Chatting Bro

A REAL-TIME CHAT APPLICATION

Introduction

This is a real- time chat application to facilitate communication between users.It uses MERN Stack to make utilities available.The front-end development utilizes tailwind CSS for styling different components whereas the backend development is catered by Node.Js Express.Js and MongoDB database.The web application uses socket.io for real time message updates.

System Architecture:

Front-end

- *React.Js* for Framework
- *Tailwind CSS* for Styling
- *Socket.io* for Real-Time Messages

Back-end

- *Express.Js* for Server Framework
- *MongoDB* for Database
- *JSON Web Token*s for Authentication

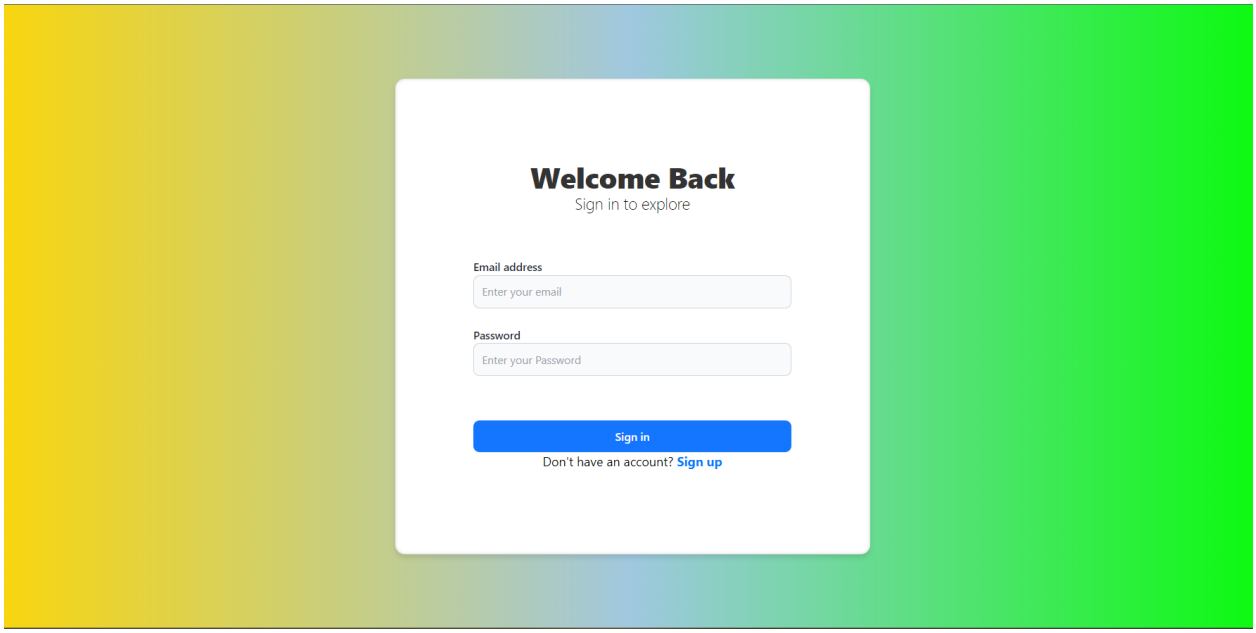
Dependencies

- *Cors*- to enable secure cross-origin interactions for servers and front-ends on separate domains.
- *BcryptJs*- to securely hash and salt passwords, enhancing data security by storing irreversible password hashes in the database.
- *Web vitals*- to measure real user experiences, focusing on metrics like page speed and interactivity for optimal website performance monitoring.
- *React-scripts*- to simplify React app development and builds with pre-configured scripts for tasks like starting a server and creating production builds.
- *React-router-dom*- to simplify React app navigation by managing components based on URL changes, enabling dynamic single-page applications.

Key Features:

1.User Registration and Authentication:

Users can Sign Up and Sign in securely using JWT tokens for authentication and authorization



2. Real-Time Messaging:

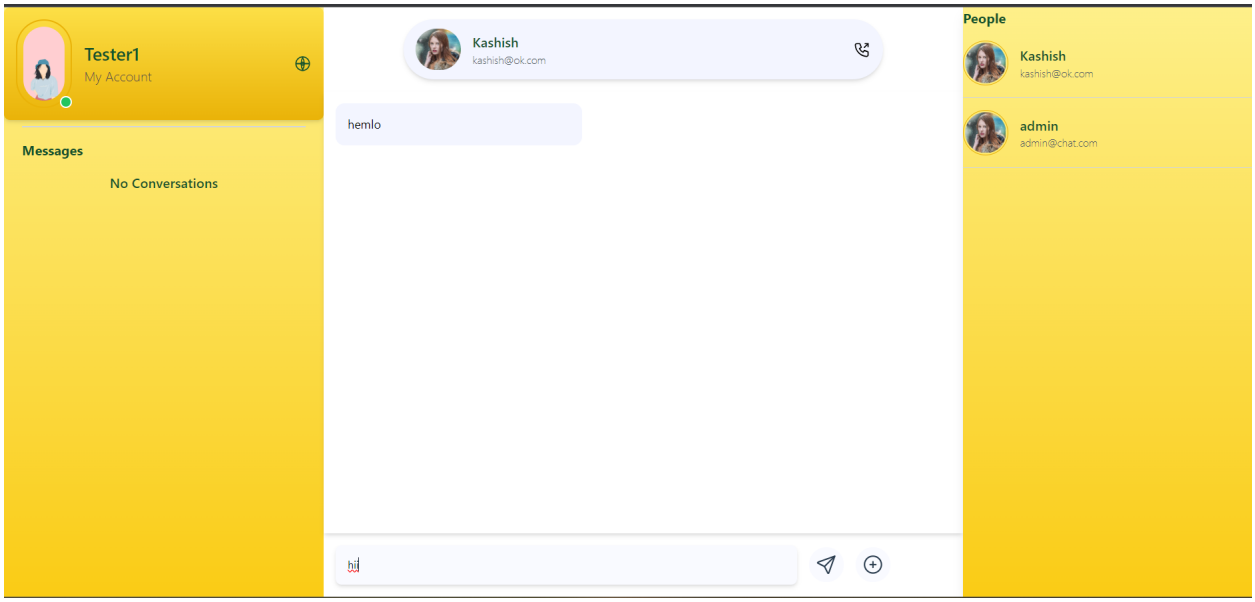
Users can send and receive real-time messages .This feature is incorporated using Socket.io. The messages are stored in a database provided by MongoDB.

3.Awaited Message Notifications:

Users can find new,awaited messages on the left sidebar of the application.

4.Conversations:

Users can create new conversations and add other users.The data of the user and receiver are stored in the MongoDB database.



Data Flow:

User Registration and Authentication:

- New users can Sign up whereas existing users can Sign in
- The back-end verifies user credentials and generates JWT tokens for authentication.

Real-time Messaging:

- Messages are sent from one client and broadcasted to another using Websockets like Socket.io
- They are also stored in a MongoDB database

Conversations:

- Users can create new conversations and add other users.
- The data of the user and receiver are stored in the MongoDB database.

Awaited Messages:

- New messages displayed under profile section

Database Schema:

- **Users Collection :** Stores user data like username and email
- **Conversations Collection:** Stores conversation data like Sender_Id,Receiver_Id and Conversation_Id
- **Messages Collection:** Stores message data

Third-Party Services:

- **Socket.io:** Used for real-time communication.
- **MongoDB Atlas :** Cloud-hosted MongoDB database for storing user data, conversations, and messages.

Security:

- User authentication and authorization are handled securely using JWT tokens.
- Data transmission between clients and the server is encrypted using HTTPS.
- Proper input validation and sanitation are implemented to prevent security vulnerabilities.

How to run the Project:

- Download the zipped folder from GitHub and extract all the files onto your device.
- Open the unzipped folder using Visual Studio Code.
- In the VS Code terminal, execute the command **npm install** to install project dependencies.
- Navigate to the client directory by running the command **cd client.**
- Install all the necessary dependencies for the client directory by executing **npm install.**
- Return to the root folder by using the command **cd ..**
- Access the server directory with the command **cd server**, then start the backend server by running **npm run dev**.
- In a new terminal window, navigate to the public directory using **cd public**.
- Start the frontend application by executing **npm run start**
- Ensure that MongoDB is running in the background for database operations.
- Additionally, in the **connection.js** file located in the **db** folder, update the MongoDB URL accordingly.