**🛠️ Multi-User Chat System – Backend Documentation**

**Overview**

The backend is developed using **Node.js** with **Express.js**, connected to **MongoDB** via **Mongoose**. It handles user authentication, chat room creation, message storage, file uploads, and real-time communication via **Socket.IO**. All interactions are RESTful or event-based for efficient, scalable communication.

**Core Modules & Their Purpose**

| **Module** | **Description** |
| --- | --- |
| **User Controller** | Manages user registration, login, and JWT-based authentication. Validates credentials and issues tokens. |
| **Chat Controller** | Handles creation and retrieval of chat threads between users. Ensures only valid participants can initiate/access a chat. |
| **Message Controller** | Stores, fetches, and manages messages sent in a chat. Supports text, emoji, and media message formats. |
| **Media Upload Handler** | Uses multer to accept and validate file uploads (images, videos, docs). Saves paths and returns file metadata. |
| **Socket.IO Handler** | Enables real-time messaging. Listens to message events and broadcasts them to the appropriate recipients. |
| **Authentication Middleware** | Protects private routes by validating JWT tokens sent by clients. Ensures only logged-in users access APIs. |

**Functional Flow**

1. **User Authentication**  
   Users sign up or log in using credentials. Backend issues a JWT for authenticated requests.
2. **Chat Initialization**  
   When two users start chatting, the backend checks for an existing chat thread or creates a new one. Only authorized users can access the chat.
3. **Message Handling**  
   Messages sent by users are stored in MongoDB. The backend supports messages with text, emojis, and file references.
4. **File Uploads**  
   When a user sends a file, it's processed by multer, saved to a storage path, and linked to the message document.

**Technologies Used**

* **Node.js + Express.js** – Backend server and routing
* **MongoDB + Mongoose** – Document database with schema models for users, chats, and messages
* **Socket.IO** – Enables WebSocket communication for real-time messaging
* **Multer** – Middleware for handling multipart/form-data for file uploads
* **JSON Web Token (JWT)** – Used to authenticate and authorize users
* **Cloudinary**  – For hosting uploaded media (optional production config)

**Example Usage**

* A user logs in and receives a JWT token.
* On the frontend, the user starts a chat with another user.
* The frontend sends a message to the backend via Socket.IO or REST API.
* The backend saves the message to MongoDB and emits it to the recipient in real time.
* If a file is included, it's processed by Multer, uploaded to storage, and its link is stored with the message.
* The receiving user gets a live notification and the new message instantly appears in their UI.