Chat Feature API

Event: connectRooms

- **Description**: When a user connects to the chat service, the `connectRooms` event is triggered. This event signifies the user's entry into the chat environment and facilitates the retrieval of rooms the user has joined. The payload contains essential user identification details, such as their user ID and first name. Upon emitting the event, the server responds with data consisting of a list of all rooms that the user has joined.
- **Payload**:
  - `user_id`: ID of the connecting user.
  - `first_name`: First name of the connecting user.
- **Emits**: `socket.emit('connectRooms', allRooms)`

Explanation: This data represents a list of all rooms that the user has joined. It is emitted by the client upon connecting to the chat service and triggers the server to respond with information about the user's joined rooms.

- ***Example***

```
{
"user_id": "65e01d25c002d6508c631ad9",
"first_name": "Mazrui"
}
```

Event: sendMessage

- **Description**: This event is used to send a message to a specific chat room. It includes details such as the message content, room ID, recipients, sender details, etc.
- **Usage:** When a user sends a message in the chat interface, package the message details into the payload and emit this event to send it to the server for distribution.

- **Payload**: Contains the message details, including sender information, message content, and other relevant data.
    - `message`: The text message content.
    - `room_id`: ID of the target room.
    - `recipients`: Array containing recipient information, such as recipient ID and whether the message has been seen.
- **Emits**: `io.sockets.in(room).emit('message', payload)`

Explanation: This event is emitted by the server to notify all clients in the specified room about the new message.

- ***Example***


```
{

"message": "Can you see this message?",
"room_id": "663c8020fefd1941540edbca",
"recipients": [{
"_id": "65e01d25c002d6508c631ad9",
"msg_seen": false }],
"isStoryReply": false,
"reply_message": {},
"message_type": "",
"attachments": [],
"sender_id": "65e01d25c002d6508c631ad9",
"first_name": "Abdul",
"last_name": "Poyilthodi Chathamparambil",
"dateCreated": "2024-05-15T07:19:36.621+00:00"
}
```


Event: typing

- **Description**: This event indicates that a user is currently typing a message in a specific chat room.
- **Usage:** Implement this event to provide real-time typing indicators to other users in the chat room.
- **Payload**: Carries the room ID and a message indicating that the user is typing
    - `room_id`: ID of the room where typing is happening.
    - `message`: A notification indicating that the user is typing.
- **Emits**: `io.sockets.in(room).emit('typing', payload)`

Explanation: Emits the typing indicator to all clients in the specified room.

- ***Example***

```
{ "room_id": "663c8020fefd1941540edbca", "message": "I am typing..." }
```

Event: sendNewMessage

- **Description**: This event is used to send a new message to the chat, including details such as room name, message type, recipients, sender details, etc.
- **Usage:** Similar to 'sendMessage', but primarily used for initiating a new conversation or sending messages in a different context.
- **Payload**: Contains the message details, including sender information, message content, and other relevant data.
    - `room_name`: Name of the chat room or group.
    - `type`: Type of message (e.g., peer-to-peer).
    - `recipient_id`: ID(s) of the recipient(s).
    - `user_name`: Name of the sender.
    - `image_url`: URL of the sender's profile image.
    - `message`: Details of the message, including content, sender, and message type.
- **Emits**: `io.sockets.emit(user_id, data.message)`

Explanation: Sends the new message data to the specified user.

- ***Example***

```
{ "room_name": "Amazing group", "type": "peer-to-peer",
"recipient_id": ["5fa24825a8e9cddf8f35df71"], "user_name": "Bharat
Subbaraman", "image_url": "https://nathanhrerp.s3.eu-central-
1.amazonaws.com/users/Bharat-Subbaraman/Bharat-Subbaraman.webp",
"message": { "_id": "63cfbfc722c59b536467f03c", "message": "",
"recipients": [{ "_id": "", "msg_seen": "" }], "message_type": "text",
"attachments": [], "sender_id": "616e6d5195ae5b3f1adc87dc",
"first_name": "Abdul", "last_name": "Poyilthodi Chathamparambil" } }
```

Event: createGroup

- **Description**: This event is triggered when a new group chat is created. It includes details such as the group name, members, creator ID, etc.
- **Usage:** Implement this event when creating a new group chat in the application interface.
- **Payload**:
    - `_id`: ID of the newly created group.
    - `room_name`: Name of the group.
    - `recipients`: Array of member IDs.
    - `type`: Type of chat (e.g., group-chat).

    - `creator_id`: ID of the user who created the group.
    - `image_url`: URL of the group's profile image.
- **Emits**: `io.sockets.in(room).emit(group_{user_id}, payload)`

Explanation: Emits the group creation data to all clients involved in the group chat.

- ***Example***

{ "_id": "63cfbe252ff0dd6ab8615e95", "room_name": "Future group", "recipients": ["63cfbe252ff0dd6ab8615e95", "63cfbe252ff0dd6ab8615e95"], "type": "group-chat", "creator_id": "65e01d25c002d6508c631ad9", "image_url": "https://nathanhrerp.s3.amazonaws.com/users/Abdul%2%E2%80%A6rambil/Abdul%20-Poyilthodi%20Chathamparambil.webp" }

Event: deleteMessage

- **Description**: This event is used to delete a message from a specific chat room. It carries the message ID, user ID, and room ID as payload.
- **Usage:** Implement this event to allow users to delete their messages within the application.
- **Payload**: Contains the message ID, user ID, and room ID for identification.
    - `message_id`: ID of the message to be deleted.
    - `user_id`: ID of the user requesting the deletion.
    - `room_id`: ID of the room containing the message.
- **Emits**: `io.sockets.in(room).emit('delete', payload)`

Explanation: Notifies all clients in the specified room about the deleted message.

- ***Example***

{ "message_id": "63cfbfc722c59b536467f03c", "user_id": "65e01d25c002d6508c631ad9", "room_id": "663c8020fefd1941540edbca" }

Event: editMessage

- **Description**: This event is used to edit a message in a specific chat room. It includes details such as the message ID, updated message content, room ID, and user ID.
- **Usage:** Implement this event to enable users to edit their messages after they've been sent.
- **Payload**: Contains the updated message details, including message ID, content, sender information, etc
    - `message_id`: ID of the message to be edited.
    - `message`: Updated content of the message.
    - `room_id`: ID of the room containing the message.

    - `user_id`: ID of the user who edited the message.
- **Emits**: `io.sockets.in(room).emit('edited', payload)`

Explanation: Emits the edited message data to all clients in the specified room.

- ***Example***

```
{ "message_id": "63cfbfc722c59b536467f03c", "message": "I have edited this message again", "room_id": "663c8020fefd1941540edbca", "user_id": "65e01d25c002d6508c631ad9" }
```

Event: messageSeen

- **Description**: This event is triggered when a message is seen by users in a chat room. It includes the message ID and room ID.
- **Usage:** Implement this event to provide read receipts or seen indicators for messages in the chat interface.
- **Payload**: Contains the message ID and room ID for identification.
    - `message_id`: ID of the message that has been seen.
    - `room_id`: ID of the room containing the message.
- **Emits**: `io.sockets.in(room).emit('seen', payload)`

Explanation: Notifies all clients in the specified room that a particular message has been seen.

- ***Example***

```
{ "message_id": "63cfbfc722c59b536467f03c", "room_id": "663c8020fefd1941540edbca" }
```

Event: pinMessage

- **Description**: This event is used to pin or unpin a message within a specific chat room. It includes details such as the message ID, room ID, and pin status.
- **Usage:** Implement this event to allow users to pin important messages for easy reference.
- **Payload**: Contains the message ID, room ID, and pin status for identification.
    - `_id`: ID of the message to be pinned or unpinned.
    - `room_id`: ID of the room containing the message.
    - `pin_message`: Boolean indicating whether to pin or unpin the message.
- **Emits**: `io.sockets.in(room).emit('pin_message', payload)`

Explanation: Notifies all clients in the specified room about the pinned message status.

- ***Example***

```
{ "_id": "63cfbfc722c59b536467f03c", "room_id": "663c8020fefd1941540edbca", "pin_message": false }
```