



सिद्धिर्भवति कर्मजा
www.famt.ac.in



Finolex Academy of Management and Technology

Topic : Real Time Chat App using SOCKET.IO

Mini Project presentation submitted
in
partial fulfilment of the requirements for the degree
of
Bachelor of Engineering
in
Computer Science and Engineering (AI & ML)

Submitted By :

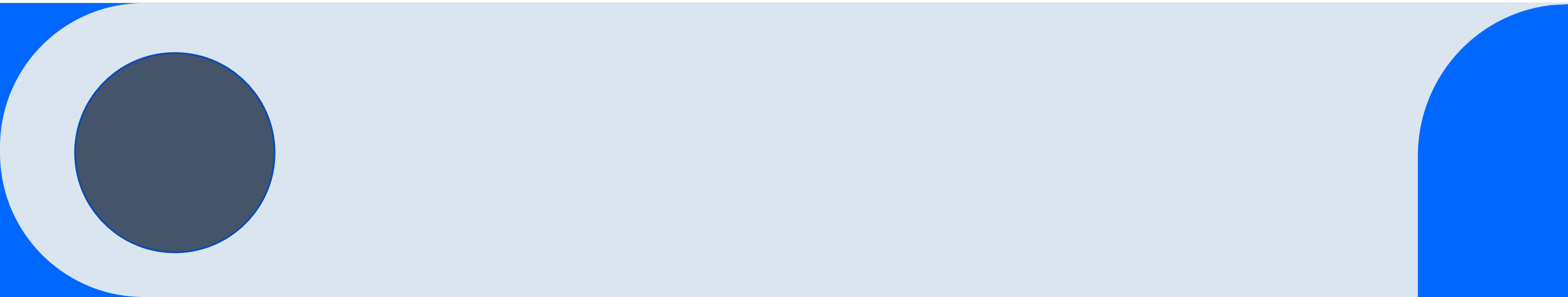
| | |
|-------------------|------------|
| Afan Fondu | RD-22-0526 |
| Raj Joshi | RD-22-0374 |
| M.Fuzala Pawaskar | R-21-0190 |
| Sahil Shivgan | RD-22-0371 |

Guided By :

Prof. Mahesh Jadhav
Department of Computer Science
and Engineering (AI & ML)

MARCH, 2024

Real Time Chat App using SOCKET.IO



Acknowledgement

We hereby sincerely acknowledge those who have dedicated their valuable time, energy, and intellect to the enhancement of our project titled “Real Time Chat App using SOCKET.IO.”

It brings us immense pleasure to convey our sincere and heartfelt gratitude to our guide, Prof. Mahesh Jadhav, Department of Computer Science and Engineering (AIML), FAMT, Ratnagiri, for his keen interest, inspiring guidance, valuable advice, encouragement, and assistance throughout the project.

We are indebted to Prof. V. V. Nimbalkar (H.O.D), Department of Computer Science and Engineering (AIML), FAMT, Ratnagiri, for her unwavering encouragement and unparalleled support in bringing this project to fruition. Our gratitude also extends to other faculty members, non-teaching staff of the CSE Dept., who have directly and indirectly contributed to our project and have motivated us to present it in its current form.



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. |

Introduction

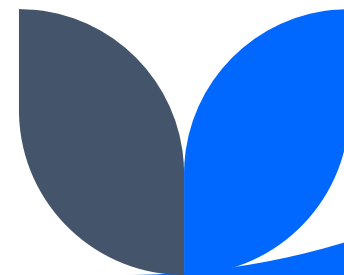
Real-time communication is crucial in today's digital era, and our application aims to provide a smooth chatting experience.

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.



Problem Statement

Real-Time Updates: Chat apps often have delays, disrupting conversations. (Gmail)

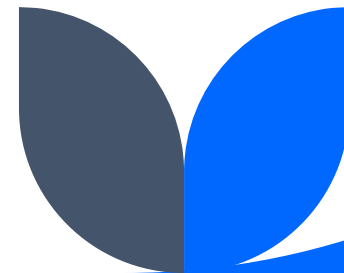
Scalability Challenges: Growing user bases strain traditional apps, causing crashes. (WhatsApp)

Security Concerns: Many apps lack strong encryption, risking user data. (Telegram)

Limited Functionality: Some apps lack multimedia sharing or group chat features. (Traditional SMS/MMS)

Cross-Platform Compatibility: Users face issues using apps across different devices. (Apple iMessage)

High Latency: Slow response times hinder real-time communication. (Facebook)



Architecture

1. High-Level Overview –

Our Chat app architecture leverages the MERN Stack – React for frontend, Node.js/Express.js for backend, and MongoDB for data storage. SOCKET.IO for instant real-time chatting.

2. Components –

- **Frontend (Client):** Built using React for user interface.
- **Backend (Server):** Developed with Node.js and Express.js for handling requests.
- **Database:** MongoDB stores product, user, and order data.

3. Communication –

- Frontend → Backend: API requests for data and functionality.
- Backend ↔ Database: Communication for data retrieval and storage.

4. Deployment –

- Deployed and Hosted on cloud server i.e., Vercel.



Architecture contd.

5. Security –

Implement authentication and authorization mechanisms for secure user interactions.

6. Scalability –

Architecture designed to handle increased traffic and users effectively.

7. Technologies –

React for UI, Node.js/Express.js for server logic, MongoDB for data storage.

8. User Experience –

Ensure a consistent user experience across modules.

Objectives

- Develop a real-time chat application using SOCKET.IO for instant messaging.
- Provide users with a seamless chatting experience with real-time updates.
- Ensure scalability and reliability of the application to handle multiple users simultaneously.
- User registration and authentication.
- Group chats functionality.
- Notification system for new messages.

Hardware and Software Requirements

Hardware Requirements

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

Software Requirements

1. Operating System : Windows 10 or higher.
2. IDE : Visual Studio Code.
3. Node.js : Version 18.1 or higher.
4. MongoDB : MongoDB Atlas (cloud).
5. Browser : Chrome (Latest version).
6. SOCKET.IO

Methodology

Step 1: Defined Project Scope

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

Step 2: Chose Technology Stack

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

Step 3: 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.

Step 4: Developed APIs and Authentication

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

Methodology contd.

Step 5: 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.

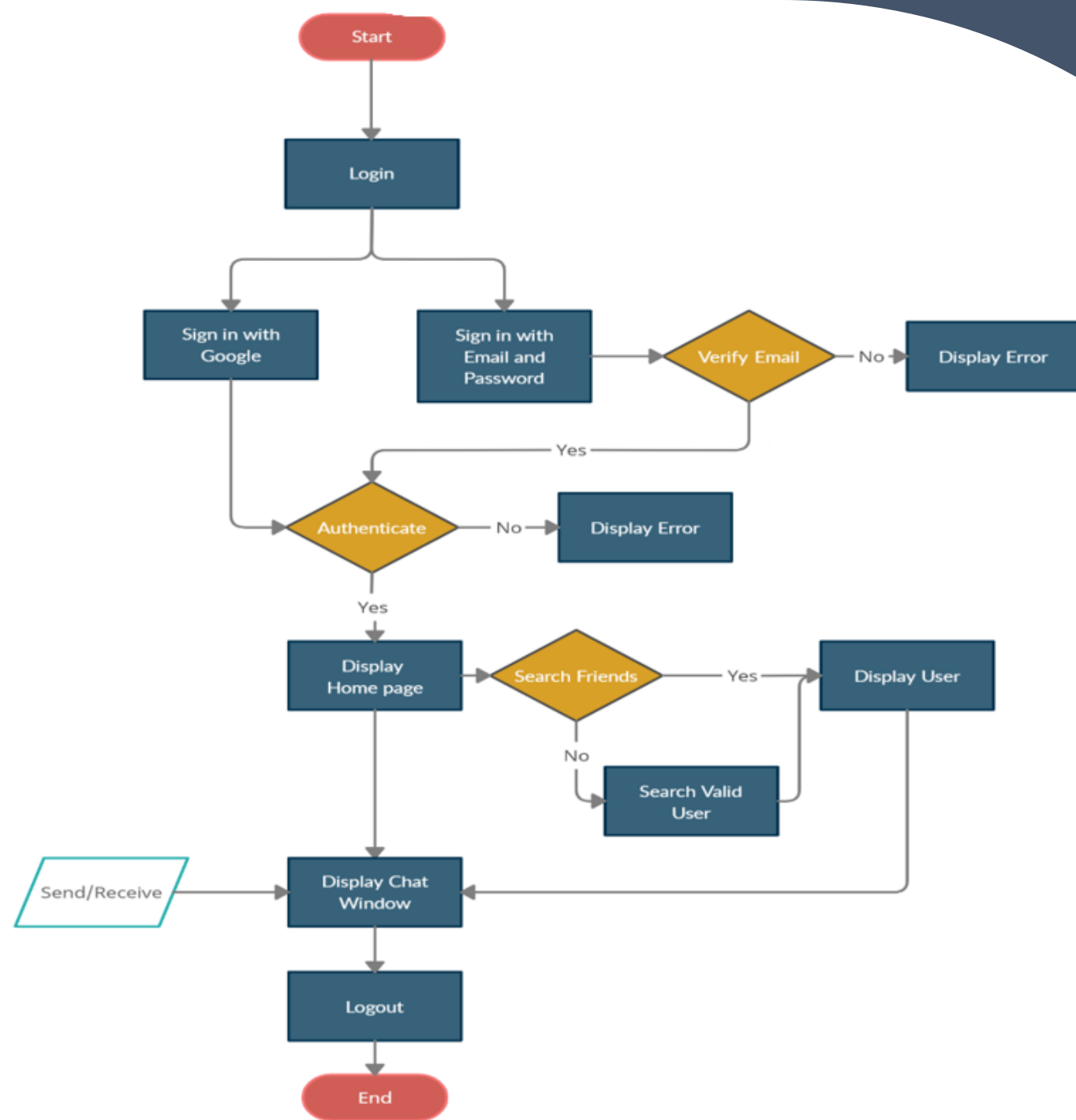
Step 6: Deployment and Documentation

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

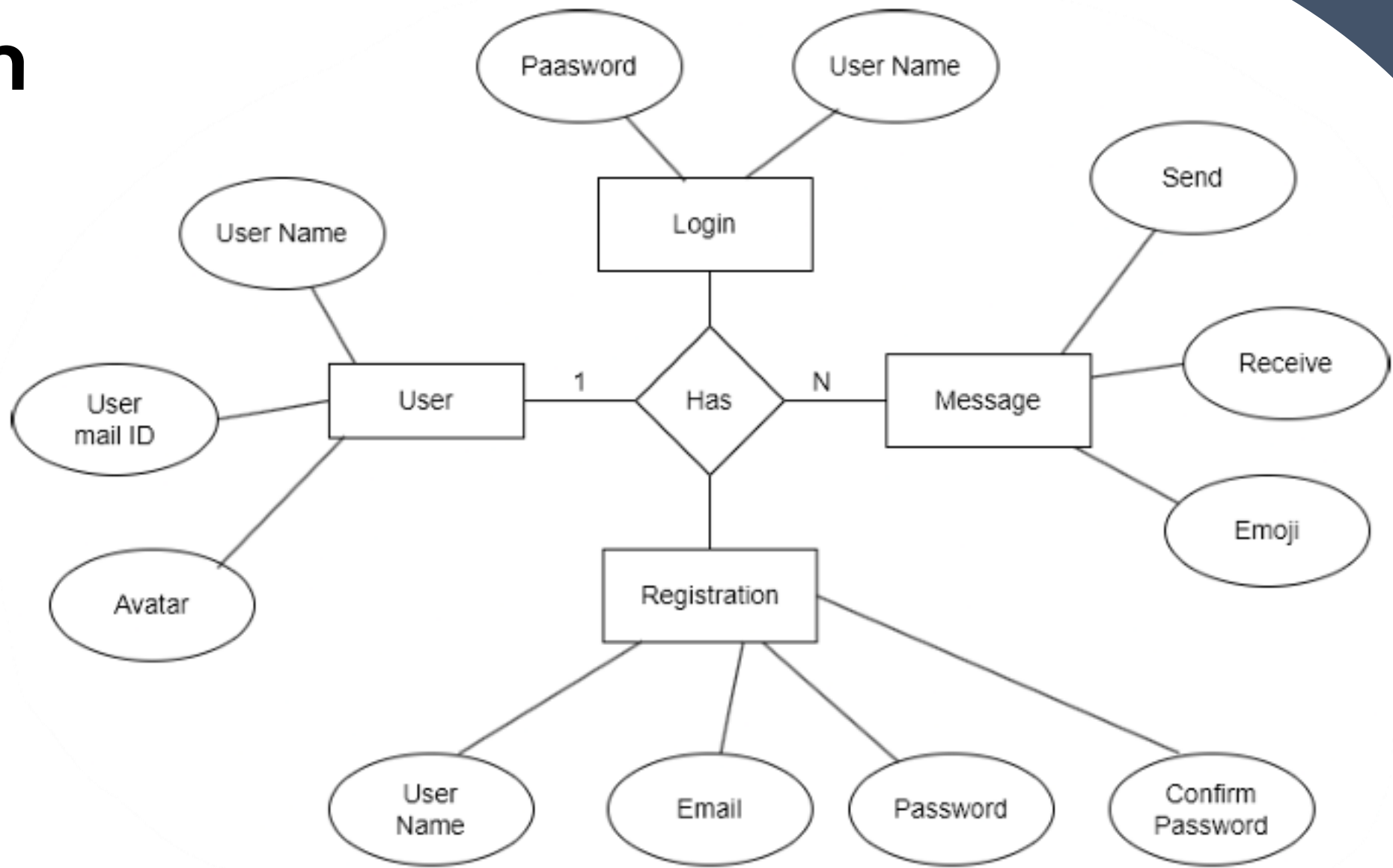
Step 7: Website Launch and Ongoing Support

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

Work Flow Diagram for Users



E-R Diagram



Packages Used

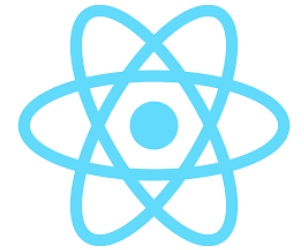
1. Mongo DB



2. Express



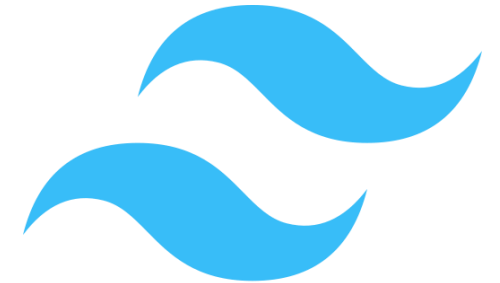
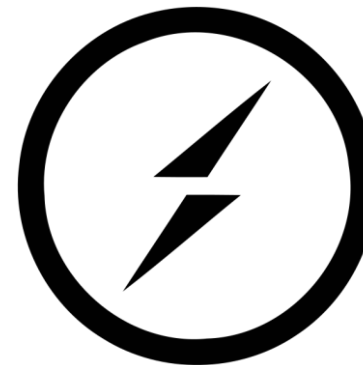
3. React



4. NodeJs



5. SOCKET.IO



6. Tailwind



7. BCRIPT

Challenges Faced

There were various technical challenges faced by us while developing this web application. Some major of them are listed below:

1. **Database Design and Modeling:** Designing an efficient and scalable database schema that meets the requirements of an Chat App platform was complex.
2. **Front-End User Experience:** Designing an intuitive and responsive user interface that provides a seamless chatting experience required careful consideration of design principles.
3. **Real-Time Updates:** Implementing real-time updates for user chats and groups required understanding Socket's and managing real-time data flow.
4. **Testing and Debugging:** Identifying and resolving bugs across the entire stack, from front-end to back-end, while maintaining a consistent user experience was time-consuming.
5. **Version Control and Collaboration:** Coordinating the work of a development team using version control systems and resolving merge conflicts were ongoing challenges.



Key Feature For End User

1. Login Page

Chat App

Login

Sign Up

Email Address *


Enter Your Email Address

Password *

Enter password

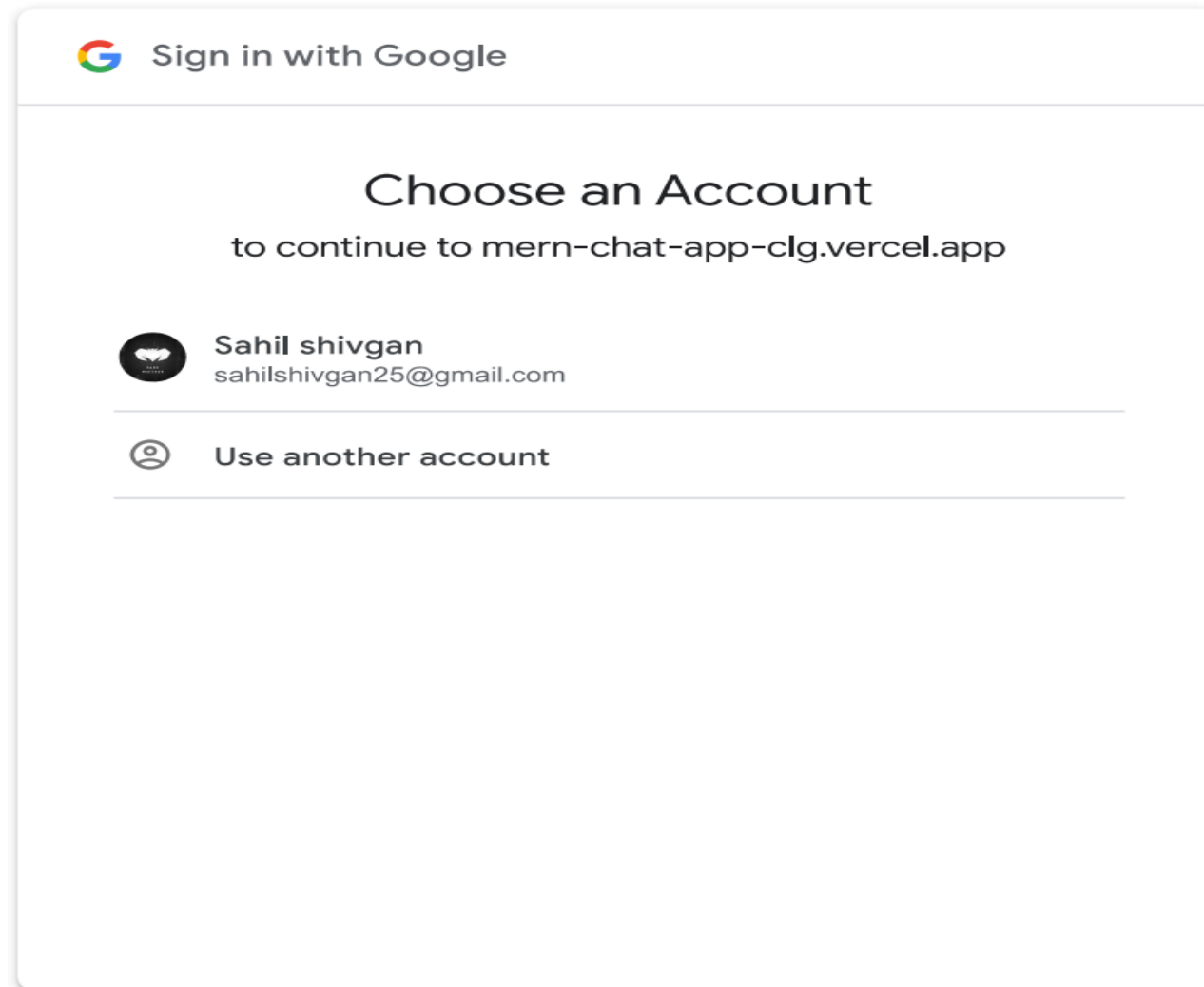
Show

Login

 Sign in with Google

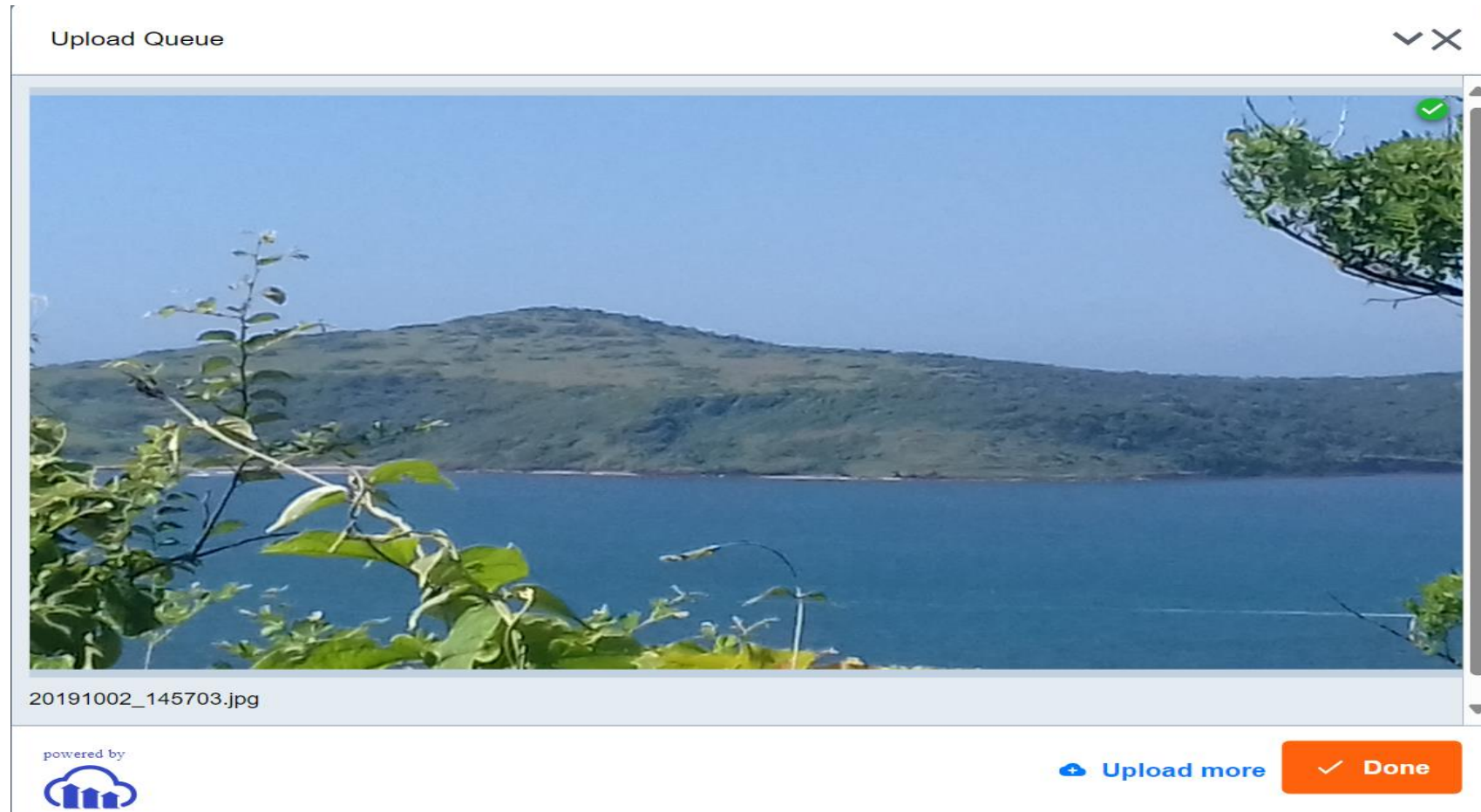
Key Feature For End User

2. Google Sign In



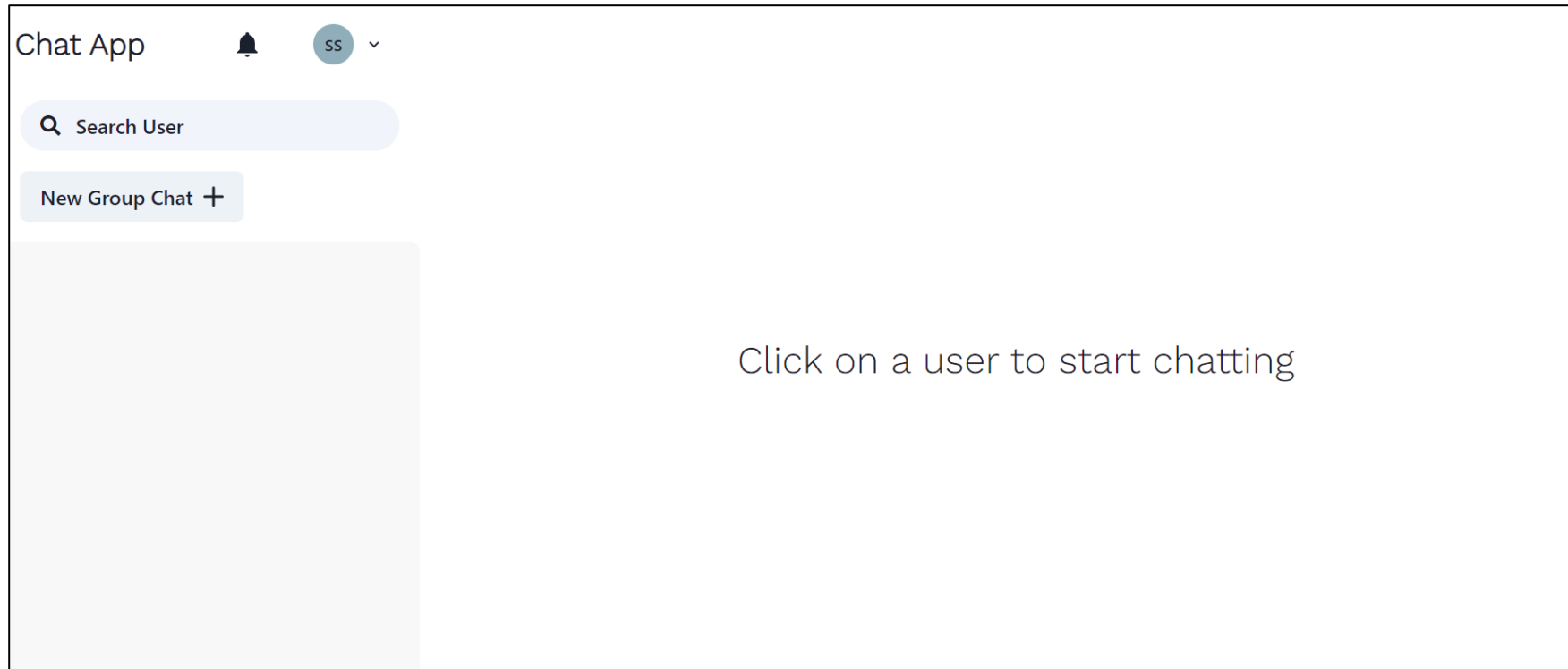
Key Feature For End User

3. Profile Photo



Key Feature For End User

4. Dashboard



Key Feature For End User

5. Group Chat

CSE

RAJ JOSHI ×

AFAN FONDU ×

JANE DOE ×

SAHIL SHIVGAN ×

Chat Name

Update

Add User to group

Leave Group

Key Feature For End User

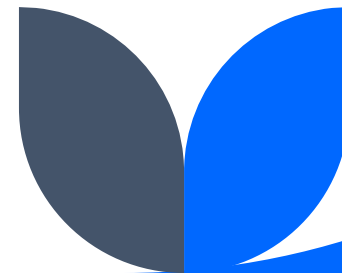
6. Emojis Supported



Future Scope

In the future, we envision enhancing the user experience and expanding the functionalities of the chat app. Some of the potential areas for future development include:

- 1.Enhanced User Experience:** Integrate multimedia support (images, videos, files) to enrich conversations and engage users more effectively.
- 2.Advanced Security Features:** Implement end-to-end encryption to ensure user privacy and security of sensitive data exchanged within the chat application.
- 3.AI-Powered Chatbot Integration:** Incorporate natural language processing (NLP) algorithms to create intelligent chatbots capable of assisting users with frequently asked questions and tasks.
- 4.Real-Time Translation:** Integrate machine translation services to enable real-time language translation, facilitating communication between users from different linguistic backgrounds.



Conclusion

In conclusion, building a single-message chat application using the MERN stack (MongoDB, Express.js, React.js, Node.js) offers numerous benefits. Leveraging web sockets, users can exchange messages instantly. The stack supports cross-platform functionality and is developer-friendly, with JavaScript throughout, reducing learning curves and facilitating maintenance. Additionally, the active community provides ample resources for development and troubleshooting. However, careful planning and adherence to best practices are vital for security and efficiency.

THANK

YOU