Real-Time Chat Application using MERN Stack and Socket.IO

Introduction

Communication is at the core of every digital application today, and chat systems are an essential component of social platforms, customer support systems, and collaboration tools. This project presents a **full stack real-time chat application** using the **MERN stack (MongoDB, Express, React, Node.js)** combined with **Socket.IO** for real-time, bidirectional communication between users.

This application allows users to **sign up, log in, see online users, and chat in real time —** all with a clean, responsive UI and persistent message storage.

Abstract

The objective of this project is to build a modern, full-stack chat application that:

- Enables **real-time communication** without refreshing the page.
- Uses **JWT authentication** to secure user sessions.
- Utilizes **MongoDB** to store chat messages and user data.
- Uses **Socket.IO** to manage WebSocket connections between users.
- Separates the **frontend** (React) and **backend** (Express) for maintainability.
- Deploys the frontend using **Vercel** and the backend (recommended) using **Render**.

The chat app provides a user-friendly interface with real-time interactions, reflecting changes such as typing indicators and online user status instantly.

Tools Used

Tool / Library	Purpose
React.js	Frontend UI Framework
Tailwind CSS	UI Styling & Responsiveness
Node.js	JavaScript Runtime for Backend
Express.js	Backend Server Framework
MongoDB + Mongoose	Database & ODM
Socket.IO	Real-time bi-directional communication
JWT	Secure authentication token handling
Axios	API communication between frontend/backend
Vercel	Frontend deployment
Render	Backend deployment (recommended)

Tool / Library	Purpose
Git & GitHub	Version control and hosting

Steps Involved in Building the Project

1. Frontend Development (React + Tailwind)

- Built reusable UI components: login, signup, chat window, user list.
- Connected the frontend to the backend using Axios.
- Managed authentication tokens with `localStorage` and Axios headers.
- Displayed online users and handled UI for messaging in real time.

2. Backend Development (Node + Express + MongoDB)

- Created user and message schemas using Mongoose.
- Built RESTful APIs for user login, signup, authentication, and messaging.
- Implemented middleware to protect routes using JWT.
- Connected to MongoDB Atlas for cloud-based storage.

3. Real-Time Features with Socket.IO

- Established WebSocket connections between clients and server.
- Created `userSocketMap` to map user IDs to socket connections.
- Broadcasted events like new message, typing indicators, and online users.
- Handled disconnect events and updated online user lists.

Conclusion

This project successfully demonstrates the design and implementation of a **full stack real-time chat application** using modern web technologies. The use of **Socket.IO** enabled real-time communication, while the **MERN stack** ensured a scalable and modular architecture. The final application is responsive, interactive, and can be extended with features like group chat, file sharing, or notifications.

Through this project, we have gained hands-on experience in:

- Full stack development using JavaScript.
- Real-time communication protocols.
- Cloud-based deployment strategies.
- Managing state, authentication, and persistent user sessions.