Chat Application Project Report

Introduction

This project is a real-time chat application built to facilitate instant messaging between users. It supports private and group chats, message status updates (sent/delivered/seen), and user authentication. The application uses MERN stack for frontend and backend technologies to deliver a seamless user experience.

Abstract

The chat application enables users to communicate in real-time using web sockets. It features user registration, login, group chat creation, message delivery status, and emoji support. The system is designed with a MVC architecture, separating frontend and backend concerns, and leverages REST APIs and Socket.IO for efficient data exchange and event handling.

Tools Used

- **Frontend:** React.js, Vite, Redux Toolkit, RTK Query, Socket.IO Client, Tailwind CSS, Emoji Mart
- **Backend:** Node.js, Express.js, MongoDB, Mongoose, Socket.IO, bcrypt, JWT
- **Other:** Postman (API testing), Visual Studio Code (IDE), Git (version control)

Steps Involved in Building the Project

1. **Project Initialization**
    - Set up separate frontend and backend folders.
    - Initialized Node.js and React projects with necessary dependencies.
2. **Backend Development**
    - Designed MongoDB schemas for users, chats, and messages using Mongoose.
    - Implemented RESTful APIs for user authentication, chat management, and message handling.
    - Integrated Socket.IO for real-time communication.
    - Secured endpoints with JWT authentication and password hashing.
3. **Frontend Development**
    - Built UI components using React and Tailwind CSS.
    - Managed application state with Redux Toolkit and RTK Query.
    - Connected to backend APIs for authentication and chat operations.
    - Integrated Socket.IO client for real-time message updates and typing indicators.
    - Added emoji picker and message status features.
4. **Testing and Debugging**
    - Used Postman for API testing.

    - Debugged socket events and ensured proper separation of frontend and backend logic.

5. **Finalization**
    - Ensured environment variables were correctly set for both frontend and backend.
    - Verified real-time features and UI along with authentiaction.
    - Prepared documentation and finalized the project for submission.

Conclusion

The chat application successfully demonstrates the use of modern web technologies to create a responsive, real-time messaging platform. By separating frontend and backend logic and leveraging tools like Socket.IO and Redux, the project achieves scalability and maintainability. The experience gained in handling real-time data, authentication, and state management is valuable for future full-stack development projects.