A

MINI PROJECT REPORT ON

“REAL-TIME CHAT APPLICATIONS”

Project carried out

By

Asit Kumar	**2200350100018**
Dheeraj Kumar	**2200350100023**
Akash Mahato	**2200350100008**
Saif Ullah	**2200350100056**

Babu Banarsi Das Institute of Technology (BBDIT)

Near, Duhai, Ghaziabad, Uttar Pradesh 201206

COURSE: - B. TECH CSE 3RD YEAR

SESSION (2024-25)

Under the supervision of

Date of Submission

Declaration

I hereby declare that the project titled " REAL-TIME CHAT APPLICATIONS " is a result of my own work and research. This project has been developed as part of my mini project for B.Tech in Computer Science Engineering (CSE) 3rd year at Babu Banarsi Das Institute of Technology (BBDIT).

I affirm that the work presented in this report is original and has not been submitted for any other academic purpose. All sources of information and references used in the preparation of this project have been duly acknowledged.

I would like to express my gratitude to my project supervisor, Ass. Prof. ANVISHA SHARMA, for their guidance and support throughout the development of this project. I also extend my thanks to my peers and family for their encouragement and assistance.

Course: B.Tech (CSE) 3rd year

Date:

Asit Kumar 2200350100018

Dheeraj Kumar 2200350100023

Akash Mahato 2200350100008

Saif Ullah 2200350100056

Prof. Name ANVISHA SHARMA

Department of Computer science Engg. **Department of CSE**

Acknowledgment

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of my project, "Real-Time Chat Applications."

First, I thank my project supervisor, Ass. Prof. ANVISHA SHARMA, for their invaluable guidance and support throughout the development of this project. Their insights were instrumental in shaping my work.

I also appreciate my classmates and friends for their collaboration and assistance. Their ideas and suggestions greatly enhanced the quality of the application.

Additionally, I am grateful to my family for their unwavering support and motivation, which kept me focused throughout this journey.

Finally, I acknowledge the online resources and documentation that provided essential information on the technologies used in this project, including HTML, CSS, JavaScript, Node.js, Socket.IO, and MongoDB.

Thank you all for your support and encouragement.

Project carried out by

Asit Kumar **2200350100018**

Dheeraj Kumar **2200350100023**

Akash Mahato **2200350100008**

Saif Ullah **2200350100056**

Date

Abstract

REAL-TIME CHAT APPLICATIONS is a real-time chat application designed to facilitate instant communication over the internet. Utilizing modern web technologies such as HTML, CSS, and JavaScript for the front-end, and Node.js with Socket.IO for real-time communication, REAL-TIME CHAT APPLICATIONS provides a seamless user experience. The application features user registration, instant messaging, and chat history storage using MongoDB. This project aims to enhance online communication by integrating these technologies into a user-friendly platform.

Technologies Used

The following technologies were utilized in the development of the ChatUp! application:

- **HTML: For structuring the web pages and creating the user interface.**
- **CSS: For styling the application and enhancing the user experience.**
- **JavaScript: For implementing client-side logic and handling user interactions.**
- **MongoDB: For storing user data, chat history, and other relevant information.**
- **Socket.IO: For enabling real-time, bidirectional communication between clients and the server.**

System Architecture

The architecture of the REAL-TIME CHAT APPLICATIONS application consists of the following components:

- **Client-Side: The front-end of the application is built using HTML, CSS, and JavaScript. Users can access the chat interface through their web browsers.**
- **Server-Side: The server is built using Node.js and Socket.IO, which handles real-time communication and manages user connections.**
- **Database: MongoDB is used to store user profiles, chat messages, and other relevant data.**

Features

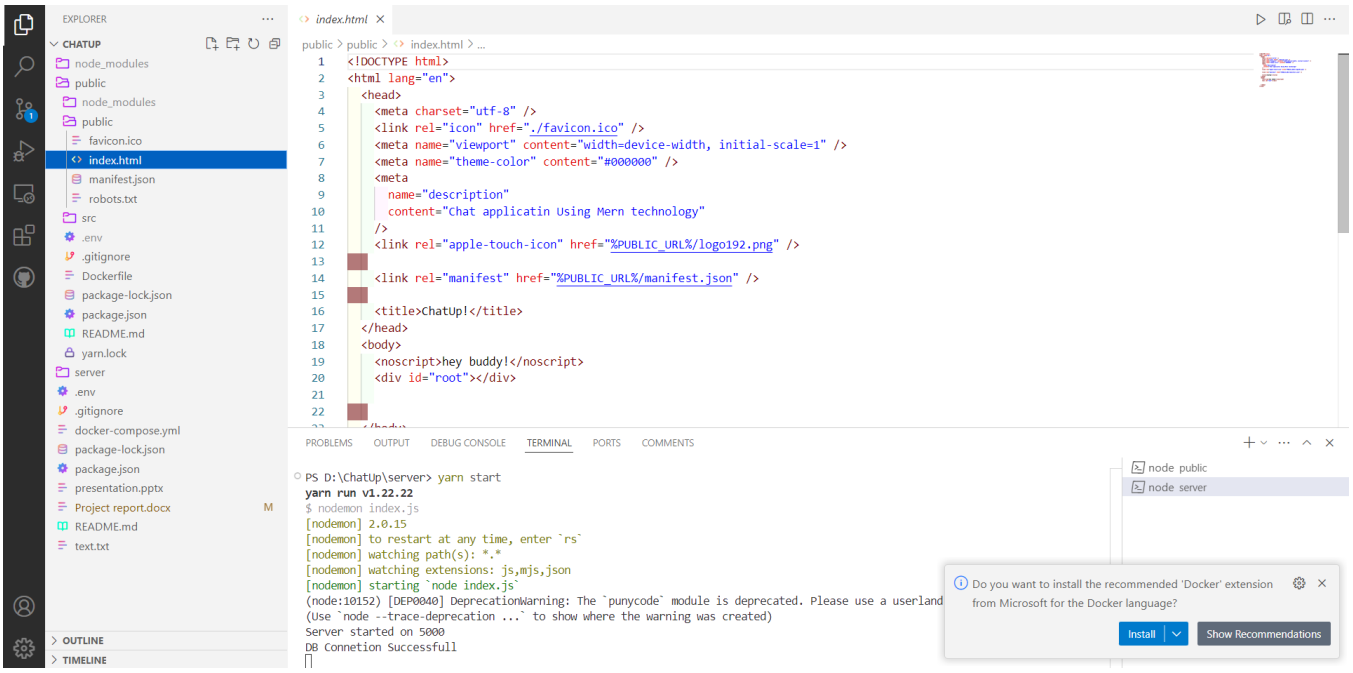
The REAL-TIME CHAT APPLICATIONS application includes the following features:

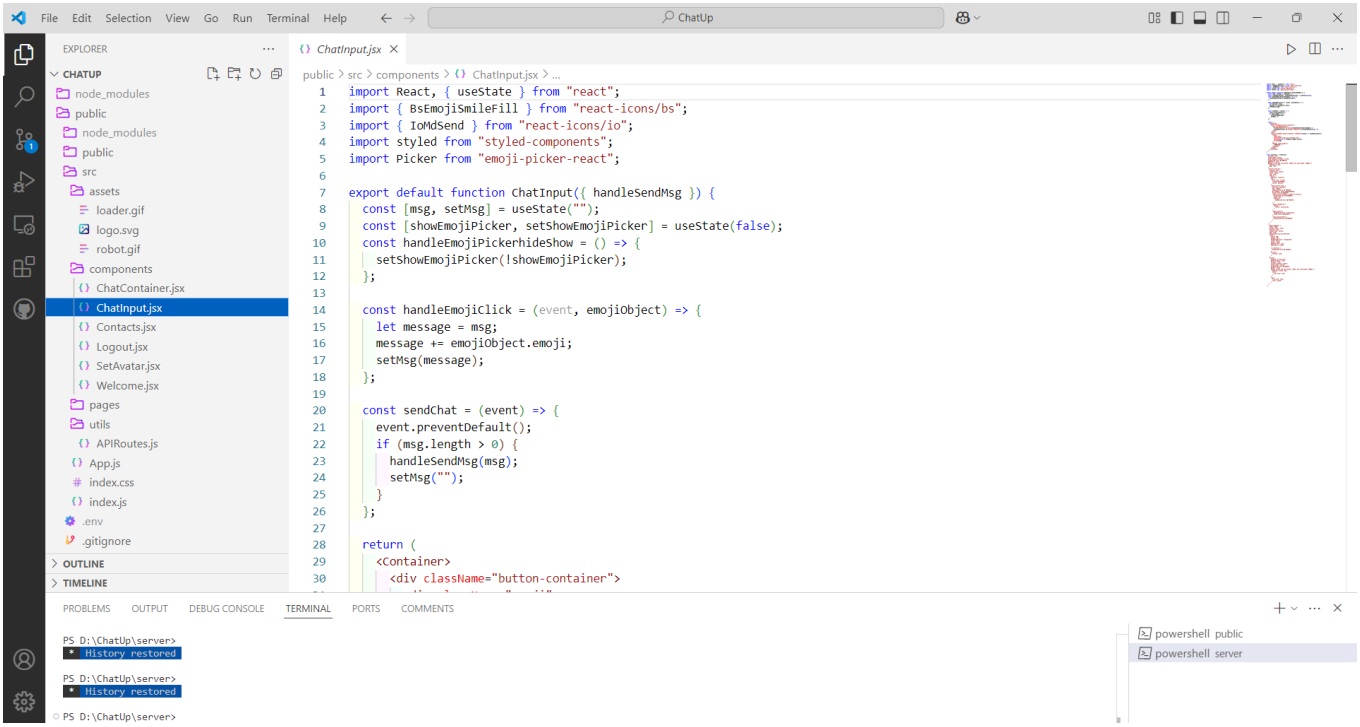
- **User Registration and Login: Users can create an account and log in to access the chat system.**
- **Real-Time Messaging: Users can send and receive messages instantly without refreshing the page.**
- **Chat History: Users can view their previous conversations stored in the MongoDB database.**
- **Responsive Design: The application is designed to be responsive and accessible on various devices.**

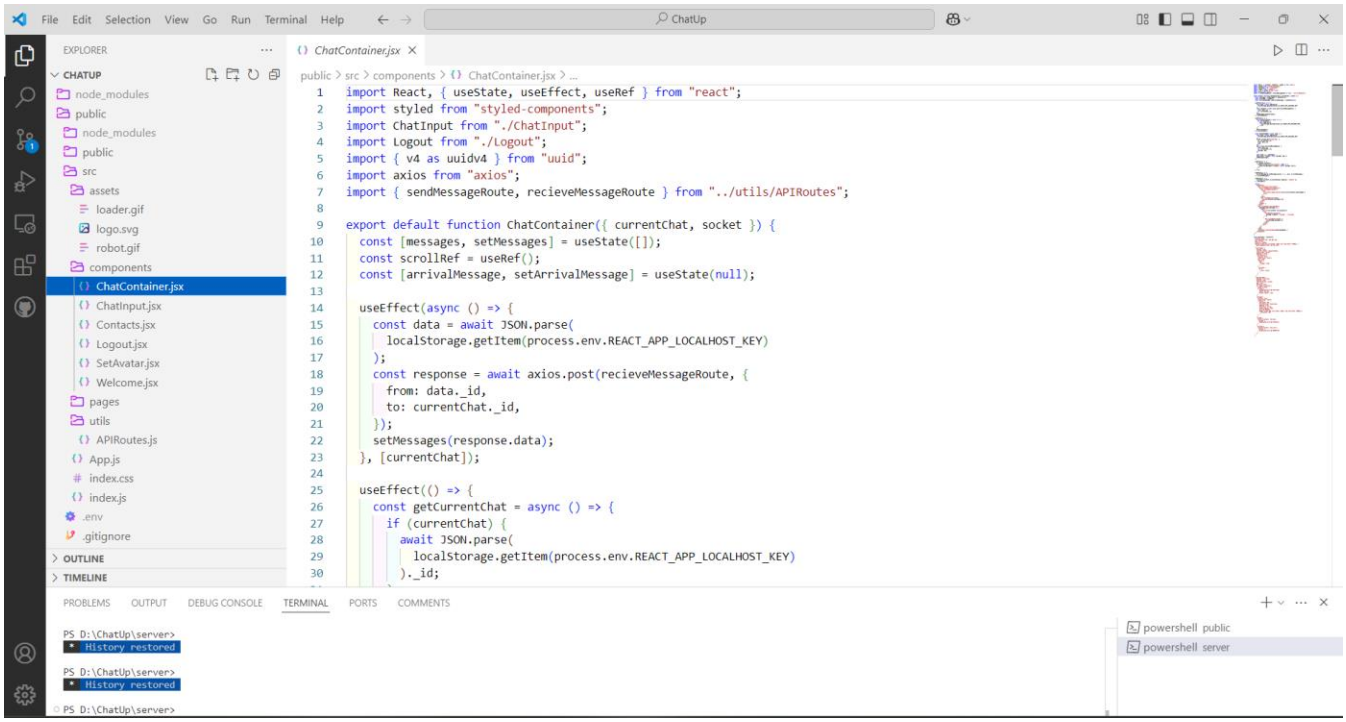
Implementation

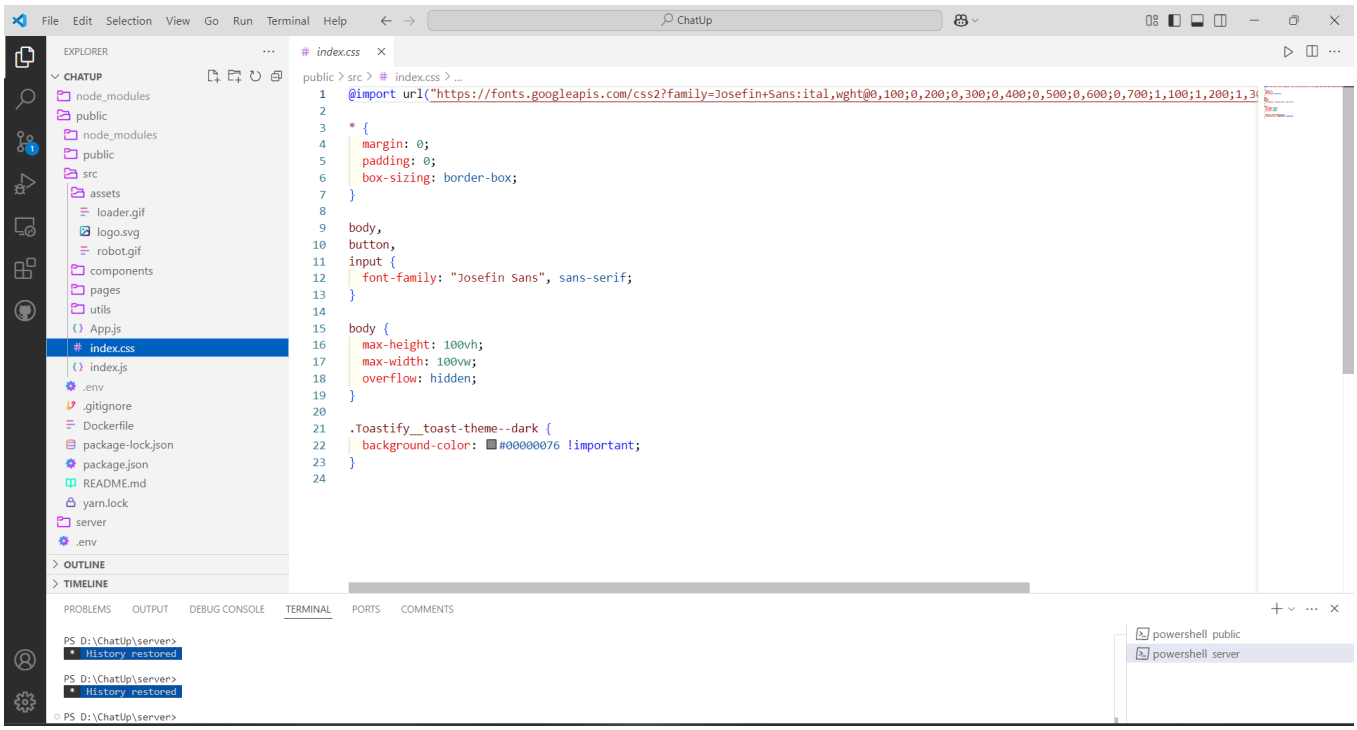
Front-End Development

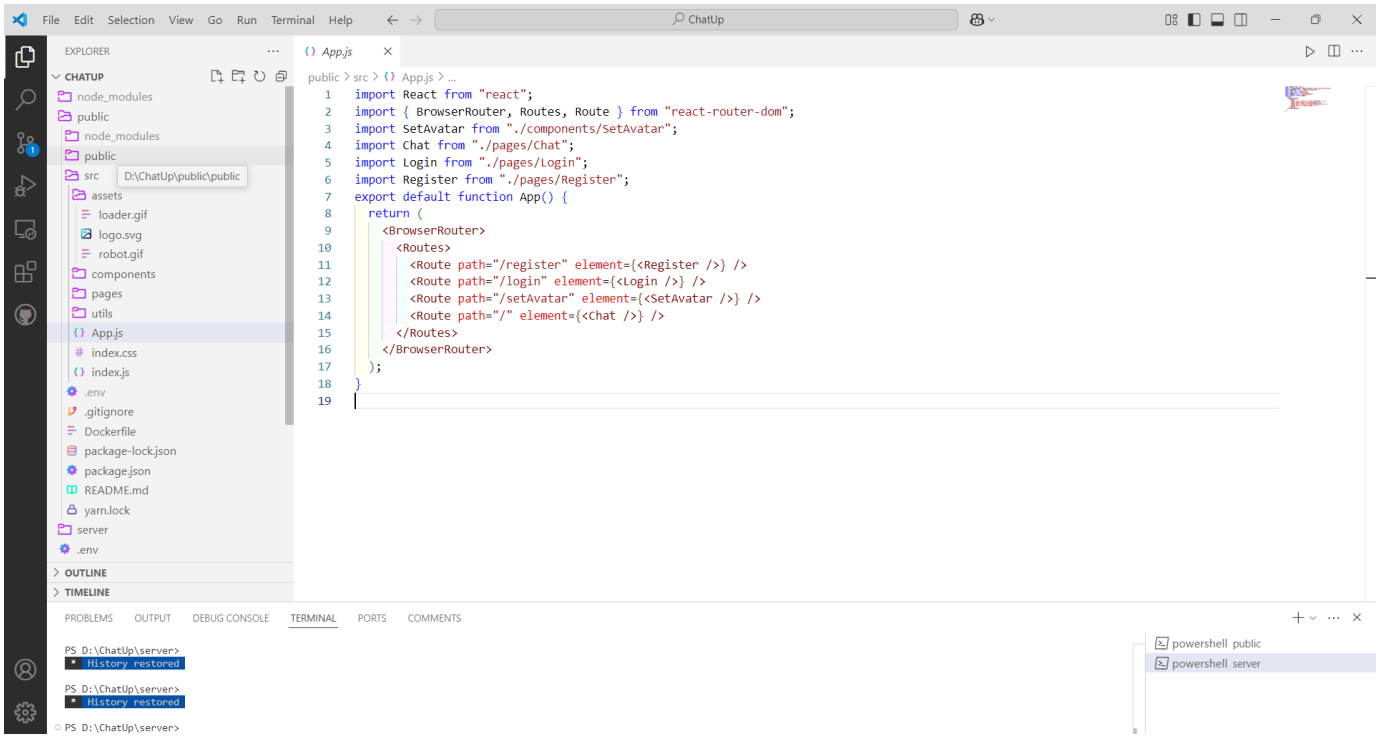
The front-end of the REAL-TIME CHAT APPLICATIONS application is developed using HTML and CSS to create a clean and intuitive user interface. JavaScript is used to handle user interactions, such as sending messages and updating the chat window in real-time.





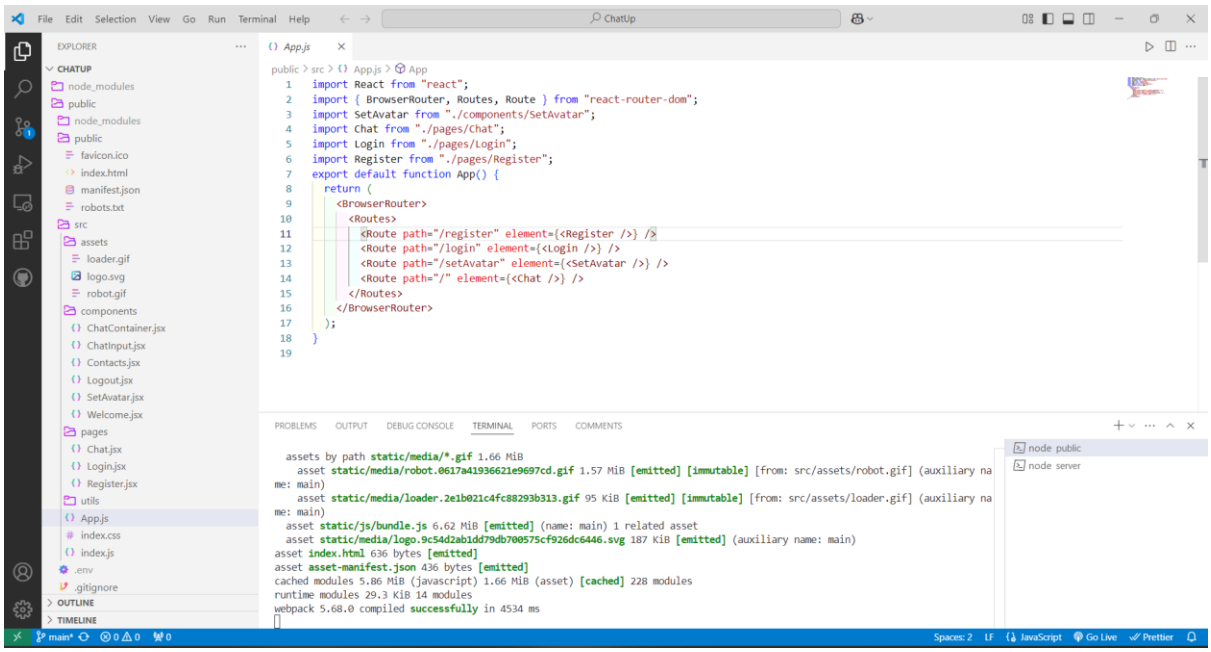

File Edit Selection View Go Run Terminal Help
ChatUp
EXPLORER
CHATUP
node_modules
public
node_modules
public
src
assets
loader.gif
logo.svg
robot.gif
components
ChatContainer.jsx
ChatInput.jsx
Contacts.jsx
Logout.jsx
SetAvatar.jsx
Welcome.jsx
pages
utils
APIRoutes.js
App.js
index.css
index.js
.env
.gitignore
OUTLINE
TIMELINE
ChatContainer.jsx
public > src > components > ChatContainer.jsx > ...
1 import React, { useState, useEffect, useRef } from "react";
2 import styled from "styled-components";
3 import ChatInput from "./ChatInput";
4 import Logout from "./Logout";
5 import { v4 as uuidv4 } from "uuid";
6 import axios from "axios";
7 import { sendMessageRoute, recieveMessageRoute } from "../utils/APIRoutes";
8
9 export default function ChatContainer({ currentChat, socket }) {
10 const [messages, setMessages] = useState([]);
11 const scrollRef = useRef();
12 const [arrivalMessage, setArrivalMessage] = useState(null);
13
14 useEffect(async () => {
15 const data = await JSON.parse(
16 localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
17);
18 const response = await axios.post(recieveMessageRoute, {
19 from: data._id,
20 to: currentChat._id,
21 });
22 setMessages(response.data);
23 }, [currentChat]);
24
25 useEffect(() => {
26 const getCurrentChat = async () => {
27 if (currentChat) {
28 await JSON.parse(
29 localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
30)._id;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
powershell public
powershell server

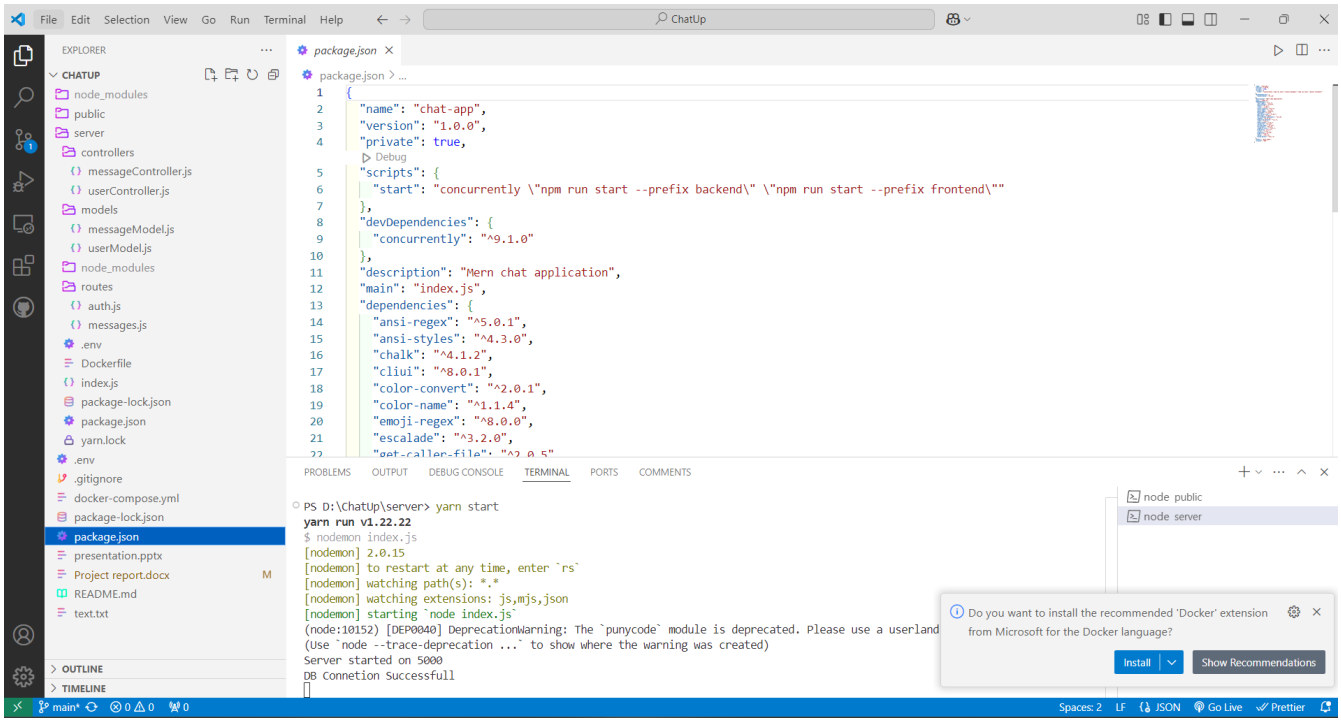

File Edit Selection View Go Run Terminal Help
ChatUp
EXPLORER
CHATUP
node_modules
public
node_modules
public
src
assets
loader.gif
logo.svg
robot.gif
components
pages
utils
App.js
index.css
index.js
.env
.gitignore
Dockerfile
package-lock.json
package.json
README.md
yarn.lock
server
.env
OUTLINE
TIMELINE
index.css
public > src > index.css > ...
1 @import url("https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;1,100;1,200;1,
2
3 * {
4 margin: 0;
5 padding: 0;
6 box-sizing: border-box;
7 }
8
9 body,
10 button,
11 input {
12 font-family: "Josefin Sans", sans-serif;
13 }
14
15 body {
16 max-height: 100vh;
17 max-width: 100vw;
18 overflow: hidden;
19 }
20
21 .Toastify__toast-theme--dark {
22 background-color: #00000076 !important;
23 }
24
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
powershell public
powershell server

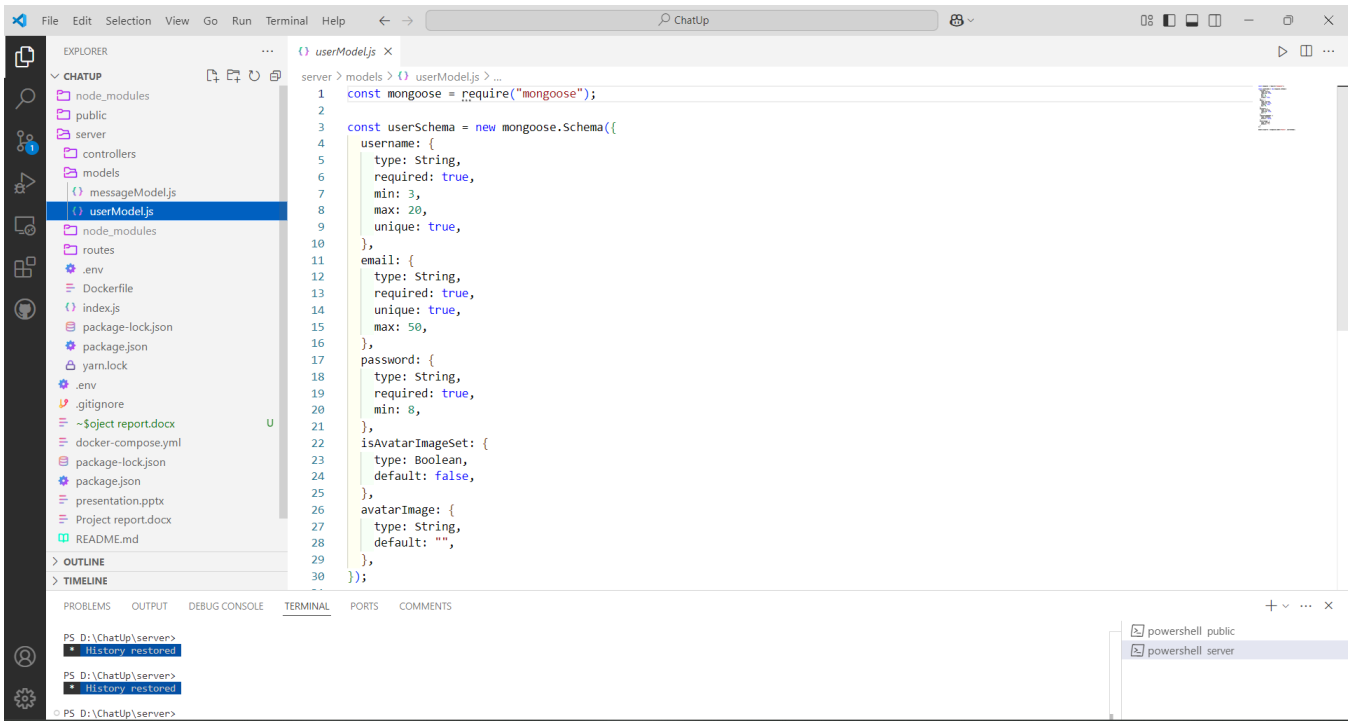


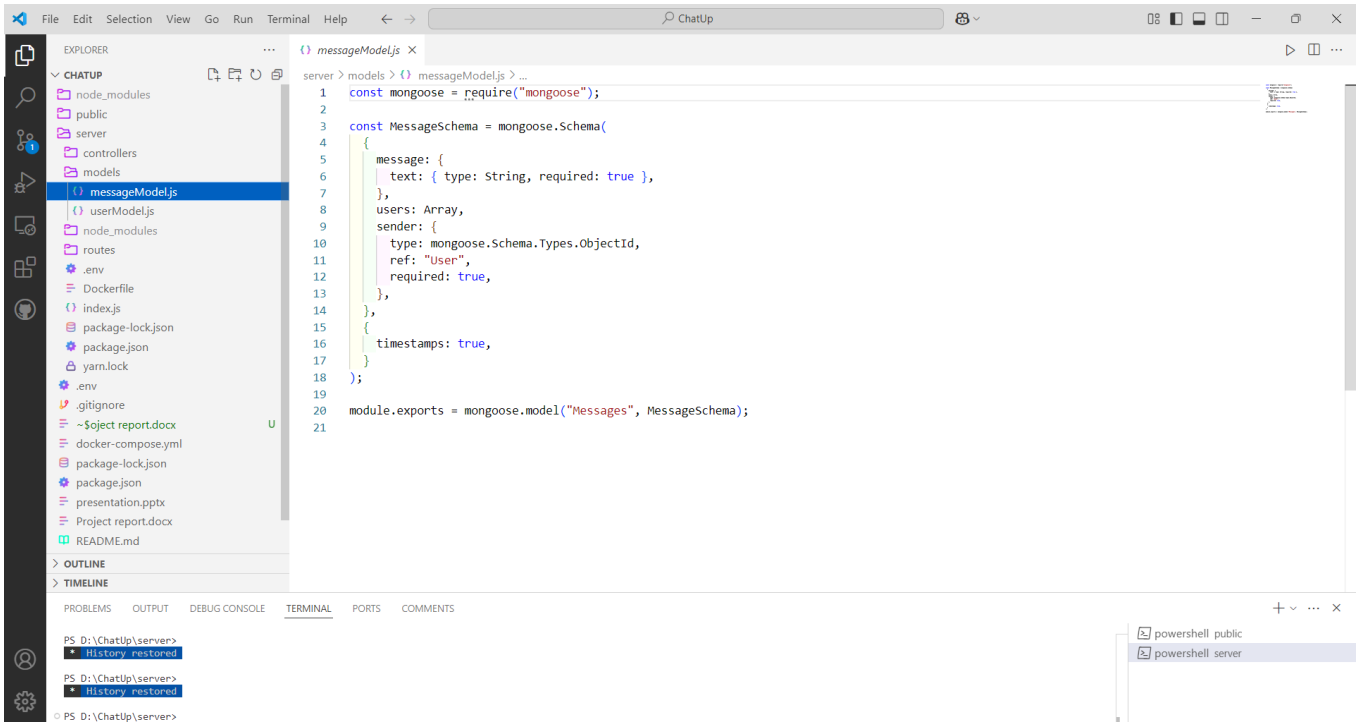
Back-End Development

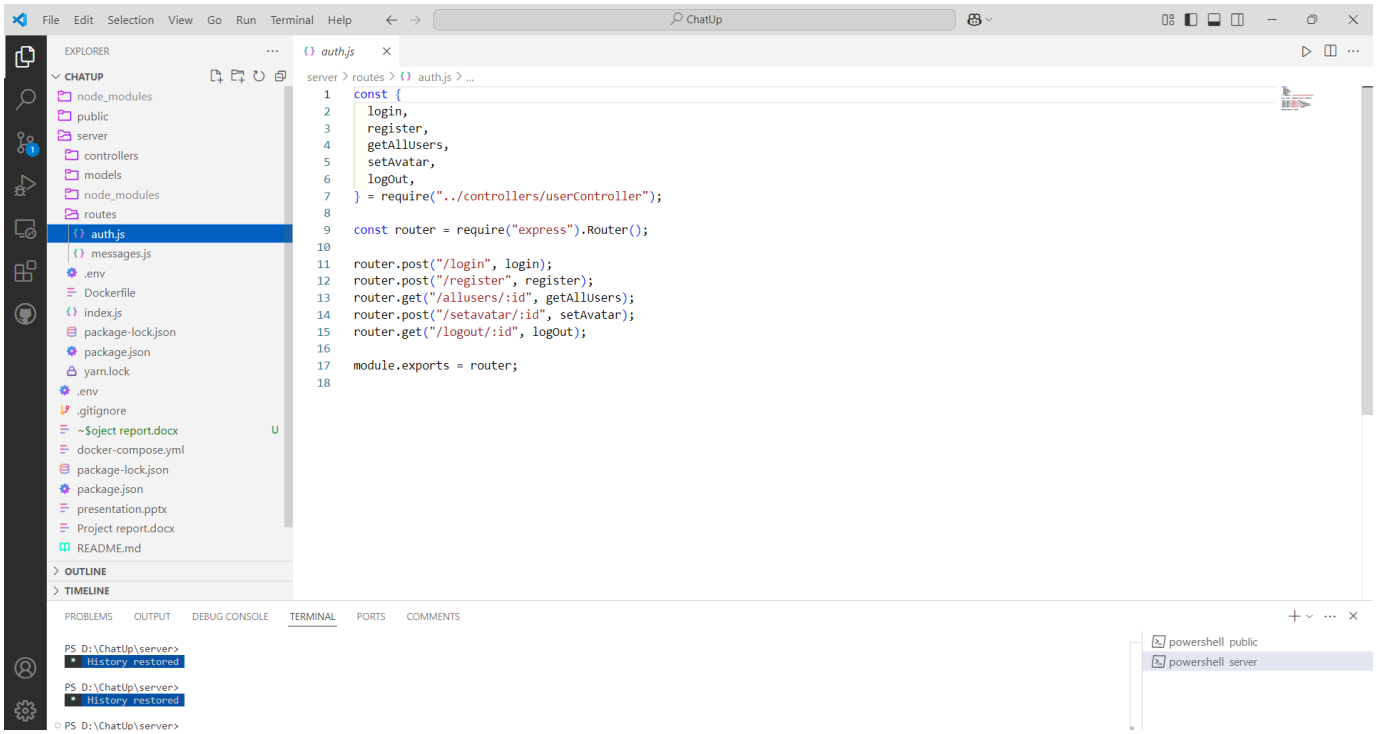
The back-end is built using Node.js and Socket.IO. The server listens for incoming connections and manages real-time communication between users. When a user sends a message, the server broadcasts it to all connected clients.




File Edit Selection View Go Run Terminal Help
ChatUp
EXPLORER
CHATUP
node_modules
public
server
controllers
messageController.js
userController.js
models
messageModel.js
userModel.js
node_modules
routes
auth.js
messages.js
.env
Dockerfile
index.js
package-lock.json
package.json
yarn.lock
.env
.gitignore
docker-compose.yml
package-lock.json
package.json
presentation.pptx
Project report.docx M
README.md
text.txt
OUTLINE
TIMELINE
package.json
package.json > ...
{
"name": "chat-app",
"version": "1.0.0",
"private": true,
Debug
"scripts": {
"start": "concurrently \"npm run start --prefix backend\" \"npm run start --prefix frontend\""
},
"devDependencies": {
"concurrently": "^9.1.0"
},
"description": "Mern chat application",
"main": "index.js",
"dependencies": {
"ansi-regex": "^5.0.1",
"ansi-styles": "^4.3.0",
"chalk": "^4.1.2",
"cliui": "^8.0.1",
"color-convert": "^2.0.1",
"color-name": "^1.1.4",
"emoji-regex": "^8.0.0",
"escalade": "^3.2.0",
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\ChatUp\server> yarn start
yarn run v1.22.22
$ nodemon index.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
(node:10152) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland
(Use `node --trace-deprecation ...` to show where the warning was created)
Server started on 5000
DB Connetion Successfull
node public
node server
Do you want to install the recommended 'Docker' extension from Microsoft for the Docker language?
Install
Show Recommendations
main*
Spaces: 2 LF JSON Go Live Prettier

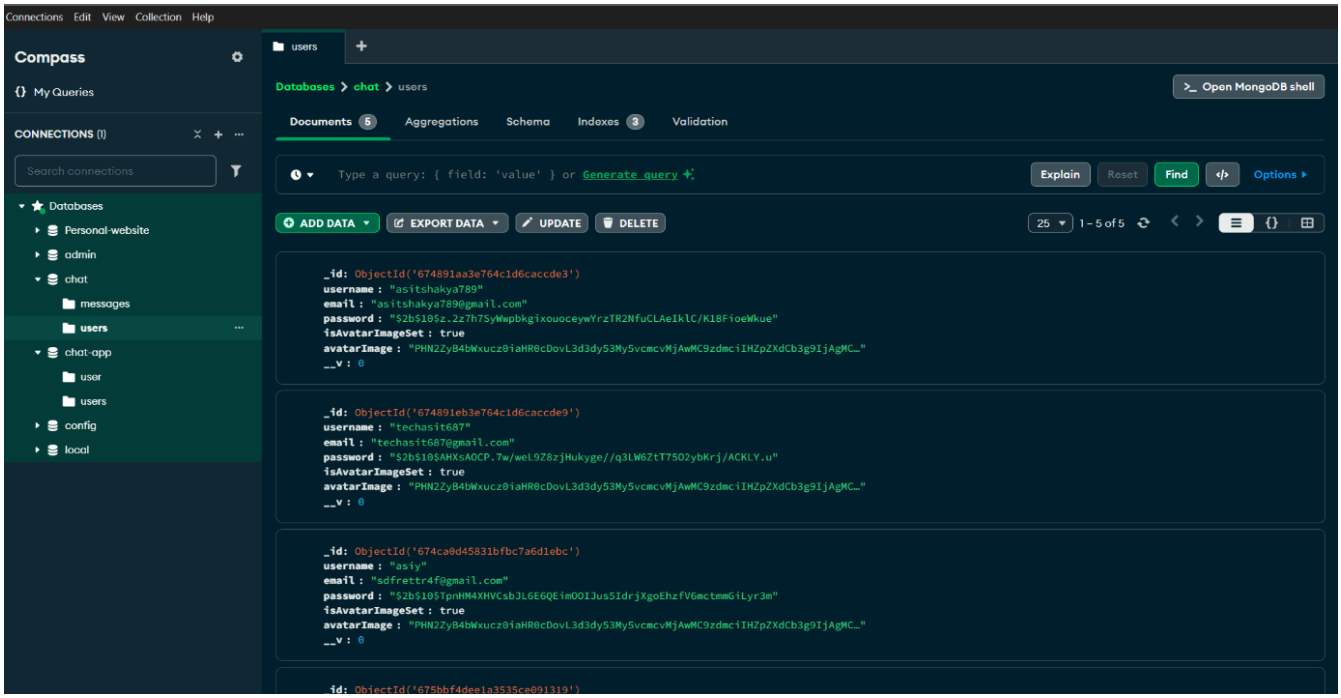

File Edit Selection View Go Run Terminal Help
ChatUp
EXPLORER
CHATUP
node_modules
public
server
controllers
models
messageModel.js
userModel.js
node_modules
routes
.env
Dockerfile
index.js
package-lock.json
package.json
yarn.lock
.env
.gitignore
~$oject report.docx U
docker-compose.yml
package-lock.json
package.json
presentation.pptx
Project report.docx
README.md
OUTLINE
TIMELINE
userModel.js
server > models > userModel.js > ...
const mongoose = require("mongoose");
const userSchema = new mongoose.Schema({
username: {
type: String,
required: true,
min: 3,
max: 20,
unique: true,
},
email: {
type: String,
required: true,
unique: true,
max: 50,
},
password: {
type: String,
required: true,
min: 8,
},
isAvatarImageSet: {
type: Boolean,
default: false,
},
avatarImage: {
type: String,
default: "",
},
});
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
powershell public
powershell server

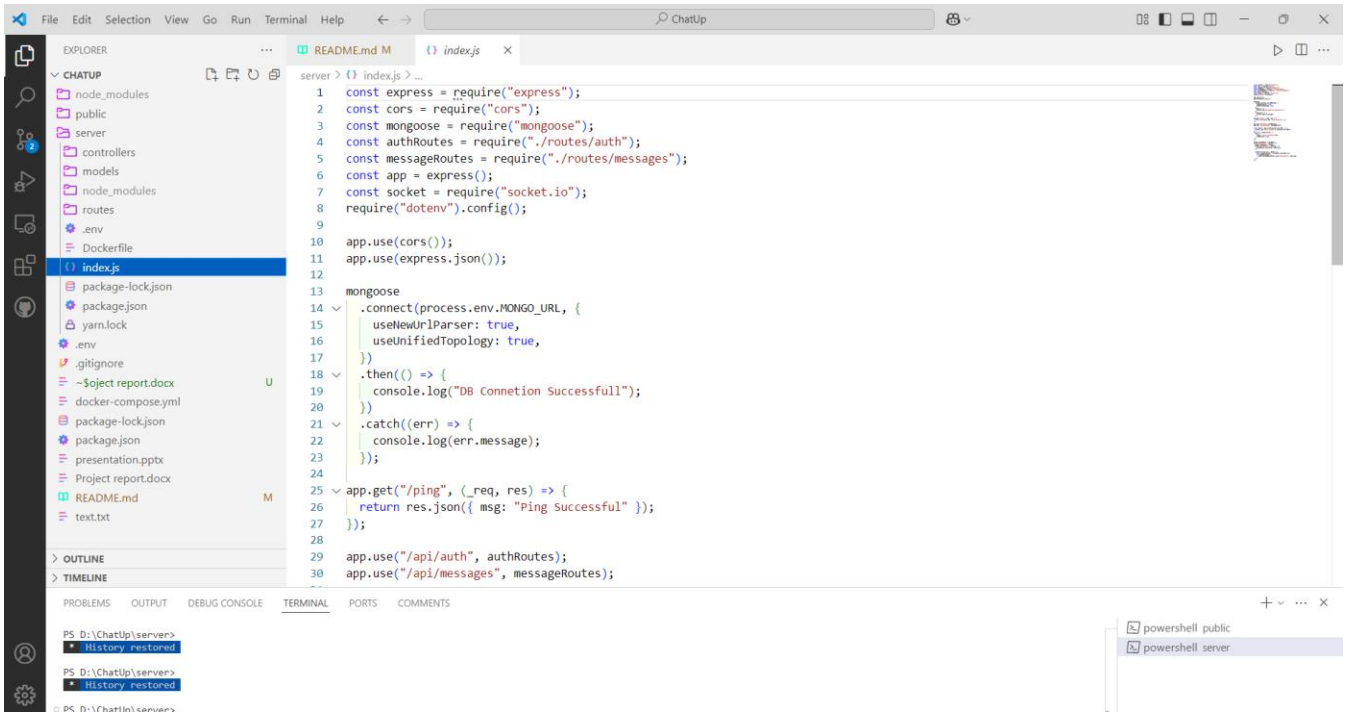




Database Integration

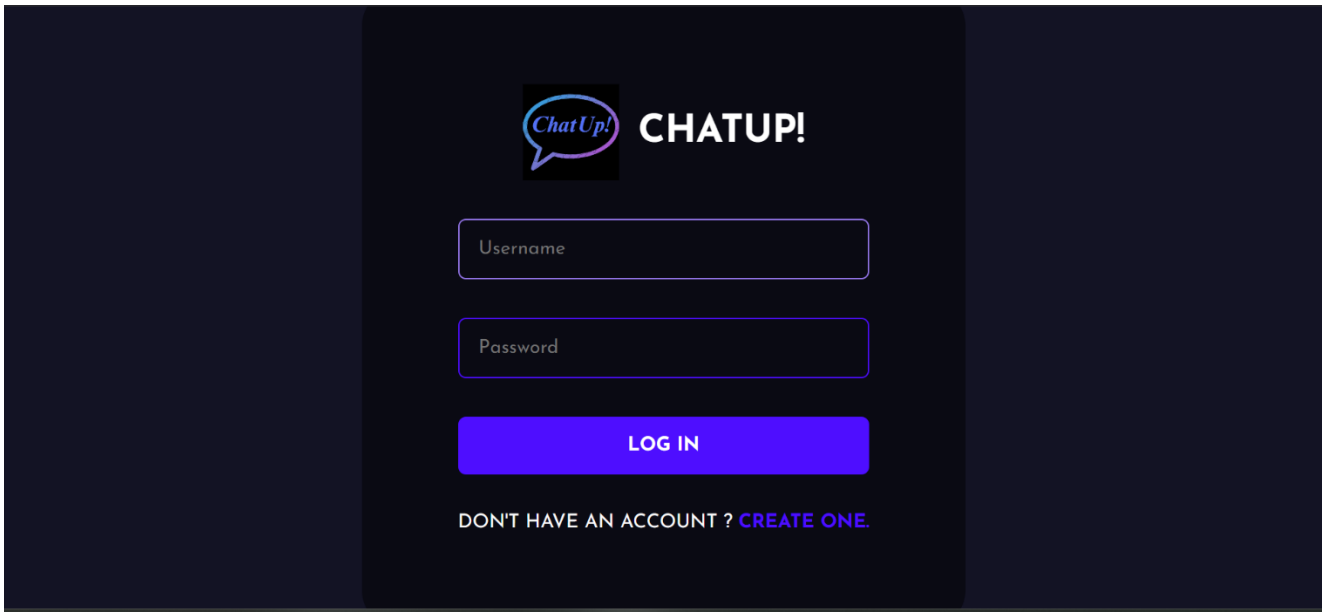
MongoDB is used to store user data and chat history. The application connects to the MongoDB database to save messages and retrieve chat history when users log in.


Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
Databases
Personal-website
admin
chat
messages
users
chat-app
user
users
config
local
users
Databases > chat > users
Open MongoDB shell
Documents 5
Aggregations
Schema
Indexes 3
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 5 of 5
_id: ObjectId('674891aa3e764c1d6caccde3')
username : "asitshakya789"
email : "asitshakya789@gmail.com"
password : "$2b$10$z.2z7h7SyWwpbkgixouoceywYrzTR2NfuCLAeIklC/K18FioeWkue"
isAvatarImageSet : true
avatarImage : "PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMC…"
__v : 0
_id: ObjectId('674891eb3e764c1d6caccde9')
username : "techasit687"
email : "techasit687@gmail.com"
password : "$2b$10$AHXsAOCP.7w/weL9Z8zjHukyge//q3LW6ZtT75O2ybKrj/ACKLY.u"
isAvatarImageSet : true
avatarImage : "PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMC…"
__v : 0
_id: ObjectId('674ca0d45831bfbc7a6d1ebc')
username : "asiy"
email : "sdfrettr4f@gmail.com"
password : "$2b$10$TpnHM4XHVCsbJL6E6QEimOOIJus5IdrjXgoEhzfV6mctmmGiLyr3m"
isAvatarImageSet : true
avatarImage : "PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMC…"
__v : 0
_id: ObjectId('675bbf4deeia3535ca091319')

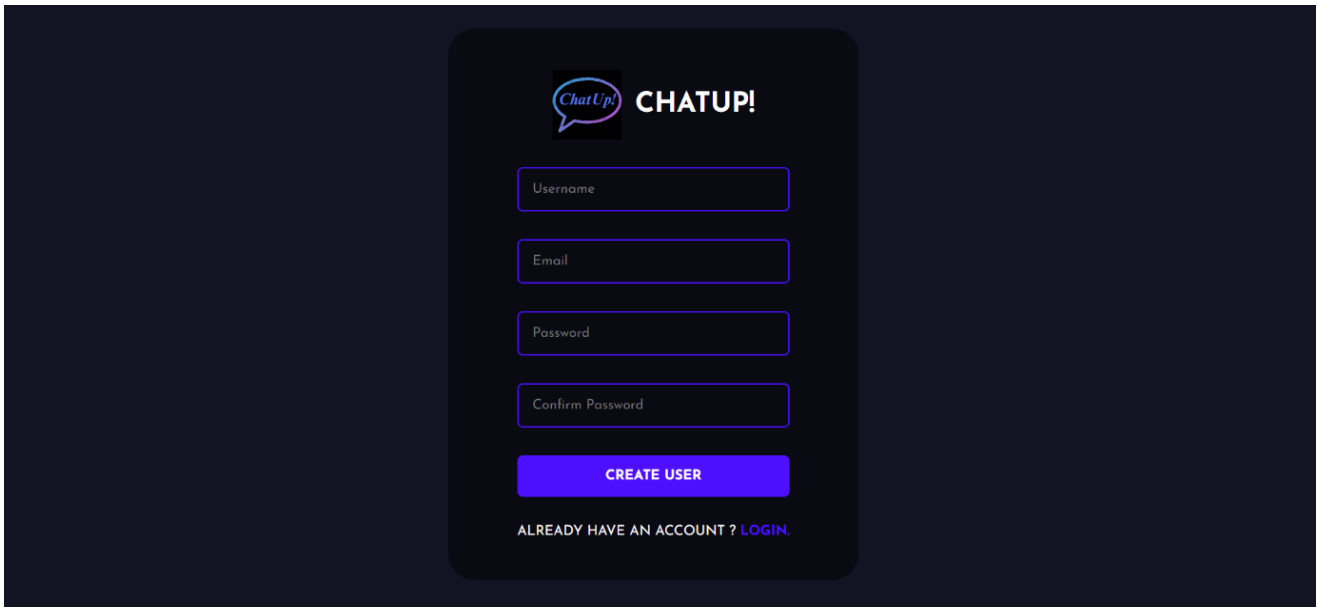

File Edit Selection View Go Run Terminal Help
ChatUp
EXPLORER
CHATUP
node_modules
public
server
controllers
models
node_modules
routes
.env
Dockerfile
index.js
package-lock.json
package.json
yarn.lock
.env
.gitignore
~$oject report.docx
docker-compose.yml
package-lock.json
package.json
presentation.pptx
Project report.docx
README.md
text.txt
OUTLINE
TIMELINE
README.md M
index.js
server > index.js > ...
1 const express = require("express");
2 const cors = require("cors");
3 const mongoose = require("mongoose");
4 const authRoutes = require("./routes/auth");
5 const messageRoutes = require("./routes/messages");
6 const app = express();
7 const socket = require("socket.io");
8 require("dotenv").config();
9
10 app.use(cors());
11 app.use(express.json());
12
13 mongoose
14 .connect(process.env.MONGO_URL, {
15 useNewUrlParser: true,
16 useUnifiedTopology: true,
17 })
18 .then(() => {
19 console.log("DB Connetion Successfull");
20 })
21 .catch((err) => {
22 console.log(err.message);
23 });
24
25 app.get("/ping", (_req, res) => {
26 return res.json({ msg: "Ping Successful" });
27 });
28
29 app.use("/api/auth", authRoutes);
30 app.use("/api/messages", messageRoutes);
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\ChatUp\server>
History restored
PS D:\ChatUp\server>
History restored
powershell public
powershell server

Code Output

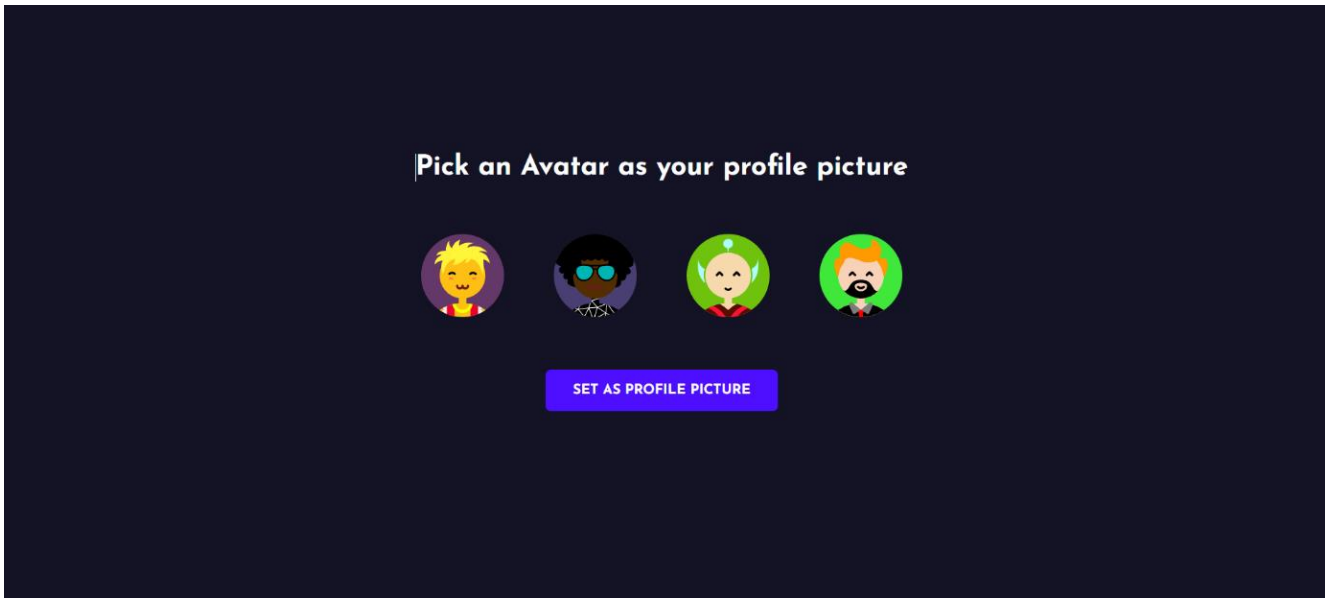
Here is the first page of the project If we already have an account then login using the username and password



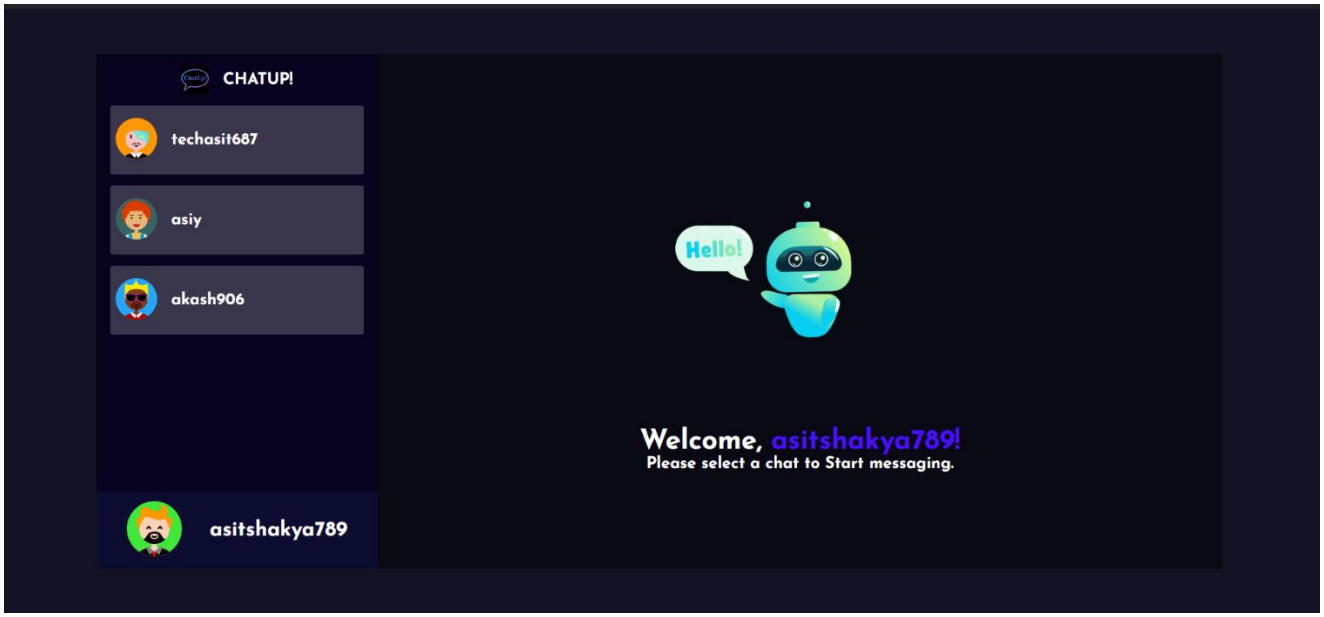
if we are new this application then Create one and fill the basic details then your account was created and then click too the



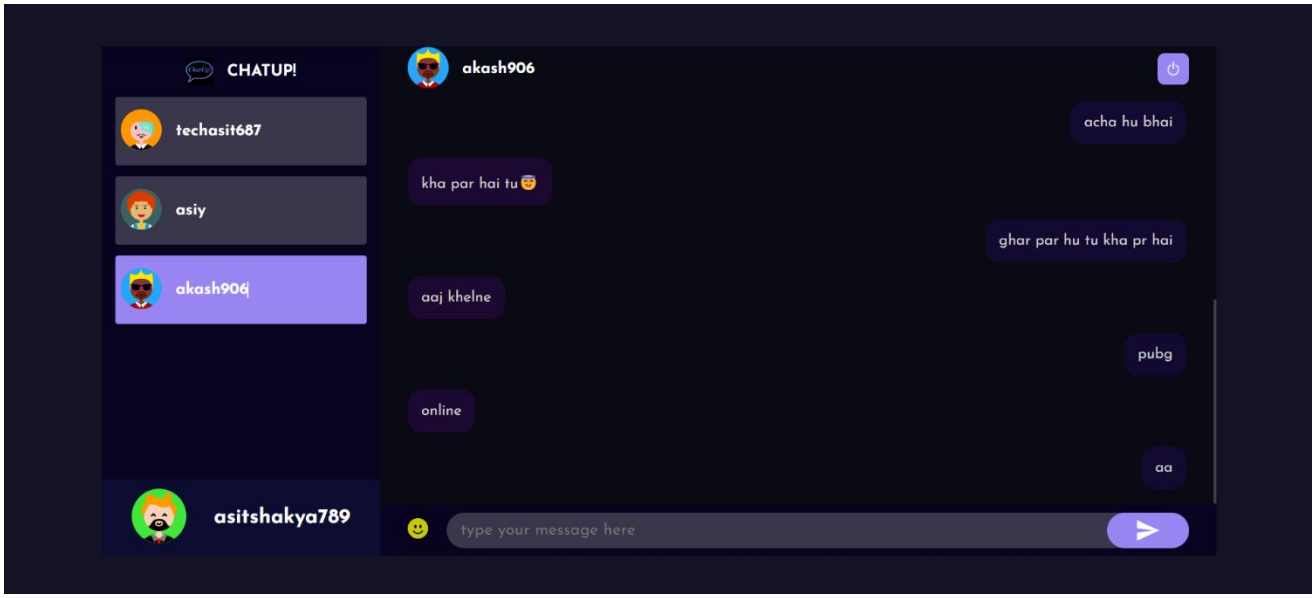
Pick the Avatar for your Suitable Profile picture



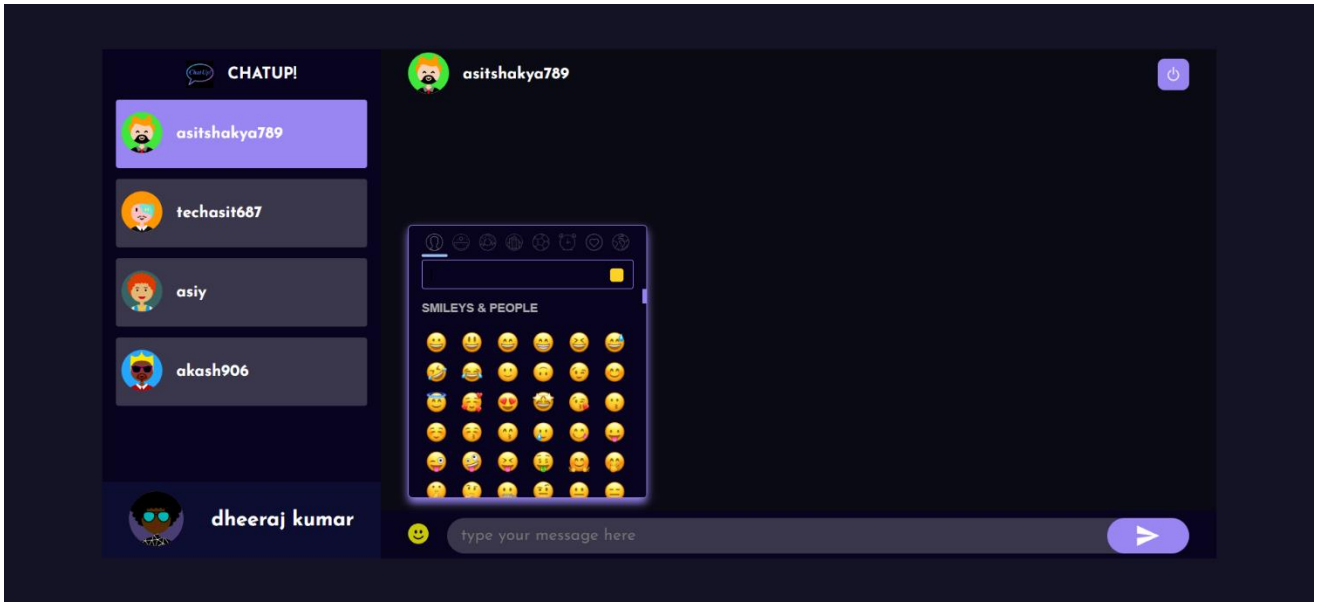
Application well shows all the user that are use this application then you randomly connect the people and start the conversations



Here is the overview of the conversation



Have Feature like we can express our feelings using Emoji



Conclusion

The REAL-TIME CHAT APPLICATIONS project successfully demonstrates the implementation of a real-time chat system using modern web technologies. The application provides a seamless communication experience for users and showcases the capabilities of HTML, CSS, JavaScript, MongoDB, and Socket.IO. This project not only enhances our understanding of web development but also serves as a foundation for future projects in real-time communication.

Future Work

In the future, we plan to enhance the REAL-TIME CHAT APPLICATIONS application by adding features such as:

- **User profiles with customizable settings.**
- **Group chat functionality.**
- **Enhanced security measures, including message encryption.**

Certificate

This is to certify that the Project Report entitled, " **REAL-TIME CHAT APPLICATIONS** " was submitted by Mr. Asit kumar and Mr. Dheeraj Kumar and Mr. Akash Mahato or Saif Ullah under the coordination of Ms. Anvisha Sharma.

The project was carried out during the 5th SEMESTER (2023-24) at Babu Banarasi Das Institute of Technology, Duhai, Ghaziabad affiliated with Dr. APJ Abdul Kalam Technical University, Lucknow.

The work done is the result of the efforts of the project's participants and has not been submitted elsewhere for award of any other degree.

Ms. Anvisha Sharma

Project Coordinator

Department of

Computer Science and Engineering

Babu Banarasi Das Institute of Technology

Duhai, Ghaziabad