Real-Time Chat Application

Overview

This is a simple real-time chat application that enables users to join a chat room, send messages, and see real-time updates from other participants. The application is built using HTML, JavaScript, and the Socket.IO library for real-time communication.

Setup

1. **Install Dependencies:** Ensure that you have Node.js installed on your machine. If not, download and install it from Node.js website.
2.
3. **Clone the Repository:**
   bashCopy code
   ```bash
   git clone <repository-url> cd <repository-directory>
   ```
4. **Install Node Modules:**
   bashCopy code
   ```bash
   npm install
   ```

Usage

1. **Run the Server:**
   bashCopy code
   ```bash
   node server.js
   ```
   The server will run on `http://localhost:3000`.
2. **Open the Application:** Open `index.html` in a web browser by navigating to `http://localhost:3000`.
3. **Join the Chat:**
   - Enter your name when prompted.
   - Start sending and receiving messages in real-time.

Files and Structure

1. `index.html`

- **HTML structure:** Defines the basic structure of the chat application.
- **Script tags:** Include Socket.IO library and link to the custom JavaScript file (`script.js`).
- **CSS styles:** Basic styling for the chat layout.

2. `script.js`

- **Socket.IO connection:** Establishes a connection to the Socket.IO server.
- **Event listeners:** Listens for server events ('chat-message', 'user-connected', 'user-disconnected') and updates the UI accordingly.
- **Form submission:** Listens for form submission, sends messages to the server, and updates the UI.

3. `server.js`

- **Socket.IO server:** Creates a Socket.IO server and listens on port 3000.
- **User management:** Keeps track of connected users using the `users` object.
- **Event handling:** Listens for 'new-user', 'send-chat-message', and 'disconnect' events, broadcasting corresponding events to all clients.

Dependencies

- **Socket.IO:** A JavaScript library for real-time web applications. It enables real-time, bidirectional, and event-based communication.

Possible Improvements

- Implement user authentication for a more secure chat.
- Enhance the user interface and styling.
- Add error handling in both the client and server code.
- Deploy the application to a production environment with HTTPS.