Telehealth Chat API Documentation

Overview

The **Telehealth Chat API** provides endpoints for managing chat tokens, saving messages, and retrieving chat history. This documentation explains how to use the API and integrate with your system.

Base URL

The API is hosted at:

```
http://localhost:8080/api
```

Authentication

All API endpoints are secured using a token-based system. To generate a chat token, use the /token/generate-token endpoint.

Endpoints

1. Generate Chat Token

POST /token/generate-token

Description

Generates a chat token for either a patient or a consult, which is used to authenticate users in the chat system.

Request Body

Field	Type	Required	Description
patient_id	string	No	The ID of the patient.
consult_id	string	No	The ID of the consult.
room	string	Yes	The ID of the chat room.

Example Request

```
{
  "patient_id": "123",
  "room": "room1"
}
```

Example Response

Status: 200

```
{
  "chat_token": "eyJhbGciOiJIUzI1NiIsInR..."
}
```

Status: 400

```
{
  "error": "Missing required fields"
}
```

---

2. Get Chat Messages

GET /chat/messages/{room}

Description

Retrieves all messages from the specified chat room.

Path Parameters

Field	Type	Required	Description
room	string	Yes	The ID of the chat room.

Example Request

```
GET /chat/messages/room1
```

Example Response

Status: 200

```
{
  "success": true,
  "data": [
    { "id": "1", "room": "room1", "sender": "user1", "message": "Hello" },
    { "id": "2", "room": "room1", "sender": "user2", "message": "Hi there!" }
  ]
}
```

Error Handling

Common Errors

Status Code	Description
400	Validation error.
401	Unauthorized request.
404	Resource not found.

Example Error Response

Status: 400

```
{
  "error": "Missing required fields"
}
```

Integration with Socket.IO

The chat system uses **Socket.IO** for real-time communication. Below are the supported events.

1. Connect

Establishes a connection to the server using a valid chat token.

Client Code

```
const socket = io("http://localhost:8080", {
  auth: {
    token: "your_chat_token"
  }
});
```

---

2. Join Room

Joins a specified chat room.

Event Name

join

Client Code

```
socket.emit("join", { room: "room1" });
```

---

3. Leave Room

Leaves the current chat room.

Event Name

leave

Client Code

```
socket.emit("leave", { room: "room1" });
```

---

4. Send Message

Sends a message to the chat room.

Event Name

message

Client Code

```
socket.emit("message", { room: "room1", message: "Hello, world!" });
```

---

5. Receive Messages

Listens for messages broadcasted in the room.

Event Name

message

Client Code

```
socket.on("message", (data) => {
  console.log("Message received:", data);
});
```

Notes

- Ensure the chat token is valid before making requests.
- All requests should be sent to the /api prefix.
- Use Swagger UI for detailed API documentation and testing:

  ```
  http://localhost:8080/api-docs
  ```

Contact

For further assistance, contact the development team at **witsanu091_@hotmail.com**.