Real-Time Chat Application using Socket.IO, JWT, and MongoDB

Introduction

The Real-Time Chat Application is a full-stack, browser-based platform for instant messaging. It supports **private chats**, **group chats**, and **room-based chats**, allowing users to communicate seamlessly. Key features include **profile management**, **typing indicators**, **online/offline detection**, **secure authentication**, and **message history** for every conversation. Built using modern technologies, the platform ensures real-time communication with a clean user interface and secure data handling.

Abstract

This chat system is designed to offer a scalable, secure, and responsive environment for one-on-one or multi-user communication. It uses **JWT** for secure authentication and **Socket.IO** for real-time data transmission. All chat messages—private, group, or room-based—are stored in a **MongoDB** database to maintain **conversation history** even after logout. Advanced input validation, message timestamps, and status indicators contribute to a polished user experience.

Tools Used

- **Frontend:**
  HTML, CSS, JavaScript, Tailwind CSS
- **Backend:**
  Node.js, Express.js
- **Database:**
  MongoDB
- **Authentication:**
  JWT (JSON Web Token), bcrypt.js
- **Real-Time Messaging:**
  Socket.IO
- **Others:**
  Postman (API Testing), VS Code

Steps Involved in Building the Project

1. User Authentication & Profile:

The registration form enforces strong validations: the name must start with a letter and contain no digits, the email must be in a valid format and cannot begin with a number, the password must be at least 8 characters long and include at least one uppercase letter and one special character, and the confirm password field must match the password exactly. The login form securely validates user credentials. Once logged in, a JWT token is stored on the frontend to protect private routes. A profile page displays the current user's name and email, along with a secure logout button.

2. Contact System:

Users can add contacts via email and view a real-time contact list showing online or offline status. Contacts can be refreshed manually also and private chats can be started instantly by selecting contact.

3. Private Chat:

Chats are handled in secure privately with real-time messaging, typing indicators, timestamps, and status indicators (online/last seen). All conversations are stored in MongoDB for history. The UI places user messages on the right and others on the left.

4. Group Chat:

Users can create or join groups via a group Join ID. Group chat supports sender names, typing status, timestamps, and persistent chat history. UI provides options to copy group ID, leave group, and refresh the list.

5. Room-Based Chat:

Lightweight, real-time discussion rooms using Room IDs with similar logic to group chat. Ideal for open discussions, with full message history.

6. Message System:

Key socket events (message, typing, join room, etc.) enable real-time interaction. All messages are stored for history, and the UI dynamically updates with auto-scroll.

7. Frontend UI & Form Validation:

Built using Tailwind CSS for clean design. Secure password inputs, error messages shown below fields and field-level error highlighting and alert messages ensure user-friendly interaction like Success/Failure alerts.

Conclusion

The Real-Time Chat Application is a secure and functional platform supporting all major messaging needs—**private, group, and room-based chats**. Features like **real-time updates**, **chat history**, **status tracking**, and **strong validations** ensure a high-quality user experience. This project is ideal for social and educational platforms and can be expanded with features like **media sharing**, **file uploads**, **notifications**, and **mobile support** in future versions.