Project Description

Title: `Diddyscord`

Description:

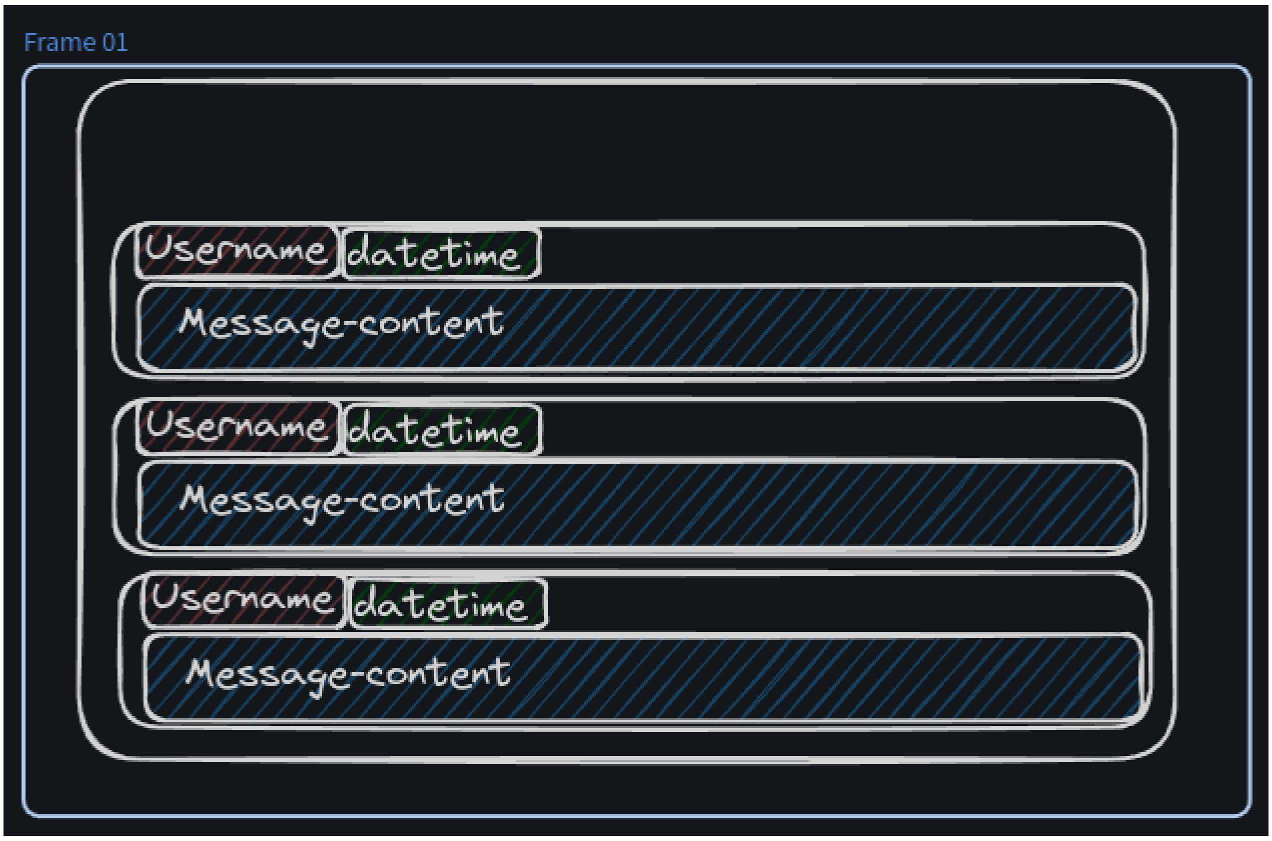
Diddyscord is a real-time chat application designed to facilitate seamless communication between users. With Diddyscord, users can register, send friend requests, and engage in private or group chats.The app guarantees that messages are sent instantly, enabling immediate communication. Users have the ability to see their chat history, handle friend requests, and get alerts about new messages and friend requests. Diddyscord strives to offer a platform that is both user-friendly and efficient for keeping in touch with friends and colleagues.

User Stories:

Chatroom

> As a *User* I am able to *get in a chat available to me* so that I can *view messages in a chatroom*

Wireframe



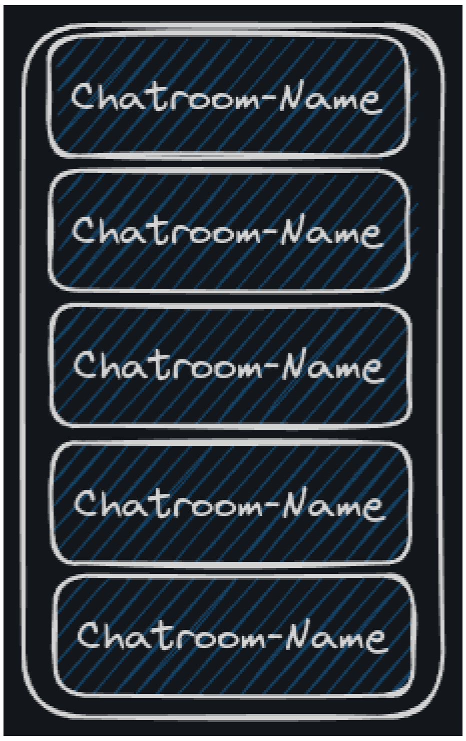
Acceptance Criteria

- Ability to view messages linked to a specific chatroom.
- Should display the user that sent the message, the date and time of the message being sent, and the message content.
- Orders them by most recent at the bottom.
- The ability to scroll up to view the message history.

Overview Chatrooms

> As a *User* I am able to *get a list of chatrooms I am assigned to* so that I can *see which chatrooms are available for me*

Wireframe



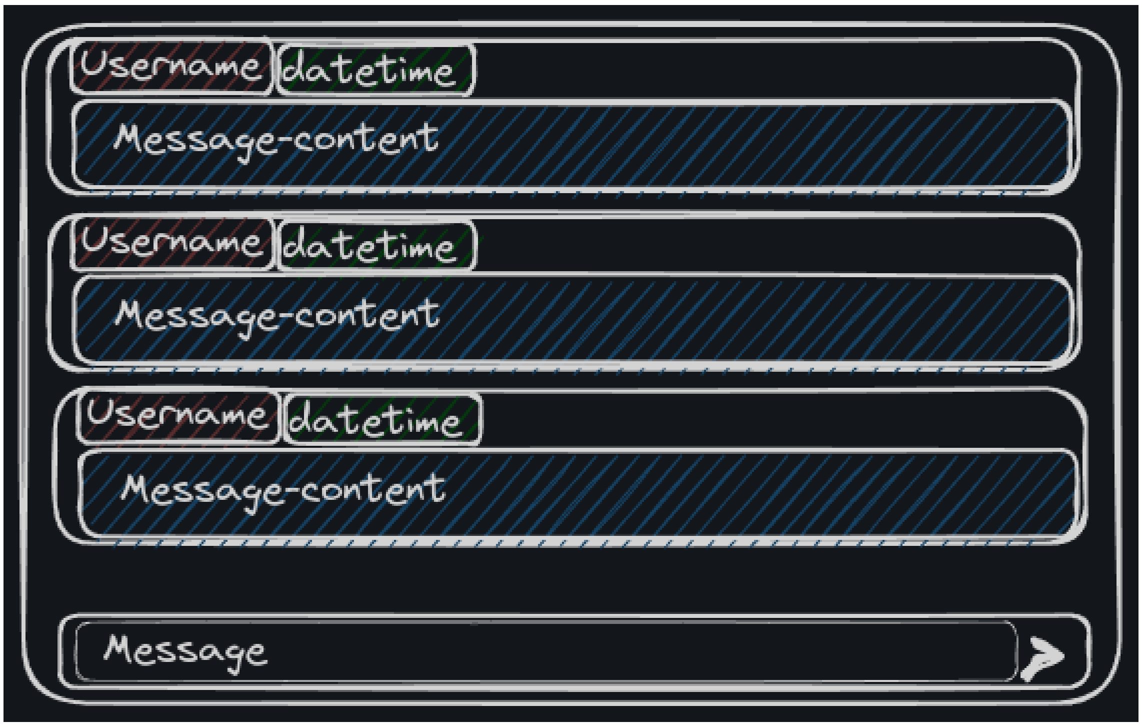
Acceptance Criteria

- A list of chats with their name displayed.
- When a chat is clicked, the chat needs to be highlighted.
- When a chat is clicked, the chat should open on the screen.

Send Message

> As a *User* I am able to *send a message to another user* so that I can *communicate with them*

Wireframe



Acceptance Criteria

- Ability to type and send a message.
- The message should appear in the chatroom in real-time.
- The message should be stored in the database with the correct sender and receiver information.

Receive Message

> As a *User* I am able to *receive messages from another user in real-time* so that I can *communicate with them*

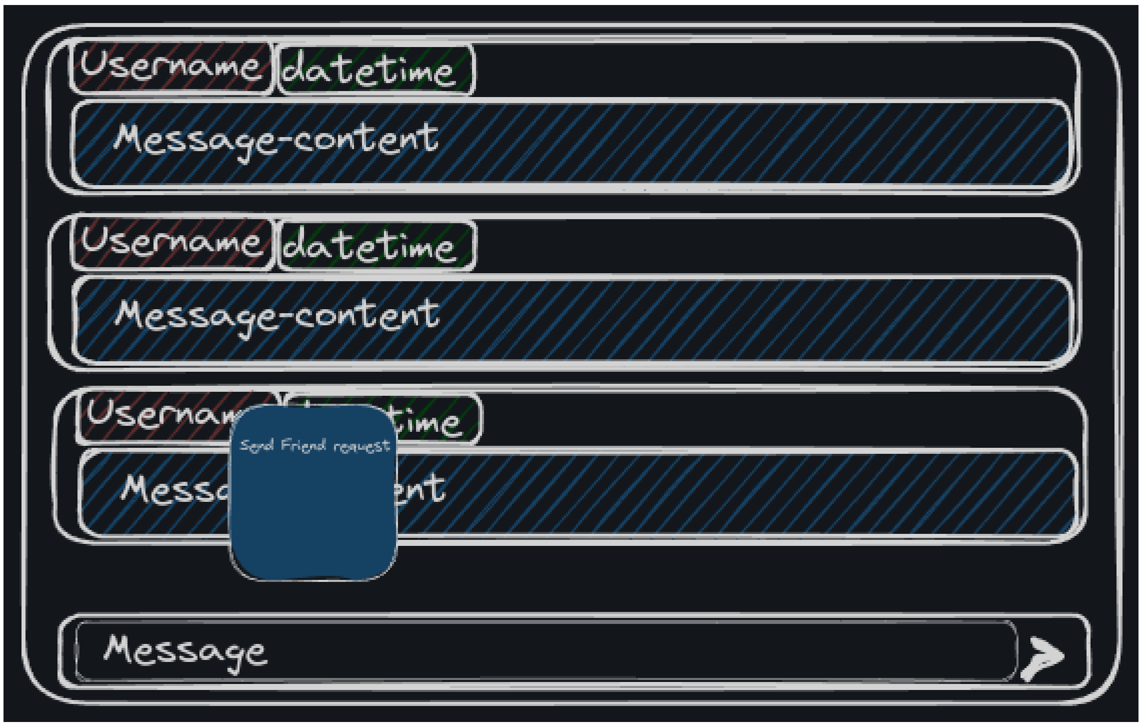
Acceptance Criteria

- Messages sent by another user should appear in the chatroom in real-time.
- The message should be stored in the database with the correct sender and receiver information.

Friend Request

> As a *User* I am able to *send a friend request to another user* so that I can *associate with a user*

Wireframe



- Clicking on a user shows a popup that allows you to send a friend request.

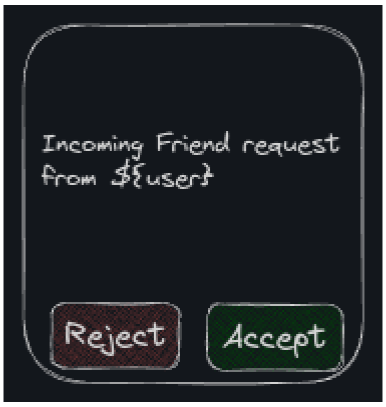
Acceptance Criteria

- Ability to send a friend request to another user.
- The friend request should be stored in the database with the correct sender and receiver information.

Overview Friend Requests

> As a *User* I am able to *view my friend requests* so that I can *accept or reject a request*

Wireframe



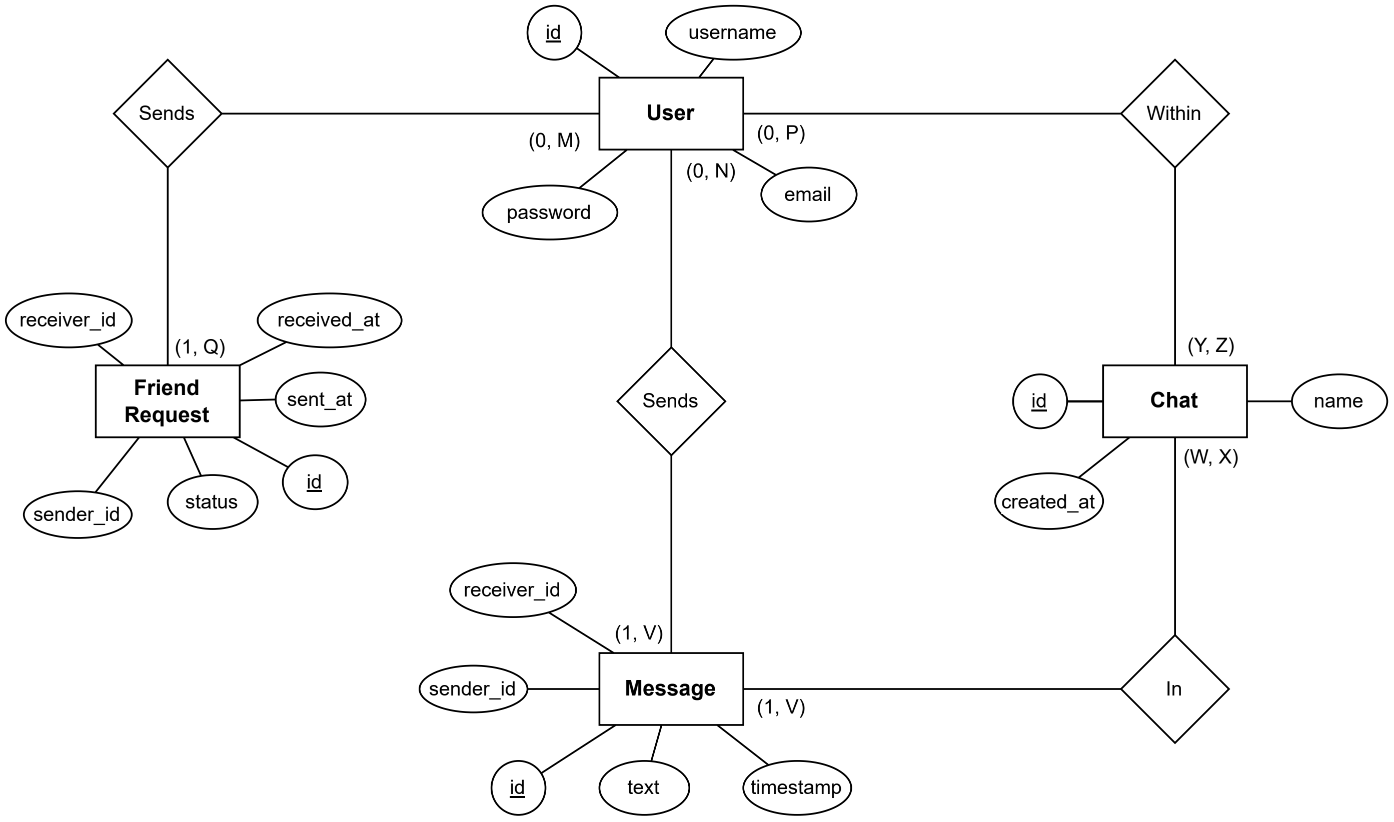
Acceptance Criteria

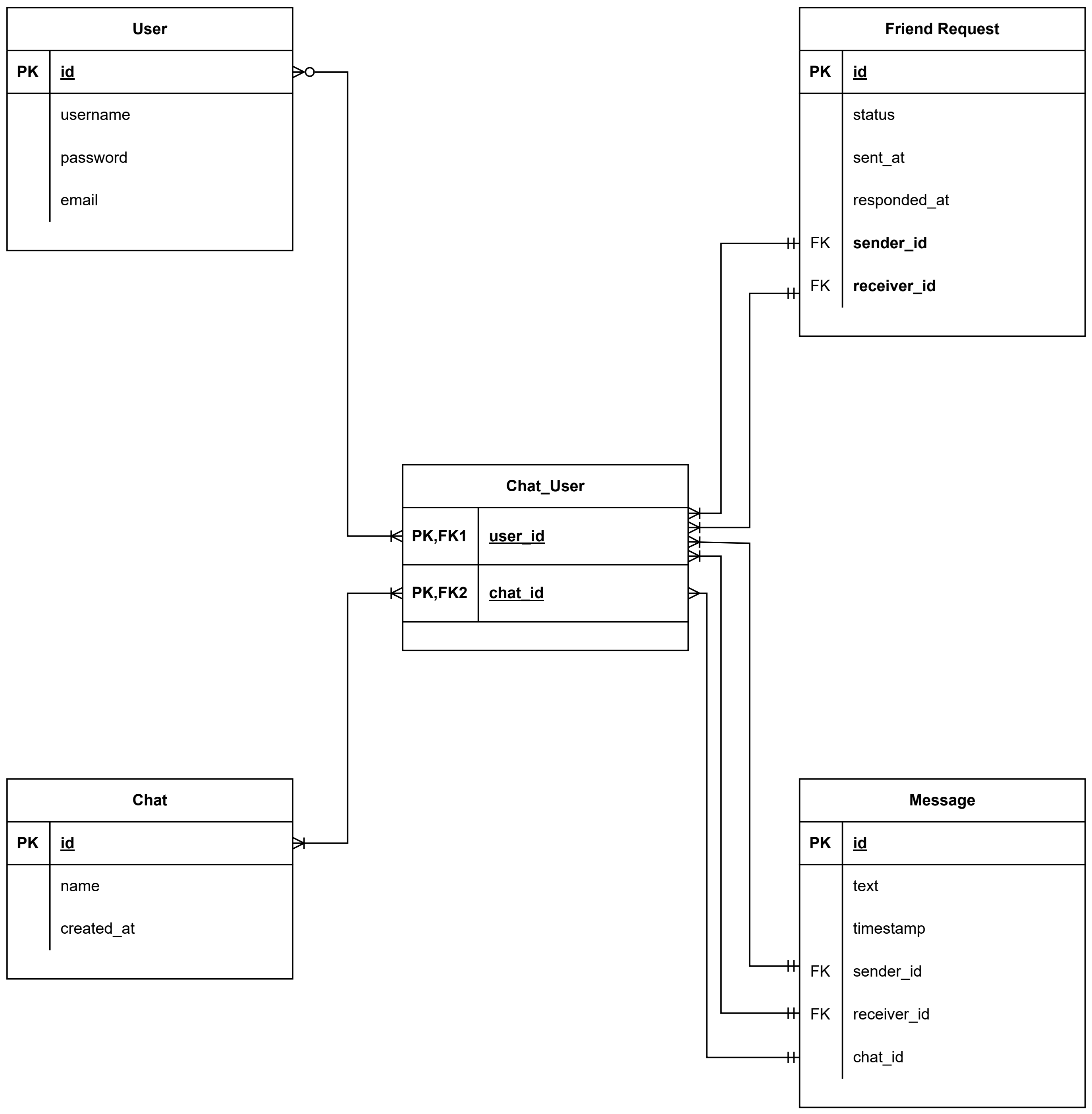
- Ability to view pending friend requests.
- Ability to accept or reject a friend request.
- The friend request status should be updated in the database.

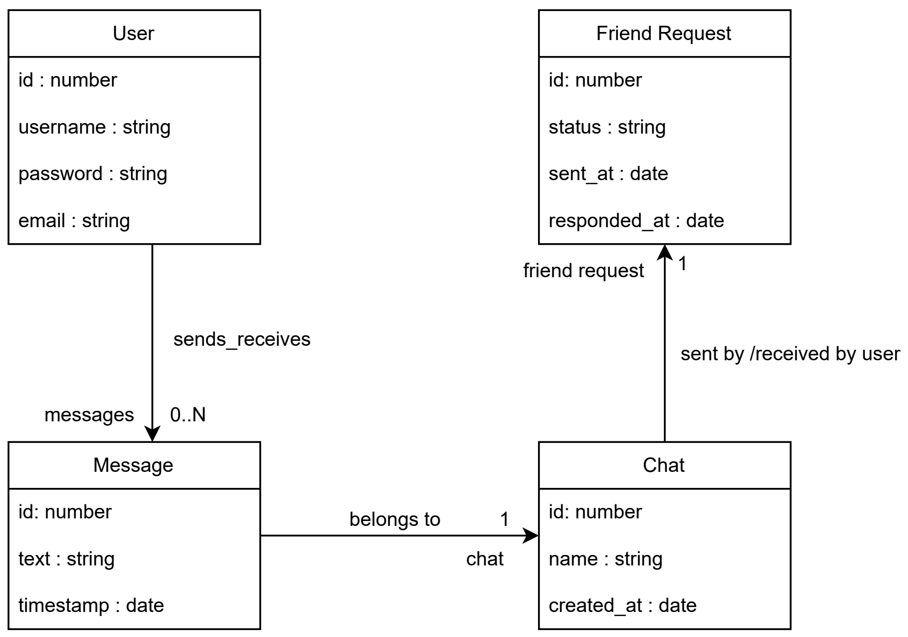
Chat API

Endpoints:

1. **User Authentication:**
    - `POST /api/auth/register`: Register a new user.
    - `POST /api/auth/login`: Log in a user.
    - `POST /api/auth/logout`: Log out a user.
2. **Friend Requests:**
    - `POST /api/friends/request`: Send a friend request.
    - `GET /api/friends/requests`: Get pending friend requests.
    - `POST /api/friends/requests/{requestId}/accept`: Accept a friend request.
    - `POST /api/friends/requests/{requestId}/reject`: Reject a friend request.
3. **Chats:**
    - `GET /api/chats`: Get a list of chatrooms for the user.
    - `GET /api/chats/{chatId}`: Get messages for a specific chatroom.
    - `POST /api/chats/{chatId}/messages`: Send a message to a chatroom.
4. **Real-time Messaging:**
    - Implement WebSocket in the backend for real-time messaging as the database supports WebSockets.


id
username
Sends
User
Within
(0, M)
(0, P)
(0, N)
password
email
receiver_id
received_at
(1, Q)
(Y, Z)
Friend Request
sent_at
Sends
id
Chat
name
sender_id
status
id
(W, X)
created_at
receiver_id
(1, V)
sender_id
Message
In
(1, V)
id
text
timestamp


User
PK id
username
password
email
Friend Request
PK id
status
sent_at
responded_at
FK sender_id
FK receiver_id
Chat_User
PK,FK1 user_id
PK,FK2 chat_id
Chat
PK id
name
created_at
Message
PK id
text
timestamp
FK sender_id
FK receiver_id
chat_id


User
id : number
username : string
password : string
email : string
Friend Request
id: number
status : string
sent_at : date
responded_at : date
sends_receives
messages
0..N
friend request
1
sent by /received by user
Message
id: number
text : string
timestamp : date
belongs to
1
chat
Chat
id: number
name : string
created_at : date