

Real Time Chat App using SOCKET.IO

*Mini Project report submitted
in
partial fulfillment of requirement for the award of
degree of*

**Bachelor
in
Engineering**

[Computer Science & Engineering (AI & ML)]

By

| | |
|----------------------|------------|
| Mr.Afan Fondu | RD-22-0526 |
| Mr.Raj Joshi | RD-22-0374 |
| Mr.M.Fuzala Pawaskar | R-21-0190 |
| Mr.Sahil Shivgan | RD-22-0371 |

Guided by

Prof. Mahesh Jadhav

**Department of Computer Science and Engineering (AI & ML)
Finolex Academy of Management and Technology, Ratnagiri**



APRIL, 2024

CERTIFICATE

This is to certify that the following Students have prepared and presented a report on **“Real Time Chat App using SOCKET.IO”** as per partial fulfillment of the term work requirements for the project prescribed by University of Mumbai of Bachelor of Engineering, Year 2023-24.

Prepared by

| | |
|----------------------|------------|
| Mr.Afan Fondu | RD-22-0526 |
| Mr.Raj Joshi | RD-22-0374 |
| Mr.M.Fuzala Pawaskar | R-21-0190 |
| Mr.Sahil Shivgan | RD-22-0371 |

Prof. M. A. Jadhav
(Project Guide)

Prof. V. V. Nimbalkar
H.O.D. (CSE-AIML)

Dr. Kaushal Prasad
Principal

External Examiner

ACKNOWLEDGEMENT

We, hereby sincerely acknowledge those who have imparted their valuable time, energy and intellectual towards the beautification of our project title as “Real Time Chat App using SOCKET.IO”.

It gives us great pleasure to express our sincere, hearty gratitude to our guide, Prof. Mahesh Jadhav Sir Department of Computer Science and Engineering (AI & ML), FAMT, Ratnagiri for his kind interest, inspiring gratitude, valuable advice encouragement and help throughout the project work.

We must thank Prof. V. V. Nimbalkar (H.O.D) Department of Computer Science and Engineering (AI & ML), FAMT, Ratnagiri for her constant encouragement and best support for the completion of this project, others staff members, non-teaching staff of CSE Dept. who have directly and indirectly contributed to our project and encouraging us to bring this project in its present.

| | |
|----------------------|------------|
| Mr.Afan Fondu | RD-22-0526 |
| Mr.Raj Joshi | RD-22-0374 |
| Mr.M.Fuzala Pawaskar | R-21-0190 |
| Mr.Sahil Shivgan | RD-22-0371 |

ABSTRACT

In today's digital age, communication plays a crucial role in connecting individuals across the globe. With the increasing demand for real-time messaging applications, building a single message chat application has become a popular endeavor. This abstract introduces a chat application developed using the MERN stack, which includes Mongo DB, Express, React, NodeJs, SOCKET.IO, Tailwind, BCRYPT technology stack, highlighting its key features and functionality. The proposed chat application provides a robust and scalable solution. MongoDB, a NoSQL database, ensures efficient data storage and retrieval, enabling seamless storage of user profiles and message history. Express.js, a web application framework for Node.js, facilitates the creation of a server-side backend that handles authentication, routing, and interaction with the database. Node.js acts as the run time environment, executing JavaScript code on the server-side, enabling high-performance communication. React.js, a JavaScript library, serves as the frontend framework, providing an interactive and responsive user interface. The chat application encompasses essential features such as user registration and authentication, login & logout, and ability to send and receive messages in real time.

Through this project we aim to demonstrate the feasibility and effectiveness of building real time chat application.

INDEX

| Sr.No | Topics | Page No. |
|--------------|--|-----------------|
| 1 | Introduction | 6 |
| 2 | Literature Survey | 7 |
| 3 | System Features | 8 |
| 4 | System Requirements | 10 |
| 5 | Methodology <ul style="list-style-type: none">➤ DATA FLOW DIAGRAM➤ E-R Diagram➤ Use Case Diagram | 11-14 |
| 6 | Implementation | 15-19 |
| 7 | Advantages | 20 |
| 8 | Disadvantages | 21 |
| 9 | Future Scope | 22-23 |
| 10 | Conclusion | 24-25 |
| 11 | References | 26 |

INTRODUCTION

Real-time communication is crucial in today's digital era, and our application aims to provide a smooth chatting experience. Chat applications have become an integral part of our day-to-day life and have had a significant impact on how we communicate with each other. With numerous chat applications available in the market, each offering unique features and capabilities, users are spoilt for choice. Companies that develop these applications compete with each other to add new features and improve the user experience with each release. However, with the growing concern of data theft, companies must ensure the security of their users' data and protect them from third-party data breaches. To address this, the basic chatting system should involve both sending and receiving processes simultaneously, which can be achieved through the MERN concept. Developers worldwide are constantly striving to enhance the user experience of chat applications and improve their workflow to deliver projects and changes quickly. This is where stacks come into play, which allow developers to build web applications quickly and efficiently. Mern and MERN are two popular stacks built on JavaScript that offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. We had recognized the need for a reliable and user-friendly chat app that could be used across different platforms and devices, and we decided to build it using the Mern stack.

A Real time Messaging App which allows users to create accounts and chat with each other in real-time.

The website is developed using MERN stack, which includes MongoDB, Express, React, and Node.js., Socket.IO is used for real-time communication between the users.

Redux Toolkit is used for state management, and Tailwind CSS is used for UI.

LITERATURE SURVEY

| Year | Title | Author | Publisher | Description |
|------|----------------------------|--|--|---|
| 2023 | Real-Time Chat Application | Ms. Archana Nikose, Sakshi Dosani, Shreya Pardhi, Deep Nikode, Anurag Jais | International Journal of Scientific Research in Science, Engineering and Technology(IJSRSET) | Building a real time chat application involves setting up a server that can handle live communication and developing client side interfaces for users to interact with. The server typically manage connection message broadcasting and maintain the state of chat. |

SYSTEM FEATURES

1. **User Authentication:** Implement user authentication to ensure that only registered users can access the chat application. You can use methods like username/password, OAuth, or biometric authentication depending on your requirements.
2. **Real-time Messaging:** Utilize Socket.IO for real-time messaging capabilities. This includes sending and receiving messages instantly between users connected to the application.
3. **Online Presence Indication:** Displaying the online status of users is crucial for a chat application. Implement features that show when a user is online, offline, or idle.
4. **Group Chat:** Allow users to create or join group chats where multiple participants can communicate simultaneously. Implement features for creating, managing, and leaving group chats.
5. **Private Messaging:** Enable users to initiate private conversations with each other. Implement features for sending private messages and managing private chat sessions.
6. **Message History:** Store chat messages in a database to maintain message history. Users should be able to view past messages when they join a chat room or reopen a conversation.
7. **Typing Indicators:** Implement typing indicators to show when a user is typing a message. This helps to improve the user experience by indicating that someone is actively engaged in the conversation.
8. **Read Receipts:** Provide read receipts to indicate when a message has been delivered and read by the recipient. This enhances communication by confirming that messages have been received and viewed.

9. **Notifications:** Implement push notifications to alert users of new messages or activities in the chat app, even when they are not actively using it. This can be achieved using technologies like Firebase Cloud Messaging (FCM) or Apple Push Notification Service (APNs).
10. **Emojis and Multimedia Support:** Allow users to express themselves using emojis, stickers. Implement features for sending, receiving, and displaying multimedia content in chat messages.
11. **Message Encryption:** Ensure the security of messages transmitted over the network by implementing end-to-end encryption. This prevents unauthorized access to sensitive information and protects user privacy.
12. **User Profile Management:** Enable users to manage their profiles, including updating their display name, profile picture, status message, and other relevant information.
13. **Blocking and Reporting:** Implement features for users to block or report other users in case of harassment or inappropriate behavior. Provide mechanisms for handling reports and enforcing community guidelines.
14. **Cross-platform Compatibility:** Develop the chat application to be compatible with various platforms and devices, including web browsers, mobile devices (iOS and Android), and desktop applications.
15. **Scalability and Performance:** Design the system architecture to be scalable and capable of handling a large number of concurrent users. Optimize performance to minimize latency and ensure a smooth user experience even under heavy load.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

- Processor : Dual-core (2.5 GHz).
- RAM : Minimum 4GB (8GB or more recommended).
- Storage : Minimum 1GB free space.
- Network : Stable internet connection.

SOFTWARE REQUIREMENTS

- Operating System : Windows 10 or higher.
- IDE : Visual Studio Code.
- Node.js : Version 18.1 or higher.
- MongoDB : MongoDB Atlas (cloud).
- Browser : Chrome (Latest version).
- SOCKET.IO

METHODOLOGY

❖ **Defined Project Scope -:**

1. Defined the scope, objectives, and requirements for the project.
2. Established a clear understanding of the project's goals.

❖ **Choose Technology Stack-:**

1. Evaluated options and selected the MERN Stack (MongoDB, Express, React, Node.js) for development.

❖ **Designed Database and UI/UX-:**

1. Designing the database schema using MongoDB for storing user information and chat messages.
2. Created responsive user interfaces to ensure a smooth user experience.

❖ **Developed APIs and Authentication-:**

1. Developed RESTful APIs using Node.js and Express for data manipulation.
2. Implemented user authentication to ensure secure access.

❖ **Added Real-Time Features and Testing-:**

1. Implemented real-time messaging using SOCKET.IO for instant messaging.
2. Conducted comprehensive testing, including unit, integration, and user acceptance testing.

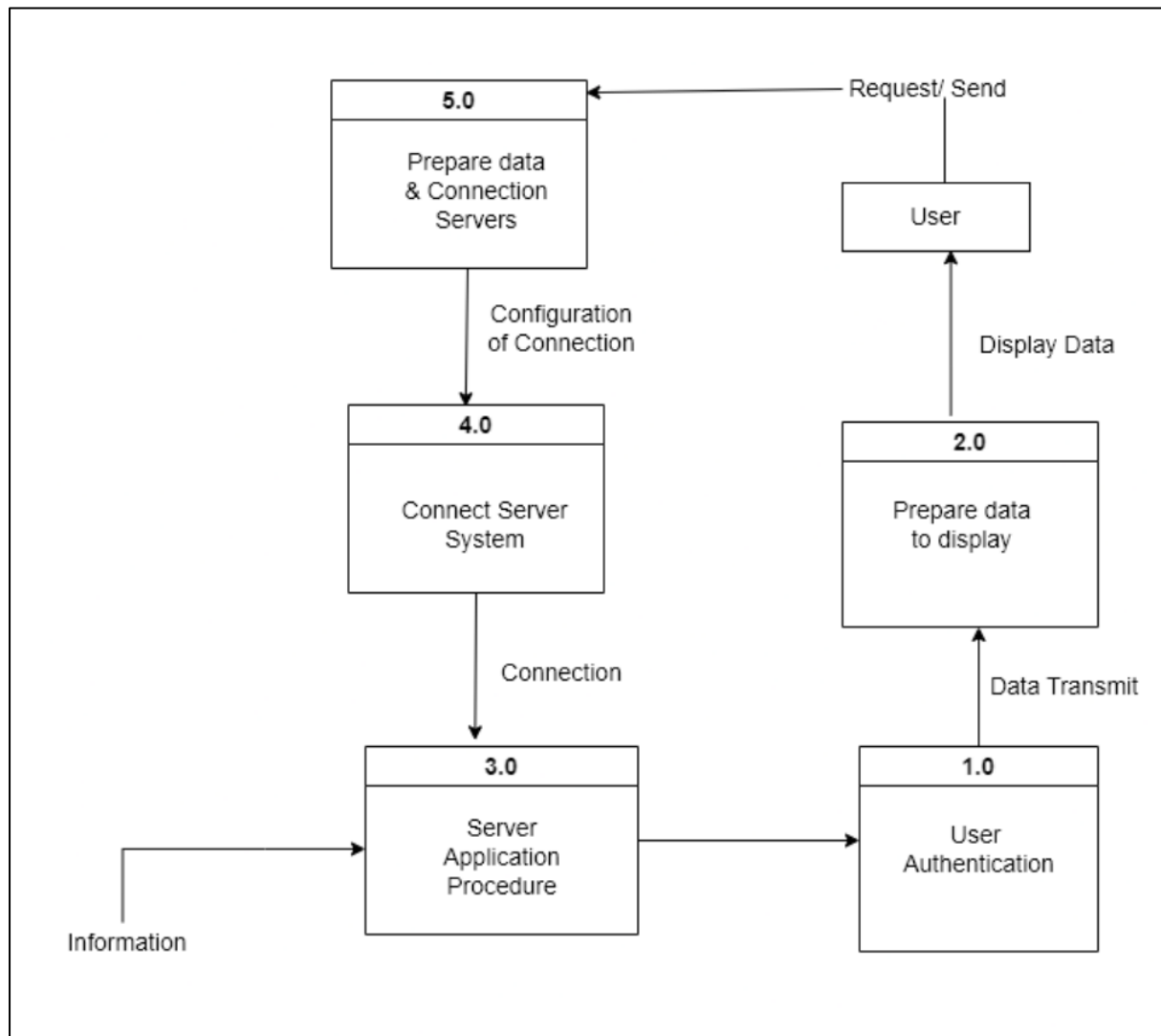
❖ **Deployment and Documentation-:**

1. Deployed the application to a cloud hosting provider.
2. Maintained clear and organized documentation for codebase and configurations.

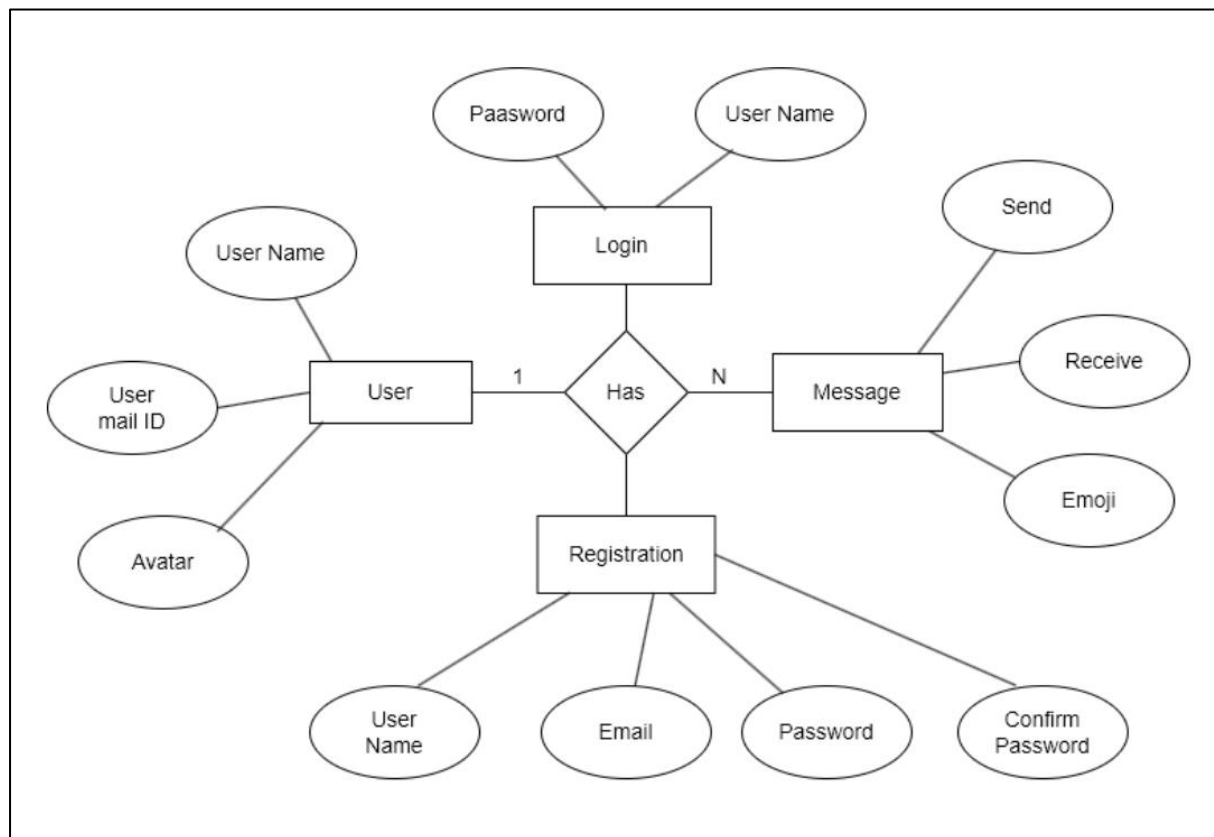
❖ **Website Launch and Ongoing Support-:**

1. Launched the application to the public and ensured accessibility.
2. Provided continuous support and monitored for issues.

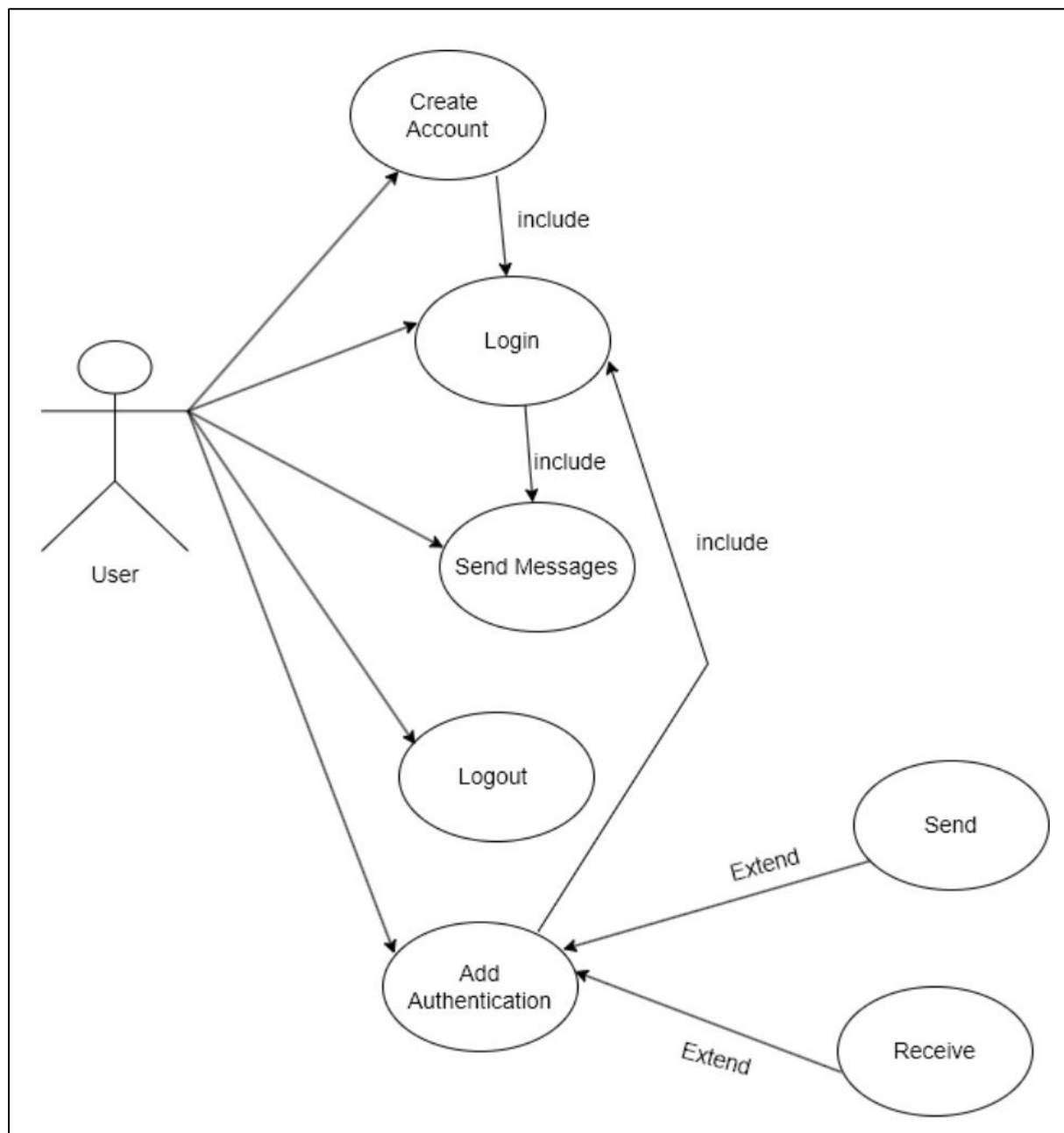
➤ DATA FLOW DIAGRAM



❖ E-R Diagram:



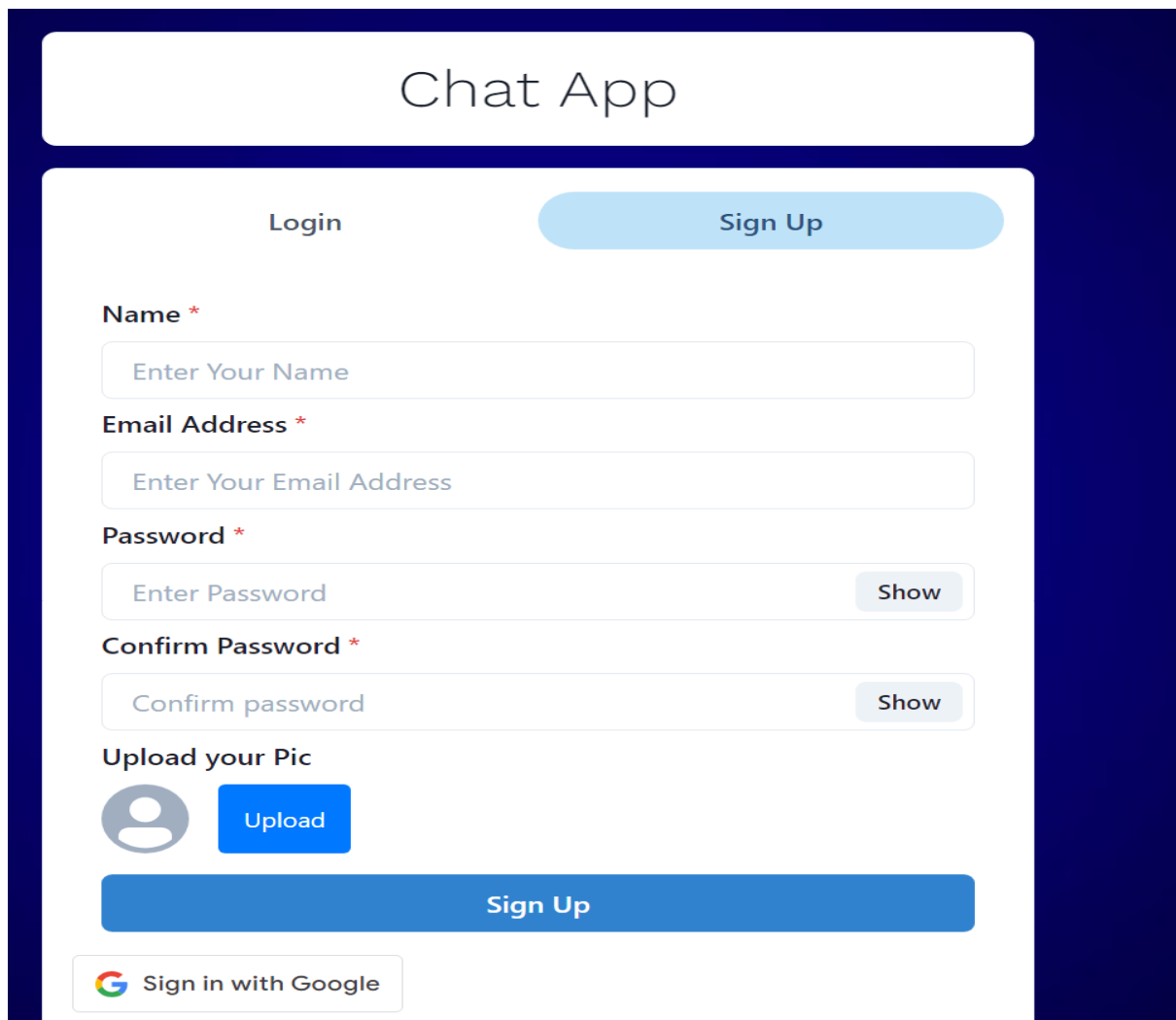
❖ Use Case Diagram :



IMPLEMENTATION

❖ RESULTS (SCREENSHOTS)

1. Sign up



The screenshot displays a web interface for a 'Chat App' with a dark blue header and sidebars. The main content area is white and contains a sign-up form. At the top of the form, there are two tabs: 'Login' and 'Sign Up', with 'Sign Up' being the active tab. The form includes several input fields: 'Name *', 'Email Address *', 'Password *', and 'Confirm Password *'. Each field has a placeholder text and a 'Show' button next to it. Below these fields is an 'Upload your Pic' section with a user icon and an 'Upload' button. A large blue 'Sign Up' button is positioned below the form fields. At the bottom of the form, there is a 'Sign in with Google' button featuring the Google logo.

Chat App

Login Sign Up

Name *

Enter Your Name

Email Address *

Enter Your Email Address

Password *

Enter Password Show

Confirm Password *

Confirm password Show

Upload your Pic

Upload

Sign Up

Sign in with Google

2. Log in

Chat App

Login

Sign Up

Email Address *


Enter Your Email Address

Password *


Enter password

Show

Login


 Sign in with Google


3. Google sign in

 Sign in with Google

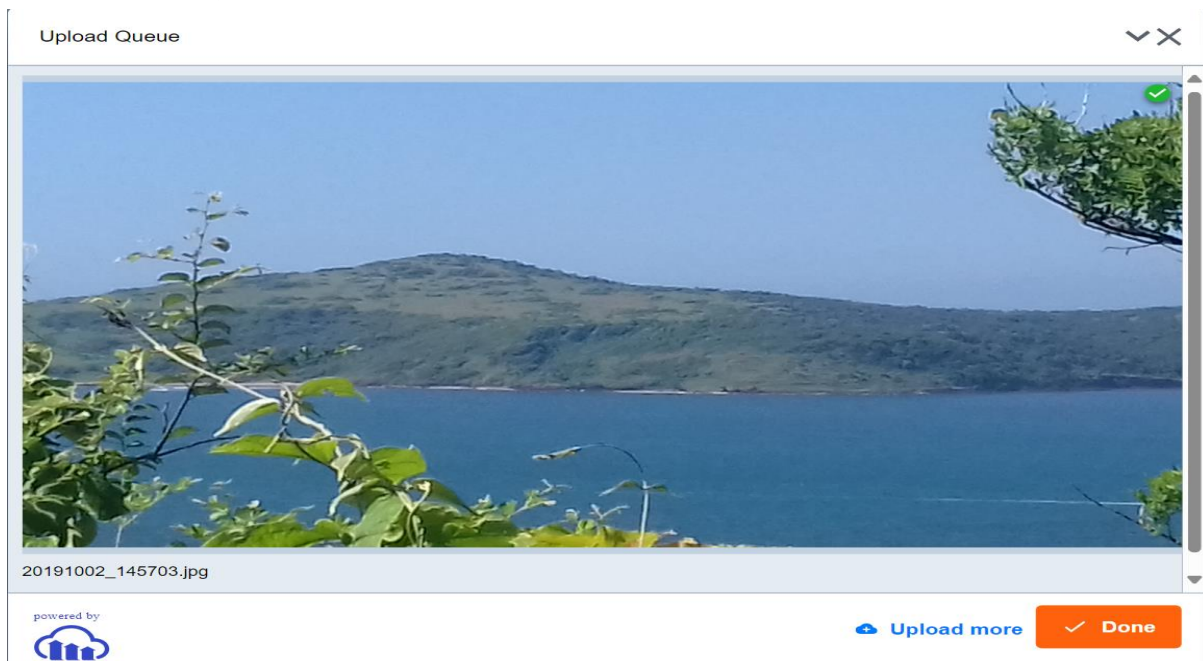
Choose an Account

to continue to mern-chat-app-clg.vercel.app

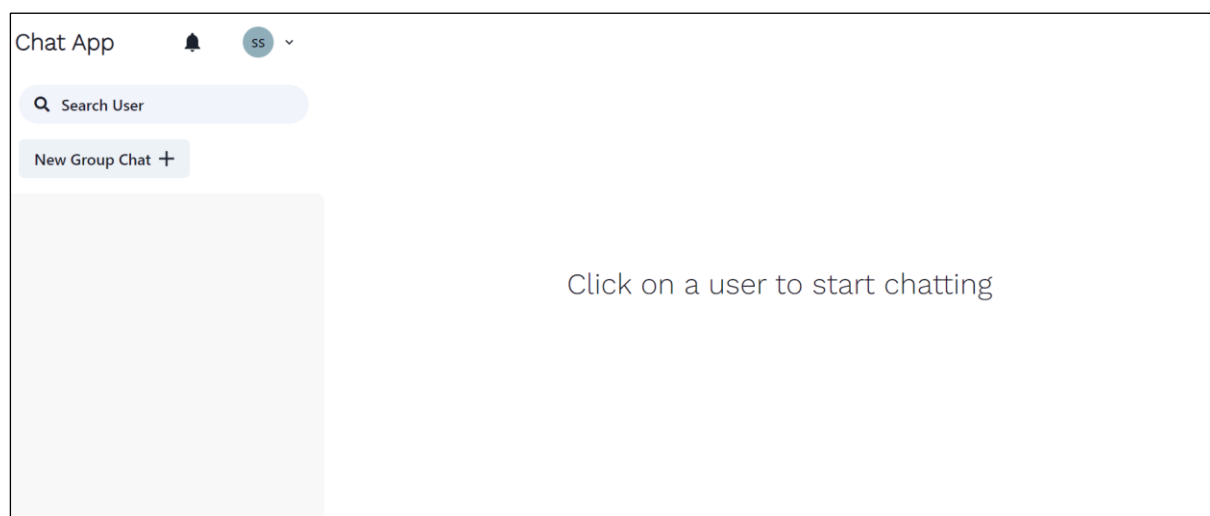
 Sahil shivgan
sahilshivgan25@gmail.com

 Use another account

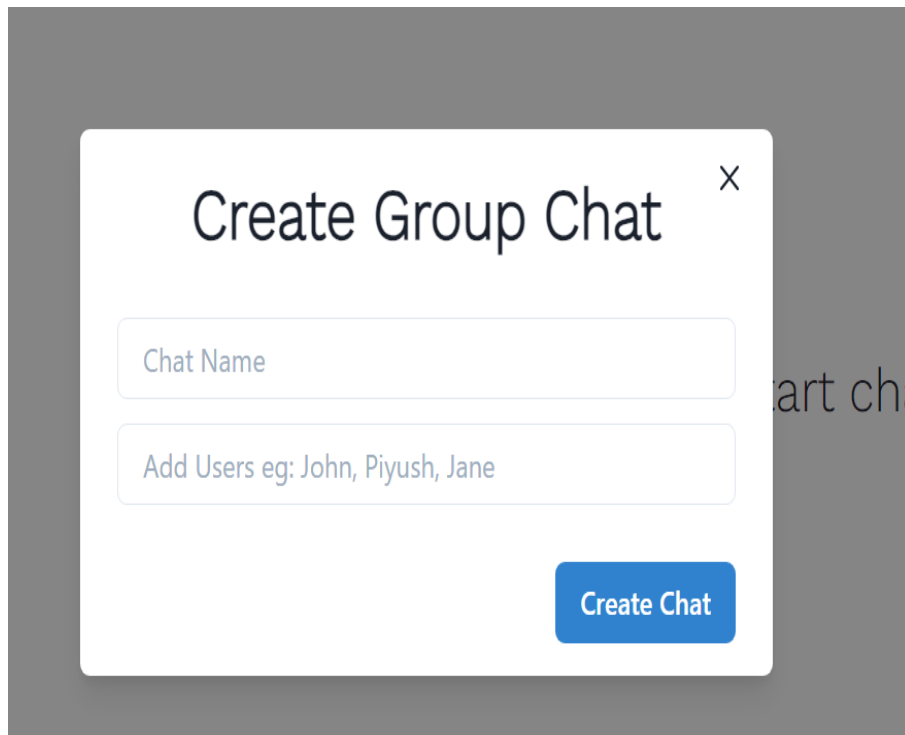
4. Profile photo



5. Messaging

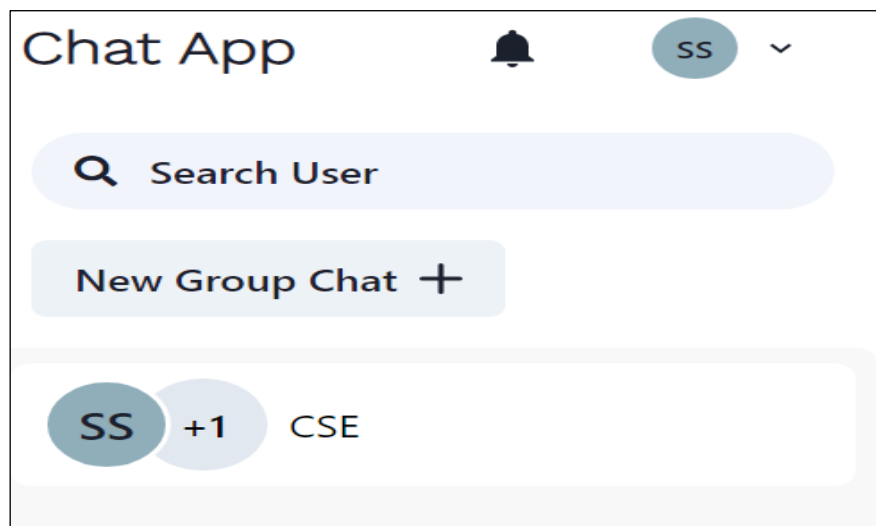


6. Group Chat



A modal dialog box titled "Create Group Chat" with a close button (X) in the top right corner. It contains two text input fields: the first is labeled "Chat Name" and the second is labeled "Add Users eg: John, Piyush, Jane". A blue button labeled "Create Chat" is positioned at the bottom right of the dialog.

7. User added into group



A screenshot of a "Chat App" interface. At the top, the title "Chat App" is on the left, a bell icon for notifications is in the center, and a user profile icon labeled "SS" with a dropdown arrow is on the right. Below the title bar, there is a search bar with a magnifying glass icon and the text "Search User". Underneath the search bar is a button labeled "New Group Chat" with a plus icon. At the bottom, a chat list item is shown with a circular profile icon labeled "SS", a grey circle containing "+1", and the text "CSE" to its right.

8. User remove from group

CSE

RAJ JOSHI ×

AFAN FONDU ×

JANE DOE ×

SAHIL SHIVGAN ×

Chat Name

Update

Add User to group

Leave Group

9. Emojis

Chat App

SS

SS

CSE

Search User

New Group Chat +

SS

+1

CSE

😊 😊 😊

ADVANTAGES

- ❖ Real-time Communication
- ❖ Low Latency
- ❖ Efficient Resource Usage
- ❖ Cross-platform Compatibility
- ❖ Scalability
- ❖ Ease of Implementation
- ❖ Customization
- ❖ Community Support
- ❖ Reliability
- ❖ Security

DISADVANTAGES

- ❖ Complexity
- ❖ Resource Consumption
- ❖ Network Overhead
- ❖ Compatibility Issues
- ❖ Security Considerations
- ❖ Performance Tuning
- ❖ Debugging and Troubleshooting
- ❖ Dependency Management
- ❖ Learning Curve

FUTURE SCOPE

❖ **Enhanced User Experience:**

Future chat applications will likely focus on providing even more immersive and engaging user experiences. This could include features such as augmented reality (AR) chat environments, voice-based interactions, and integration with virtual assistants or chatbots for natural language processing.

❖ **AI and Automation:**

Integration of artificial intelligence (AI) and machine learning (ML) technologies will play a significant role in the future of chat applications. AI-powered chatbots can assist users with tasks, provide personalized recommendations, and automate routine interactions, making chat experiences more efficient and seamless.

❖ **Interoperability and Integration:**

Chat applications may become more interconnected with other platforms and services, enabling seamless communication and data sharing across different applications and ecosystems. Integration with social media platforms, productivity tools, and smart home devices could further enhance the utility and versatility of chat applications.

❖ **Privacy and Security:**

As concerns about privacy and security continue to grow, future chat applications will prioritize robust encryption, data protection, and user privacy features. Technologies such as end-to-end encryption, decentralized architectures, and zero-knowledge proof protocols may become standard practices to ensure the confidentiality and integrity of communications.

❖ **Multi-platform Support:**

With the proliferation of devices and platforms, future chat applications will need to offer seamless experiences across a wide range of devices, including smartphones, tablets, smartwatches, smart TVs.

❖ **Customization and Personalization:**

Future chat applications may offer more customization and personalization options, allowing users to tailor their chat experiences to their preferences and needs. This could include customizable chat themes, personalized chatbots, and advanced settings for message filtering and notification management.

❖ **Collaboration and Productivity:**

Chat applications will increasingly serve as collaboration tools for teams and organizations, facilitating real-time communication, project management, and workflow automation. Integration with productivity tools such as task managers, document editors, and calendar apps will streamline collaboration and enhance productivity.

❖ **Voice and Video Communication:**

While text-based messaging remains predominant, future chat applications may place greater emphasis on voice and video communication features. This could include high-quality voice and video calling, real-time video conferencing, and immersive virtual meeting environments.

❖ **Accessibility and Inclusivity:**

Future chat applications will aim to be more accessible and inclusive, catering to users with diverse needs and preferences. This could involve implementing features such as screen reader support, text-to-speech capabilities, language translation, and inclusive design principles.

❖ **Social and Community Engagement:**

Chat applications will continue to serve as platforms for social interaction, community building, and content sharing. Future developments may focus on fostering deeper connections, facilitating online communities, and promoting positive social interactions while combating harassment and abuse.

CONCLUSION

In conclusion, our project real-time chat applications using technologies like Socket.IO offer a dynamic and engaging platform for instant communication and collaboration. With the ability to transmit messages instantly and maintain persistent connections between users, these applications provide a seamless and responsive user experience across various platforms and devices.

Despite potential challenges such as complexity, resource consumption, and security considerations, the advantages of Socket.IO in terms of low latency, cross-platform compatibility, scalability, and customization make it a popular choice for building real-time chat solutions.

Looking ahead, the future of real-time chat applications holds tremendous promise, driven by advancements in AI, interoperability, privacy, multi-platform support, customization, collaboration, voice and video communication, accessibility, and social engagement. By embracing these trends and leveraging emerging technologies, developers can create chat experiences that are not only functional and efficient but also personalized, secure, inclusive, and socially impactful.

As the demand for real-time communication continues to grow, chat applications will remain a vital tool for connecting people, fostering collaboration, and building communities in an increasingly digital world. By innovating and evolving to meet the evolving needs and preferences of users, real-time chat applications will continue to play a central role in shaping the future of communication and collaboration.

The MERN stack offers a comprehensive solution for developing a chat application that supports real-time communication. By leveraging web-sockets or similar technologies, users can instantly exchange messages. The MERN stack supports the creation of cross-platform applications, allowing users to access the chat application from various devices and operating systems. Using JavaScript throughout the entire stack provides a developer-friendly environment. It reduces the learning curve, promotes code consistency, and facilitates easier code maintenance. The MERN stack benefits from a large and active community. This means developers can find extensive resources,

tutorials, and libraries to assist in the development process and troubleshoot any issues. While considering the advantages of the MERN stack, it's crucial to take into account the specific requirements and constraints of your project. Proper planning, architectural considerations, and adherence to best practices are essential for ensuring a secure, efficient, and user-friendly single message chat application. MERN stack provides a unified and consistent development experience since JavaScript is used throughout the entire stack. This reduces the learning curve for developers and allows for easier code maintenance.

REFERENCES

- [1] "Learning Node.js Development" by Andrew Mead and Mike Cantelon: This book covers Node.js development, including Socket.IO for real-time communication.
- [2] "Node.js Design Patterns" by Mario Casciaro: Although not solely focused on Socket.IO, this book provides insights into Node.js design patterns, which can be beneficial when structuring your Socket.IO application.
- [3] <https://socket.io/get-started/chat/>
- [4] <https://www.youtube.com/watch?v=tHbCkikFfDE>
- [5] <https://github.com/socketio/socket.io>
- [6] <https://doi.org/10.32628/IJSRSET2310267>