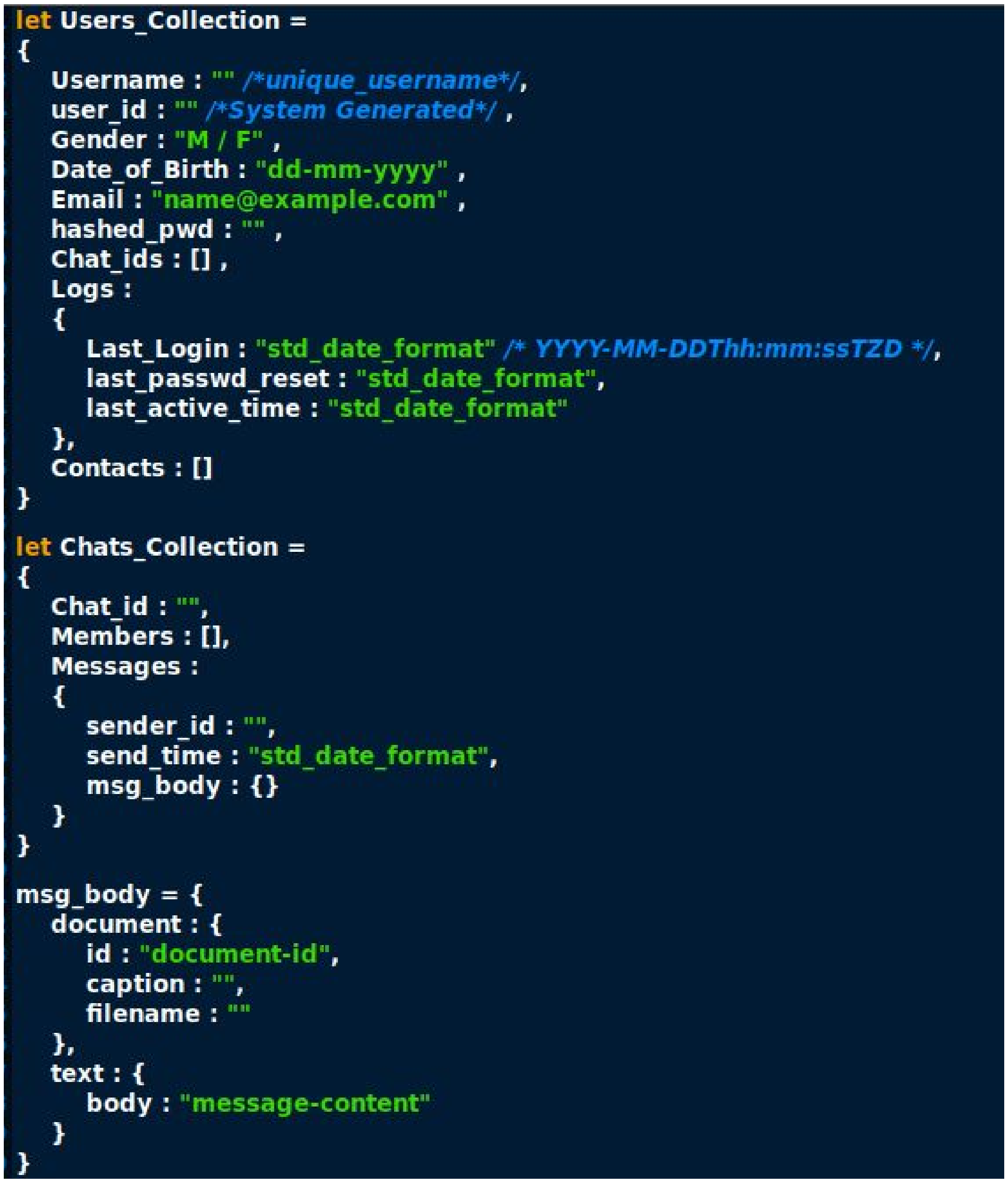
- To add a new friend in your friend’s list, your friend should have a verified account on Slabber.
- After that you can search the username of your friend and send him/her request to start a conversation.

4.1.6. Logout

- Logging out of the app won’t delete your app chats and data but you will not receive any messages but they will be received as soon as you log back in.

4.2 Non-functional Requirements

4.2.1. Schema :



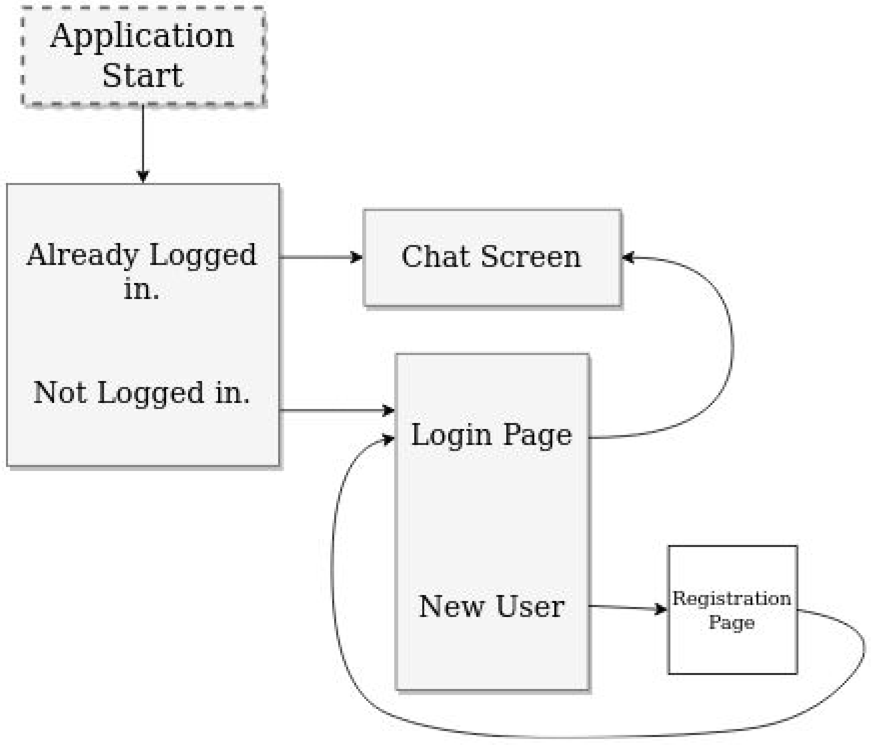
```
let Users_Collection =
{
   Username : "" /*unique_username*/,
   user_id : "" /*System Generated*/ ,
   Gender : "M / F" ,
   Date_of_Birth : "dd-mm-yyyy" ,
   Email : "name@example.com" ,
   hashed_pwd : "" ,
   Chat_ids : [] ,
   Logs :
   {
      Last_Login : "std_date_format" /* YYYY-MM-DDThh:mm:ssTZD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

4.3 Backend Apis

Backend contains different api for different functions. Using this the work loads can be divided into different api endpoints and load from both servers and user's devices can be reduced. Different api endpoints take care of different loads

4.3.1 /signup

- This api endpoint takes care of signing up users to the database of the app. This is a post api which requires all details of user i.e. full name, username, gender, email, country and password in json format. It then hashes the password using bcrypt algorithm and stores this information in mongodb database. It then goes ahead to send a confirmation email to the user's email id. Which contains another api endpoint to verify the user. Namely the '/confirmation' api. On successful registration of the user the server creates a jwt token and passes it to the frontend.

4.3.2 /login

- This api endpoint as the name suggests verifies the user and logs them in. It first checks if the request contains a json web token, if it is present then that token is verified and the user is prompted accordingly otherwise the user is directly sent to the login page which again uses the same api to verify the username and password of the user. If the user does not contain a jwt token then a new one is assigned to the user.

4.3.3 /confirmation

- This api endpoint is used to verify the email id of the user. An email is sent to the user's email id using two different apis either from /signup after successful registration of user or from /resend api which will be discussed further. The mail will contain a url to the confirmation api along with a token and email id embedded within the url. The api then verifies the token for the email id and changes the isVerified status of the user accordingly. If verified the user can now use their application.

4.3.4 /resend

- In case there was some error or by any chance the user was unable to receive the confirmation email then they can use this api endpoint (embedded in frontend as resend email button) to resend the confirmation email that contains the link for confirming the user's email id.

4.3.5 /sendrequest

- When a user adds a username to make them their friend, this api will be fired which first checks if the user with the provided username exists, if it does then a request is added to his/her receivedRequest list and the current user's sentRequest list is updated.

4.3.6 /acceptrequest

- When the user receiving the friend request clicks on the accept request button then the /acceptrequest api endpoint is fired that takes in the username of the users. It first checks that both users exist and if they do then it proceeds to add both of them in each other's friend list. It also checks whether they are already in their friends list or not. It only adds the reference of the users if they are not in the friends list to avoid redundancy in data.

4.3.7 /createroom

- This api endpoint is used for creating chat rooms. The http request to this endpoint will require a json object that contains infos of the admin's username, name of the chat room and a list of usernames which will be the members of the chat. Since there can be multiple chat rooms with the same name, it directly creates the chat room with the predefined schema for the chatroom and then proceeds to add all the users from the list of usernames. Only those who exist, the api endpoint also checks the existence of each member.

4.3.8 /createPrivateChat

- Similar to a chatroom, a private chat also contains all the messages that the users sent. Private chat however, is restricted to only two members who can send messages to each other. Hence, the shema for private chats require two different users and a list of messages. This endpoint first checks if there exists a chat that contains both the users and returns the object id of the private chat room with only the two users.

5. External Interface Requirements

5.1. User Interfaces

- **Front-end software**: **Flutter -** Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web and desktop from a single codebase. It uses only a single programming language named "Dart". Advantages that Flutter have over other frameworks and technologies is :-
    1. Fast Development - Can easily make apps in milliseconds with Stateful Hot Reload feature. Contain a rich set of fully-customizable widgets to build native interfaces in minutes.
    2. Expressive and Flexible UI - Quickly ship features with a focus on native and end-user experiences. Layered architecture allows for full customization, which results in incredibly fast rendering and expressive and flexible designs.
    3. Native Performance - Flutter's widgets incorporate all critical platform differences such as scrolling, navigation, icons and fonts, and your Flutter code is compiled to native ARM machine code using Dart's native compilers.

  Flutter uses a special widget tree model to implement the UI of the app. MaterialApp becomes the root widget of the tree. Widgets are further classified into Stateless and Stateful widgets depending upon their current and expected future states.

- **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 the 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	Flutter, which is used extensively to build cross platform mobile applications with Dart.
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