IV Semester
Mini Project
Report On

(Group Chat Web Application)

(2021-2022)

B. Tech(CSE)

Submitted by:

Vipul Chauhan

University Roll. No. : 2018855

Class Roll. No./Section:62/A

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

DECLARATION

I, **Vipul Chauhan** student of **B-tech, Semester 4,** Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun, declare that the technical project work entitled " **Group Chat Web Application** " has been carried out by me and submitted at the beginning of 5th semester for the award of degree in B-tech of **Graphic Era Hill University** during the academic year **2021-2022**. The matter embodied in this synopsis has not been submitted to any other university or institution for the award of any other degree or diploma.

Date: 13/07/22

CERTIFICATE

This is to certify that the project report entitled "**Group Chat Web Application** " is a bonafide project work carried out by Vipul Chauhan , Roll no- 2018855. in partial fulfilment of award of degree of B- tech of Graphic Era Hill University, Dehradun during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated. The project has been approved as it satisfies the academic requirements associated with the degree mentioned.

ACKNOWLEDGEMENT

Here by I am submitting the project report on **"Group Chat Web Application "** as per the scheme of Graphic Era Hill University, Dehradun.

I would like to express my sincere thanks to my project Co-Ordinator Mr. Chandradeep Bhatt, for their patience, support and encouragement throughout the completion of this project and having faith in me.

Finally, I am very much thankful to all the faculty members of the Department of Computer Science and Engineering, friends and my parents for their constant guidance and immense support for completing project within limited time frame.

Vipul
Chauhan
Roll no.
2018855

TABLE OF CONTENTS

ABSTRACT

Group 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. My project is an example of a group 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 group chat and share multimedia content with other members of the group. So the name which I have given to the project is CHATVERSE.

CHATVERSE is a social-networking tool that leverages on technology advancement thereby allowing its users communicate with multiple people at the same time and share media. It can be used for messaging, share updates and photos, enhance local socializing in pidgin English. This app was made keeping one thing in mind that the UI should be smooth and for creating the groups are for short span of time because people used to make a lot of groups and are not even active in that group so with CHATVERSE the group would be available by the time the user logout and with it people can be free from the fear that their data is being tracked or stored because in CHATVESRE we are not storing any messages or media of the user, the thing we are storing is user login information.

This is how the application logo looks like

INTRODUCTION

CHATVERSE is created with the mind set to add a level of clean UI (user interface) or a solid app structure function over a secure broadcast network. CHATVERSE is an effort for a more modern approach to internet security on a communication medium. The Implementation of CHATVERSE is based on HTML,CSS,JS,NODEJS,MONGODB, and SOCKET IO where HTML,CSS,JS is for the app structure and UI (user interface) tool, NODEJS and SOCKET IO for the server setup, MongoDB for user login details. An end to end connection stream was used for data transfer from client to server and back to client. In conclusion the application worked well without a lag and having a 95% acceptance when tested with a potential user.

PROBLEM STATEMENT

Developing a group-chat web application: The web application should allow users to chat with each other and share photos and other multimedia content.

Starting any application or service has many problems but one of the main problems is which tool, language, stack or framework to build one's service or application on. As building a real time application has to do with slow latency message delivery which in turn means latency, data transfer size over the network must be as low as possible.

INNOVATIVE IDEAS OF PROJECT

GUI: Easy to use GUI (Graphical User Interface), hence any user with minimal knowledge of operating a system can use the software.

Platform independence: The messenger operates on any system irrelevant of the underlying operating system.

Unlimited clients: "N" number of users can be connected without any performance degradation of the server.

AIMS AND OBJECTIVES

MESSAGING

One of the primary use of CHATVERSE is messaging. This feature is pivotal as people can join the group and start chatting.

This is made possible as each of the CHATVERSE user will have a unique login id and password and once the user log in we provide the user to choose his username to join the chat and buy entering the group name and group password.

And once the user is in the group than he/she can chat with their friends and share the multimedia content.

And this can be achieved by using socket io for creating the different rooms for the different groups

FILE TRANSFER

With the sophistication design of the CHATVERSE, individuals will be able to share files to the group without size constrains ranging from images, videos, to large documents files like zip, dmg, and so on.

DELIVERABLES

There are 5 major things I hope to achieve with this application, which include.

I. Speed in usage
II. Easy and friendly UI
III. Privacy Protection

BACKGROUND AND MOTIVATION

The Group Chat Application we are currently using are not that much secure because they are storing the user messages and media although they are claiming that the data is completely encrypted but that’s not the point they have the access to the private messages of the users and their media.

So keeping that thing in mind I created CHATVERSE because with it people can be free from the fear that their data is being tracked or stored because in CHATVESRE we are not storing any messages or media of the user, the thing we are storing is user login information. “Why there is the need to store the past messages for the future live the present” this is the quote followed by CHATVERSE and so the groups are for short span of time because people used to make a lot of groups and are not even active in that group so with CHATVERSE the group would be available by the time the user logout.

What is MongoDB?

- MongoDB is a cross-platform document-oriented NoSQL database used for high volume data storage that provides high performance, high availability and easy scalability.
- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in the application code, making data easy to work with.
- The data model available within MongoDB allows users to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
- MongoDB works on concept of collections and documents. Each database contains collections which in turn contains documents. Each document can have varying number of fields. The size and content of each document can also be different from each other.

Key Components of MongoDB Architecture

1. **_id** – This is a 24-digit unique identifier field required in every MongoDB document in the collection. The _id field is like the document's primary key. If the user creates a new document without an _id field, MongoDB will automatically create the field.

2. **Collection** - Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Typically, all documents in a collection are of similar or related purpose.

3. **Document** - A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

4. **Database** - Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

5. **Field** - A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.

What is Express.js?

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the Node.js foundation.
- Express provides us the tools that are required to build our app, be it single-page, multi-page or hybrid web applications. It is flexible as there are numerous modules available on **npm(Node Package Manager)**, which can be directly plugged into Express.
- Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.
- Pug (earlier known as Jade) is a terse language for writing HTML templates. It produces HTML, supports dynamic code and code reusability (DRY). It is one of the most popular template languages used with Express.
- Express can be thought of as a layer built on the top of the Node.js that helps manage a server and routes. It allows users to setup middleware to respond to HTTP Requests and defines a routing table which is used to perform different actions based on HTTP method and URL.
- Express allows to dynamically render HTML Pages based on passing arguments to templates.
- Express is asynchronous and single threaded and performs I/O operations quickly.

Why use Express?

- Ultra-fast I/O.
- Asynchronous and single threaded.
- MVC like structure.
- Robust API makes routing easy.

What is Node.js?

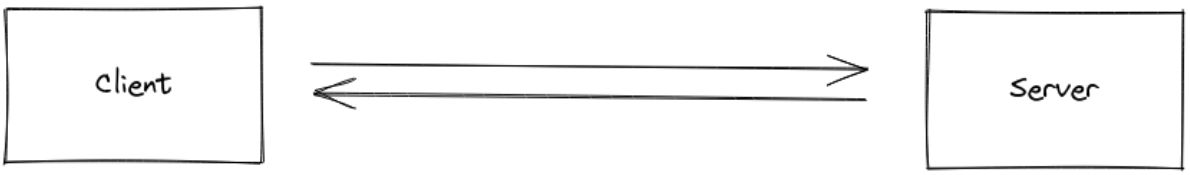
- Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single- page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.
- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009.
- Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.
- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Features of Node.js

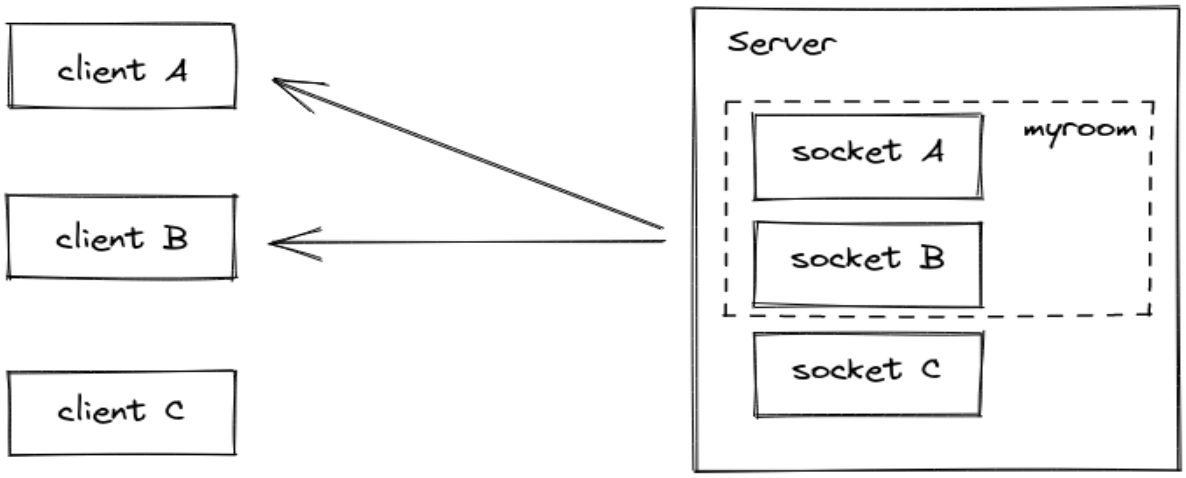
1. **Extremely fast :** Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. **I/O is Asynchronous and Event Driven :** All APIs of Node.js library are asynchronous i.e. non-blocking. So, a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.
3. **Single threaded :** Node.js follows a single threaded model with event looping.
4. **Highly Scalable :** Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
5. **No buffering :**Node.js cuts down the overall processing time while uploading audios videos and files. Node.js applications never buffer any data. These applications simply output the data in chunks.
6. **Open source :** Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js application.

What is Socket.IO?

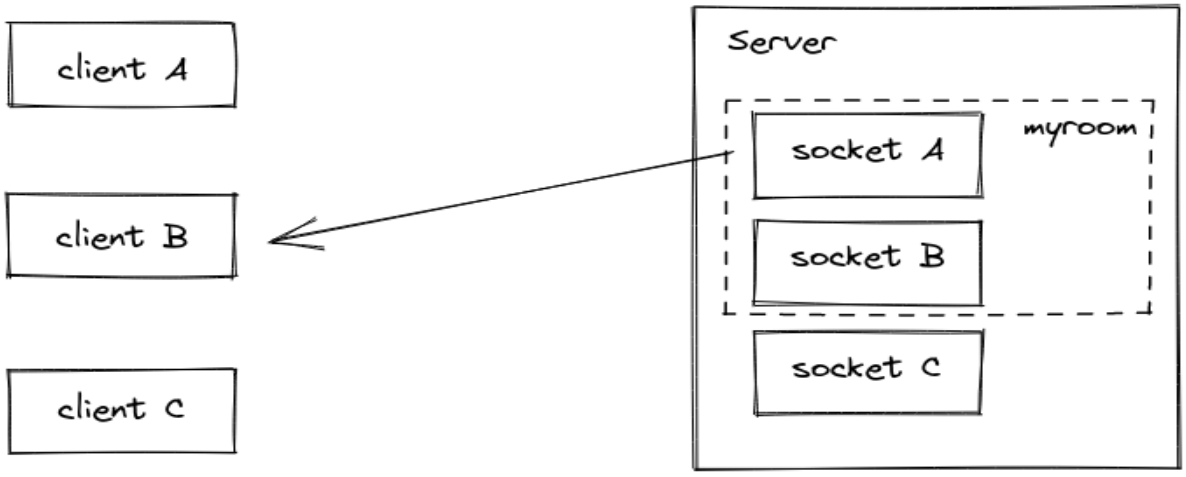
Socket.IO is a library that enables **low-latency**, **bidirectional** and **event-based** communication between a client and a server.



Rooms is socket io- A *room* is an arbitrary channel that sockets can `join` and `leave`. It can be used to broadcast events to a subset of clients:



In that case, every socket in the room **excluding** the sender will get the event.



REQUIREMENTS AND DESIGN

Hardware Requirements

For Server:

- Active internet Connection
- Minimum 8 GB RAM
- Minimum 128GB SSD

For Client:

- Active internet Connection

Software Requirements

For Server:

- Any Operating System
- Node JS
- nodemon
- Mongo Db
- Visual Studio Code-(IDE)

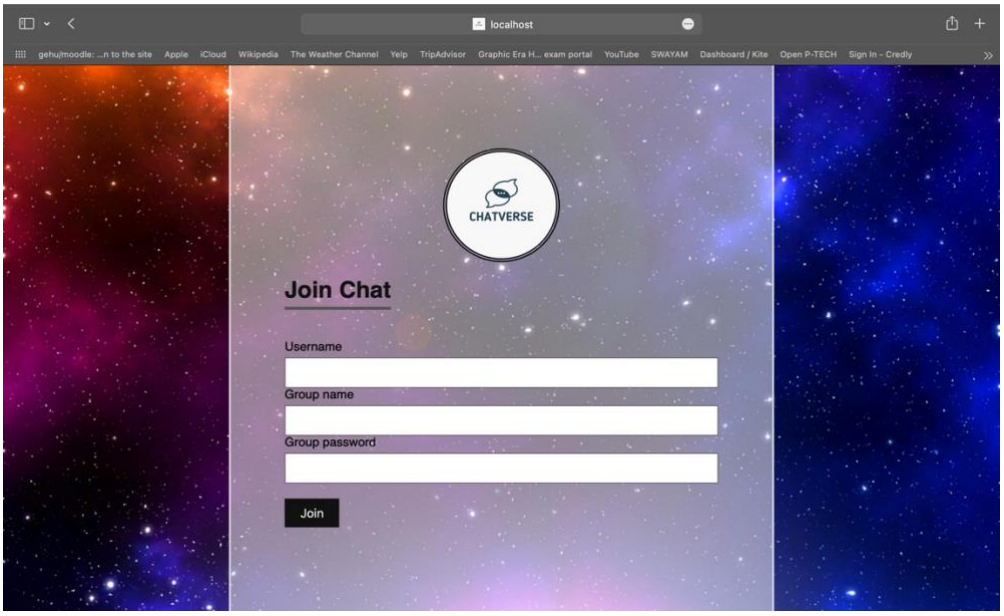
For Client:

- Any Operating System
- Any Web Browser

Libraries

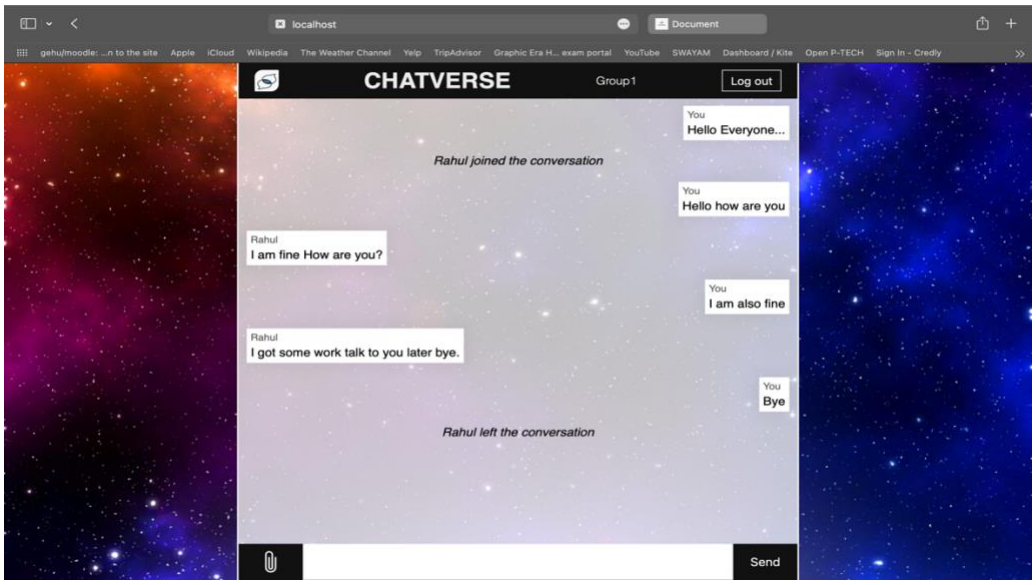
- body-parser
- express
- mongoose
- nodemon
- path
- socket.io

Group Chat Interface



This is the interface which will open when the user log into the CHATVERSE by his/her login Credentials.

Here he/she has the choice of choosing the username of his/her choice and then enter the name of the group and password of the group which they want to join if the group already exist then the user will enter the group only if the password is correct and if not then the new group will be created when the group with the name doesn't exist.

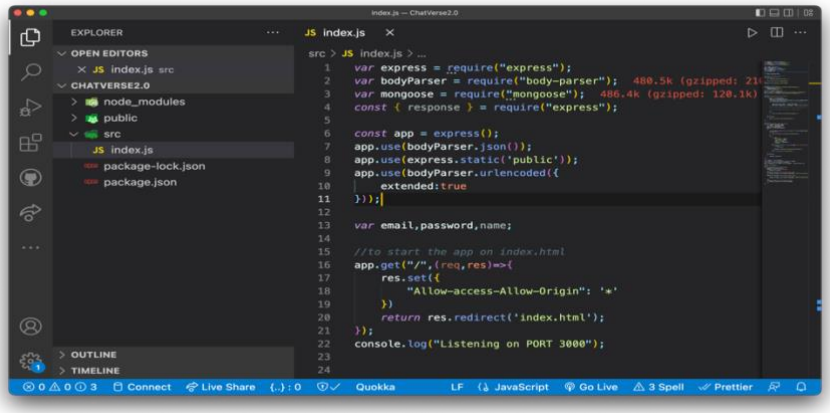


Once the user enter the group he/she can wait for the other persons to join the room and when the new user joins the group the message in the group will come that the user has joined the conversation and after that the users can chat with

each other share the multimedia content with each other and if the user want to exit the chat they can press the log out button on the top left corner of the window and when the user log out the message in the group will come that the user left the conversation in the above screenshot I have created the demo chat to display the app.

CHAT SERVER ENGINE

Chat server consist of the index.js file which is the node.js file which is run using nodemon index.js to start the server and the index.js file will somehow look like



Other than this the dependencies that are required to run the server are

SCREENSHOT

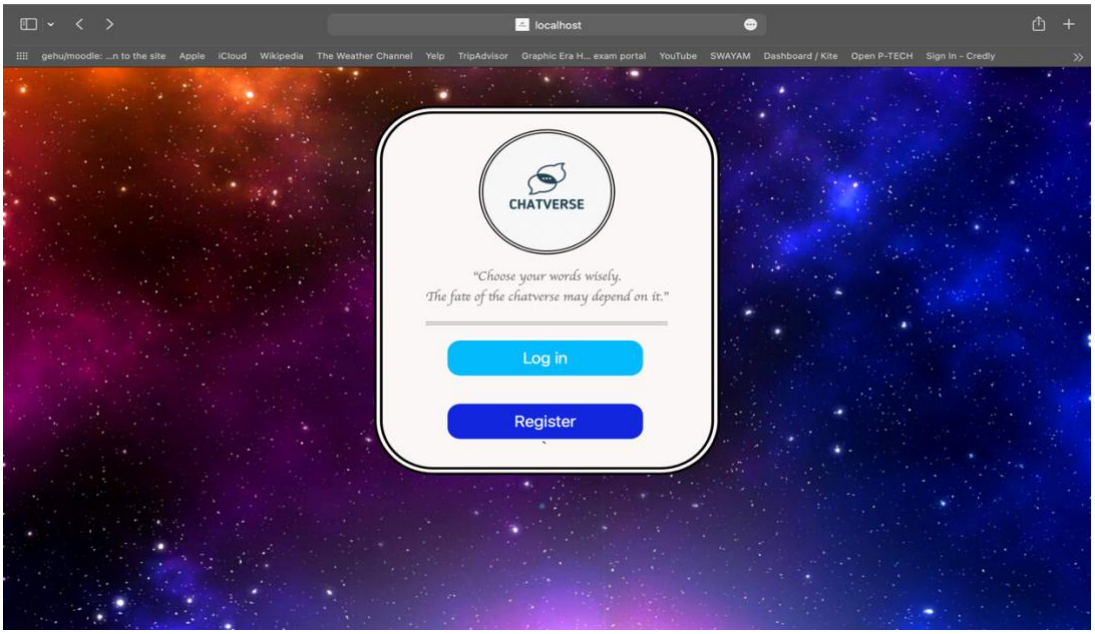


Figure 4.1 Starting page.

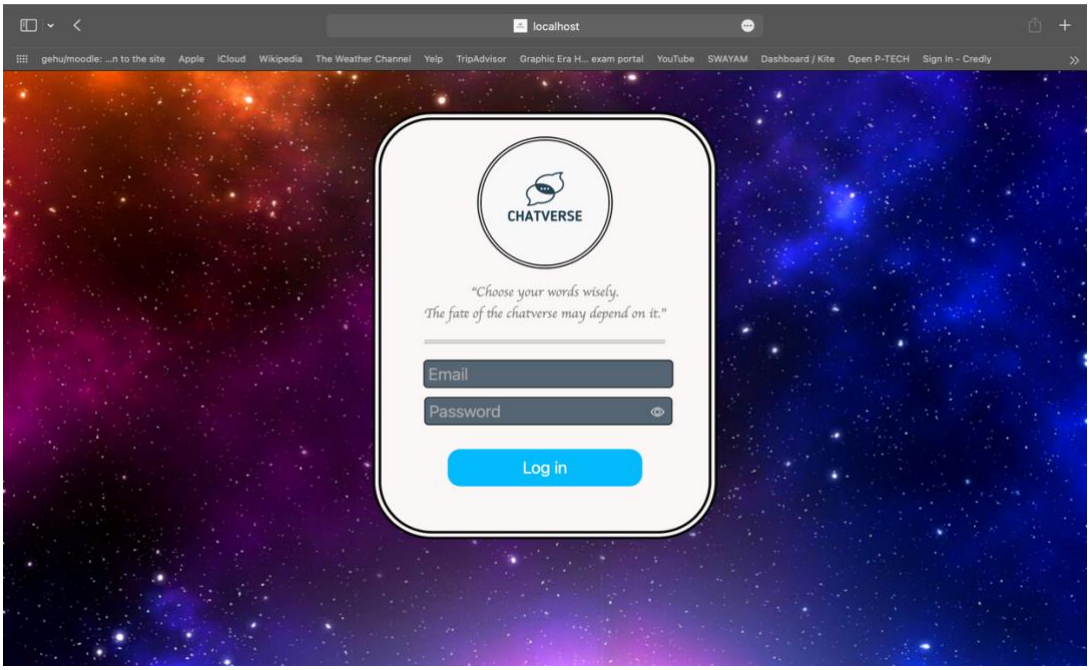


Figure 4.2 Login Interface.

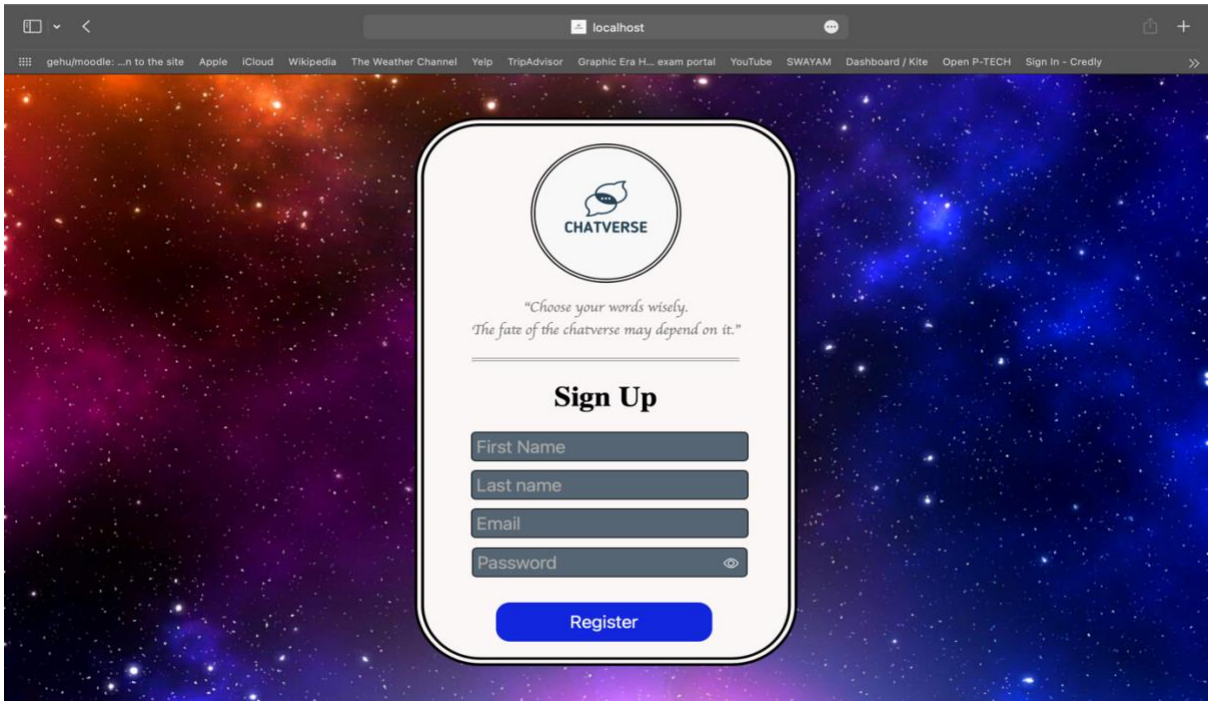


Figure 4.3 Signup interface

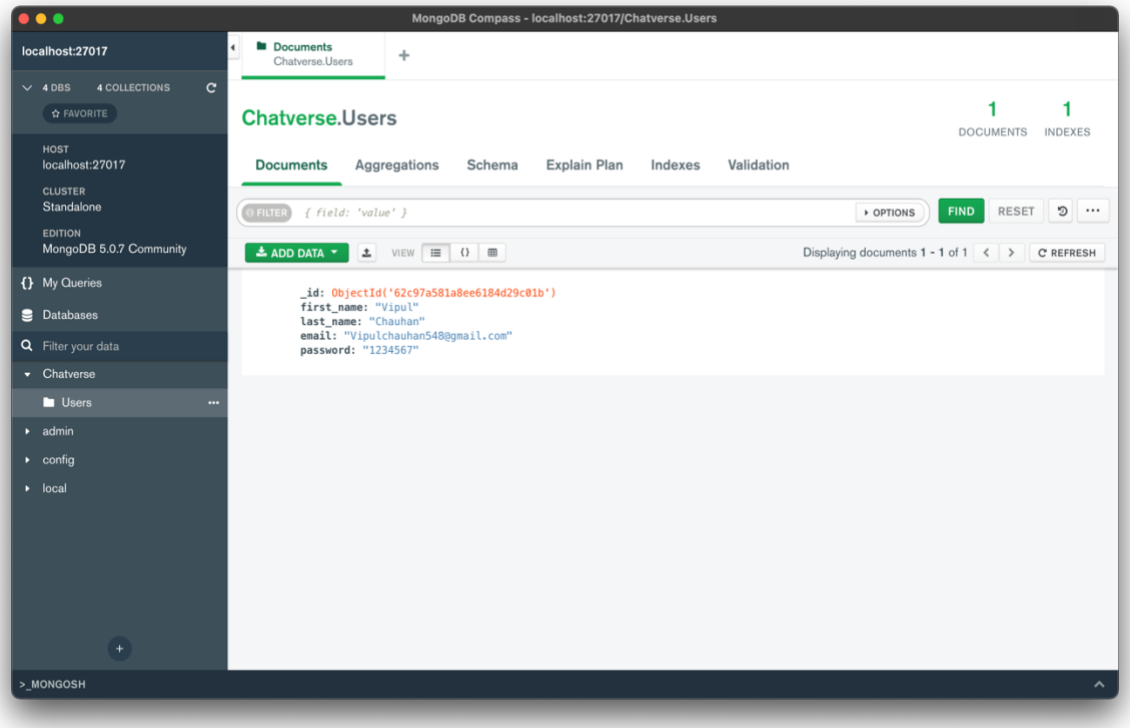


Figure 4.4 MongoDB Database for login credentials

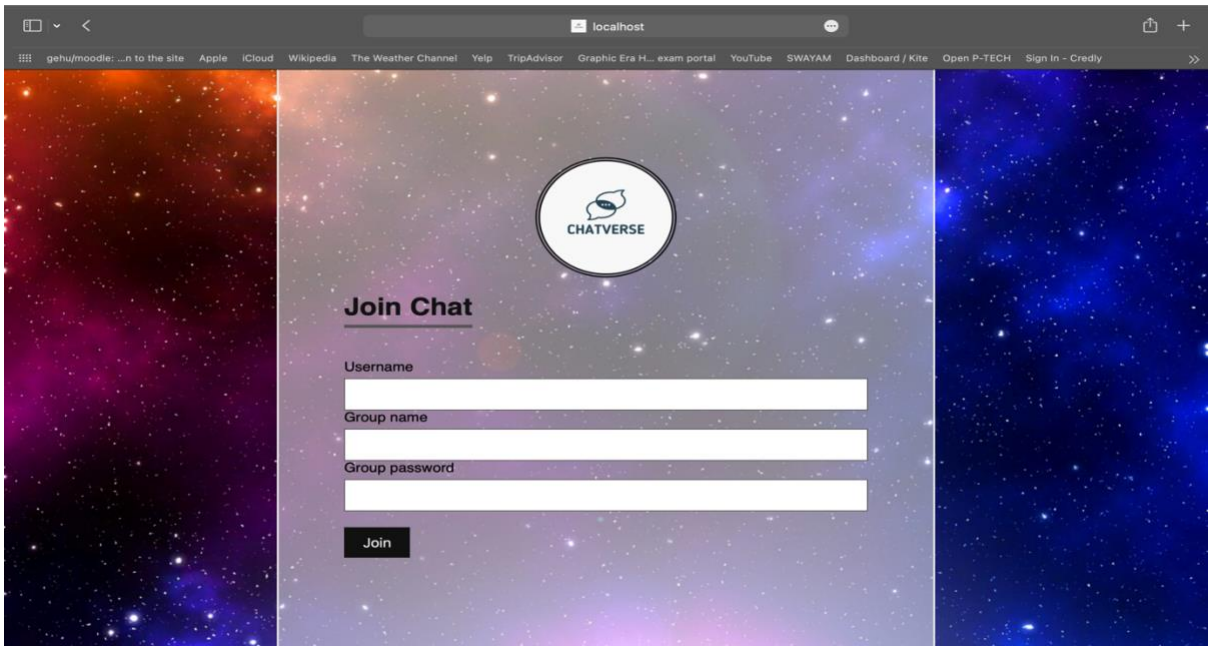


Figure 4.5 Group Chat joining interface.

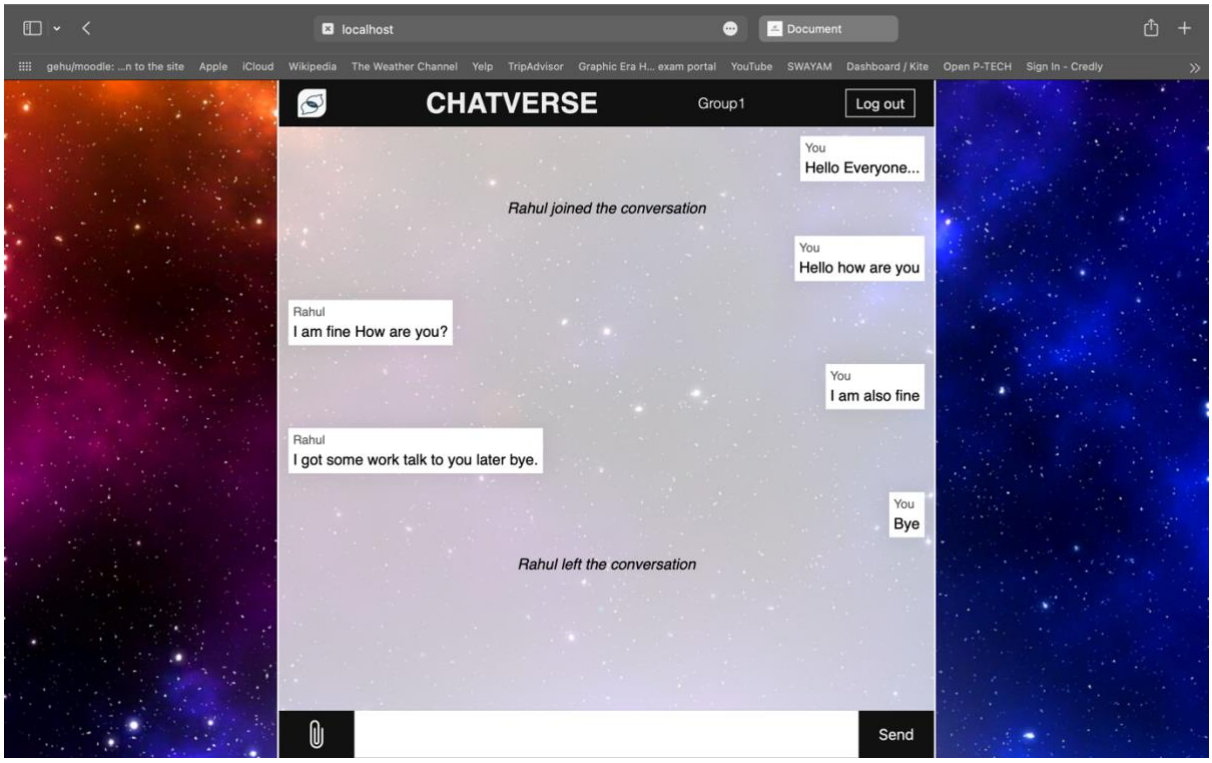


Figure 4.6 Group Chat interface.

CONCLUSION

There is always a room for improvements in any apps. Right now, we are just dealing with text communication and media sharing. There are several chat apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a chat service Web app with high quality user interface. In future we may be extended to include features such as:

- Voice Message
- Audio Call
- Video Call
- Group Call ,etc.

REFERENCE

MongoDB：https://docs.mongodb.com/ecosystem/drivers/,
https://www.guru99.com/what-is-mongodb.html

ExpressJS：https://expressjs.com/en/guide/routing.html ,

https://www.javatpoint.com/expressjs-tutorial

Npm：https://www.npmjs.com/

NodeJS：https://nodejs.org/en/docs/,
https://www.javatpoint.com/nodejs-tutorial

Socket.IO：https://socket.io/docs/v4/