REAL TIME CHAT APP WITH SOCKET.IO

Introduction

In this project, I followed a YouTube tutorial to make a real-time chat application using the MERN stack and Socket.io. MERN stands for MongoDB, Express, React, and Node.js. The video helped me understand how a chat app works, how messages are sent and received instantly, and how frontend and backend communicate with each other in real-time.

Abstract

The main goal of this project was to build a chat app where users can send and receive messages instantly without reloading the page. The app was created step-by-step starting from setting up the backend to designing the frontend. It uses Socket.io for real-time messaging and MongoDB to store chat data. The tutorial explained everything clearly and helped me understand how to manage both the server-side and client-side of a real-time app.

Tools Used

- MongoDB – For storing user details and chat messages.
- Express.js & Node.js – For creating the backend server and APIs.
- React.js – For building the user interface of the chat app.
- Socket.io – For real-time communication between users.
- Mongoose – To connect Node.js with MongoDB.
- Axios (or Fetch) – For making HTTP requests between frontend and backend.

Steps Involved in Building the Project

1.Planning and Designing

Before writing any code, I learned to plan the app properly. The video explained how to design the structure of the app, like what data will be stored (usernames, messages, etc.) and how the UI will look.

2.Backend Setup

First, I created a Node.js project using Express. Then I connected it with MongoDB using Mongoose. I made API routes to handle tasks like saving

messages or getting old messages. I also set up Socket.io to handle real-time messaging.

3.Frontend Setup

Then I created the frontend using React. I built components like the login screen, chat window, input box for sending messages, and a list to display messages. I also connected the frontend to the backend using sockets and HTTP requests.

4.Real-Time Chat Functionality

When a user sends a message, it's sent through Socket.io to the server, which then sends it to all connected users. The message is also saved to MongoDB. This allows users to chat in real-time without reloading the page.

5.Testing and Final Touches

I tested the app by opening it in different tabs to see if messages are being delivered instantly. I also added some improvements like clearing the input after sending a message and handling errors properly.

Conclusion

Overall, this project was a great learning experience. I got to work on both frontend and backend, and I understood how real-time communication works using Socket.io. The MERN stack made it easier to manage everything smoothly, and the project gave me a strong base to build more advanced real-time apps in the future. I'm happy with how it turned out, and I now feel more confident working with full-stack development.