GNDEC CHAT BOT

Minor Project Synopsis

Submitted To:-
Dr.Pankaj Bhambri
Submitted By:-
Ekam Preet Singh (1921028/1905327)
Muskan Sharma(1921067/1905366)
Kunwar Arora(1921058/1905356)

Department of Information Technology
Guru Nanak Dev Engineering College, Ludhiana

Content

Introduction

Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance was quite recent. Our project is an example of a chat server. It is made up of two applications - the client application, which runs on the user's web browser and server application, runs on any hosting servers on the network. To start chatting client should get connected to server where they can do private and group chat. Security measures were taken during the last one

- The purpose of this application is to allow users with internet connection to engage in private and public conversation
- The development of this project centered on the development of a message protocol that would allow the application to properly log in users, send messages and perform system maintenance

Objectives

1. To design and develop instant messaging solution for end users.
2. To enable multi-chatting functionality over the network for seamless communication.

Technical Interface

1 Architecture

The architecture of the application consists of the back end and the front end, both of them having their own set dependencies (libraries and frameworks). The front end is the presentation layer that the end user sees when they enter the site. The back end provides all the data and part of the logic and it is running behind the scenes.

1.1 Front End :

Having separated the server-side from the client side, SPA's dynamically fetch data from the API as the user is browsing the site, avoiding to refresh the whole page whenever the user has filled in a form or navigated to another part of the site.
Both Angular and React were being maintained by powerful corporations, Google and Facebook respectively

1. **User Registration:** User must be able to register for the application through an Email, Username and Password. The user's email will be the unique identifier of his/her account on Chat Application.

2. **Send Message :** User should be able to send instant message to any contact on his/her Chat Application contact list.

3. **Broadcast Message:** User should be able to create groups of contacts. User should be able to broadcast messages to these groups.

1.2 Back-end :

The "back end" refers to the logic and data layers running on the server side. In our case, the back end makes sure that the data introduced through the client application (the front end), is valid. we have to make sure that all the requests we receive are first verified by the server: the requested URI is supported, the user has the appropriate permissions,etc.

1. **Express**: A Node.js framework which makes web development fast. It abstracts most of the complexity behind the web server and acts as an HTTP route handler.

2. **Mongoose:** A MongoDB high-level library. By using objects as database models,it handles inserts, updates, and deletes, as well as the validation for each of its fields.

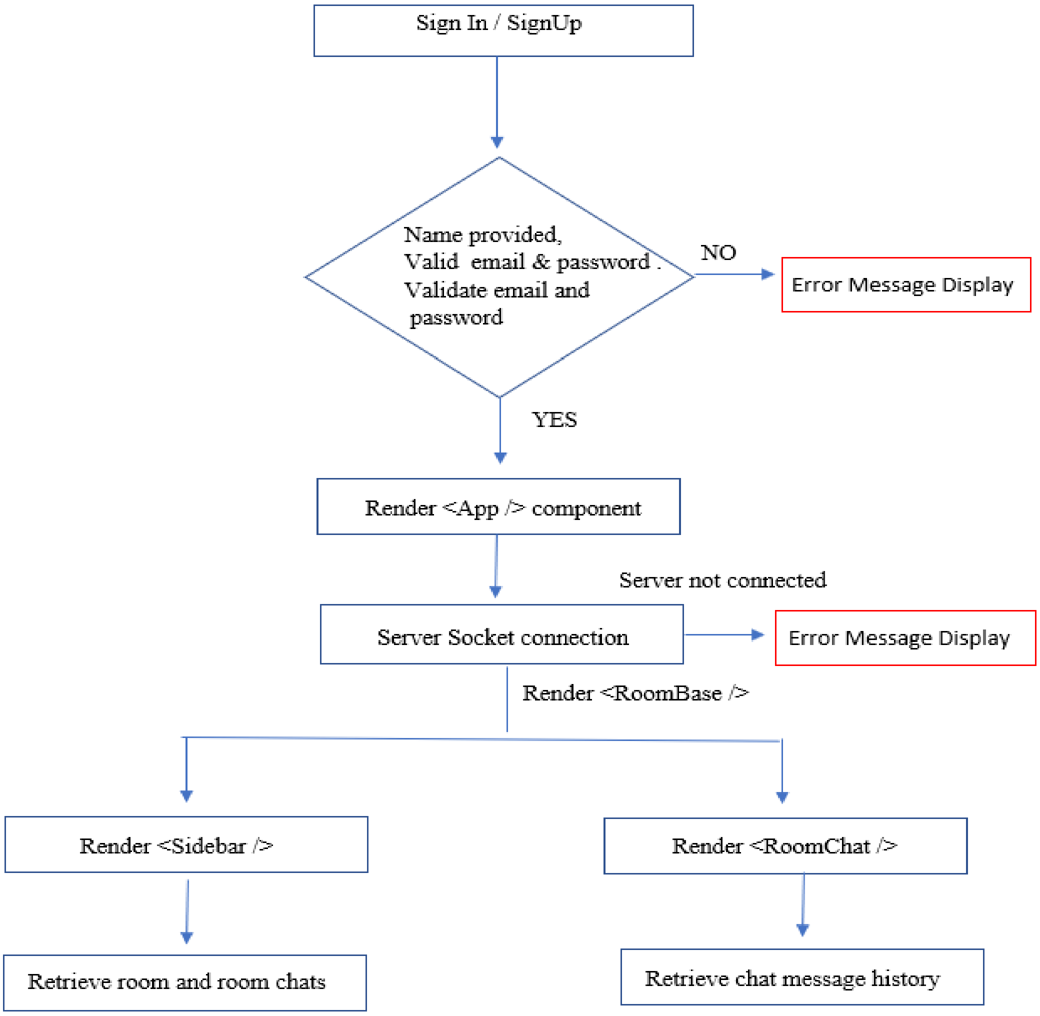
3. **Node.js:** Node.js provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent

4. **Robustness:** In case user's system crashes, a backup of their chat history must be stored on remote database servers to enable recoverability.

5. **Performance:** Application must be lightweight and must send messages instantly

6. **Dependencies:** The third-party package or modules installed using npm are specified in this segment.

Methodology



Software and Hardware Requirements

Hardware :

- Recommended Processor : Intel core i3 or above
- Recommended Processor Speed : 2GHz CPU
- Recommended RAM : 4 GB or above
- Recommended OS: Windows 7 or above, MacOS, Linux

Software :

- VSCode is a text editor and source code editor that provides and environment for developing
- MongoDB 5.x with native time series collections optimized for IoT and financial apps.
- Node.js v17 / NPM 6.0.0
- React JS 18.0.0 / Angular JS 14.0.0
- Express JS 4.17.3
- Twilio: Communication API's
- Stream API's

References

[1] `https://www.mongodb.com/docs/realm/schemas/`
Accessed: 05-04-2022

[2] `https://expressjs.com/en/guide/routing.html`
Accessed: 01-04-2022

[3] `https://www.techtarget.com/searchdatamanagement/definition/CRUD-cycle`
Accessed: 25-03-2022

[4] `https://devdocs.magento.com/guides/v2.4/frontend-dev-guide/bk-frontend-dev-guide.html`
Accessed: 30-03-2022

[5] `https://www.twilio.com/docs/api`
Accessed: 02-04-2022

[6] `https://yellow.systems/blog/guide-to-the-chat-architecture`
Accessed: 05-04-2022

[7] `https://www.toptal.com/nodejs/secure-rest-api-in-nodejs`
Accessed: 29-03-2022

[8] `https://getstream.io/blog/build-chat-messaging-app/`
Accessed: 04-04-2022

[9] Holmes, S. and Harber, C., Getting MEAN with Mongo, Express, Angular, and Node, Second Edition.