Endpoints' Explanation

Overview

This document provides a comprehensive guide for front-end developers to integrate the backend APIs of the Chat Service App and develop the UI according to the system's core functionalities. The main scenario of the project is explained step-by-step, with corresponding APIs mentioned for each phase. Front-end developers can refer to the project's flowchart image located in the Chat-Service\server\Chat Service Flowchart.jpg for a visual understanding, and they can use the FastAPI Swagger documentation to explore how each API works in detail.

Main Scenario

1. User Signup and Login

- **Signup**: The user signs up using the **POST** /signup API.
- **Login**: After successful signup, the user logs in using the **POST** /token API to get a JWT token. This token must be stored securely and used in the headers of all subsequent API requests.

2. Main Dashboard

- After logging in, the user is directed to the main dashboard where the following functionalities are available:
  - **Create a Chat Room**: The user can create a chat room using the **POST** /chat-rooms API. After creating a room, the room's unique ID will be generated and can be shared with other users.
  - **Search for a Chat Room**: The user can search for a chat room by entering its ID using the **GET** /search-chat-rooms API. If the room is found, the user will get its details.
  - **Get My Chat Rooms**: The user can fetch the list of chat rooms they have created and own using the **GET** /chat-rooms API. This is useful to manage owned rooms.
  - **Get Submitted Chat Rooms**: To retrieve the list of both group and private chats the user has joined, the **POST** /user/submitted-chat-rooms API is used. The response can include both group chats and private chats depending on the API's input.

3. Join Requests and Chat Room Management

- **Submit a Join Request**: After the user searches for a chat room, they can submit a join request to the chat room using the **POST** /join-request API. This request is sent to the owner of the chat room for approval.

- **View Chat Room Details**: The owner of the chat room can view details of the chat room and see a list of pending join requests using the **GET** /chat-room-details API. This includes the room's details and any join requests submitted by other users.
- **Delete a Chat Room**: If the owner decides to delete the chat room, they can do so by calling the **DELETE** /delete-chat-room API.
- **Handle Join Requests**: The owner of a chat room can handle the submitted join requests by viewing the request details using the **GET** /join-request-details API. They can then approve or reject the request using the **POST** /handle-join-request API. If approved, the requester is added to the chat room session, enabling them to join the chat in real-time.

4. Private Chat and Group Chat Management

- **Get Online Private Chat Users**: After logging in, the user can view a list of online users with whom they have private chats using the **GET** /pv-online-users API. This helps the user quickly see which private chat contacts are online.
- **Join a Chat Room**: To join a chat room (either private or group), real-time communications are handled using **Socket.io** events. The front-end developer should implement Socket.io to join rooms and handle live messaging. This ensures that the UI can show a list of current online users, real-time messages, and multimedia. In Chat-Service\client directory, a React.js example is given on how to use and test these Socket.io events. This example also includes a portion of the main scenario, which can be helpful.

5. Socket.io Real-Time Communication

- The following **Socket.io** events are crucial for the chat room features, enabling real-time messaging, multimedia support, and user interactions:
    - **Connect Event**:
        - **Purpose**: To establish a connection when a user joins a chat room.
        - **Usage**: On connection, the server will authenticate the user via a token, mark them as online, and broadcast the *'join'* event to other users in the room. It also sends the user the list of online users and recent messages in the chat room.
    - **Chat Event**:
        - **Purpose**: To handle real-time messaging.
        - **Usage**: This event is triggered whenever a user sends a message (text, image, or file). The message is processed and emitted to all users in the room.
    - **Get More Messages Event**:
        - **Purpose**: To retrieve older messages in a chat room when a user scrolls up.
        - **Usage**: When a user requests more messages (e.g., by scrolling up in the chat), the front-end triggers this event to load older messages.
    - **Disconnect Event**:
        - **Purpose**: To handle a user leaving the chat room.
        - **Usage**: On disconnect, the server updates the user's online status, notifies others in the chat, and removes the user from the room.

6. Group and Private Chat Room Management

- **Get Chat Room Details**: To display a chat room's details, the **GET** /chat-room/{chat_room_id} API provides comprehensive information about the room. This is useful for the UI to show the room's name, owner, and other attributes.
- **Create a Private Chat Room**: Users can create a private chat room with another user by clicking on their username in a group chat and calling the **POST** /pv-chat-room API. The UI should allow the user to start a one-on-one private chat, redirecting them to the private chat interface and adding the private chat to their list of submitted rooms.

Summary of Important APIs

- **Signup**: **POST** /signup
- **Login**: **POST** /token
- **Create Chat Room**: **POST** /chat-rooms
- **Search Chat Room**: **GET** /search-chat-rooms
- **Get My Chat Rooms**: **GET** /chat-rooms
- **Get Submitted Chat Rooms**: **POST** /user/submitted-chat-rooms
- **Submit Join Request**: **POST** /join-request
- **Get Chat Room Details**: **GET** /chat-room-details
- **Delete Chat Room**: **DELETE** /delete-chat-room
- **Get Join Request Details**: **GET** /join-request-details
- **Handle Join Request**: **POST** /handle-join-request
- **Get Private Chat Online Users**: **GET** /pv-online-users
- **Create Private Chat Room**: **POST** /pv-chat-room