Server Documentation

Justin CHALEARD
2023/12/22

Summary

1. Introduction

Purpose

The purpose of this document is to provide comprehensive documentation for the server component of the chat application. It includes details about the system architecture, class diagram, database schema, server operations, error handling, security considerations, configuration, logging, and potential future enhancements.

Scope

This documentation covers the server-side implementation of the chat application, detailing the server's functionality, interactions with clients, database operations, and security measures.

Audience

This document is intended for developers, system administrators, and anyone involved in understanding, maintaining, or extending the functionality of the chat server.

2. System Architecture

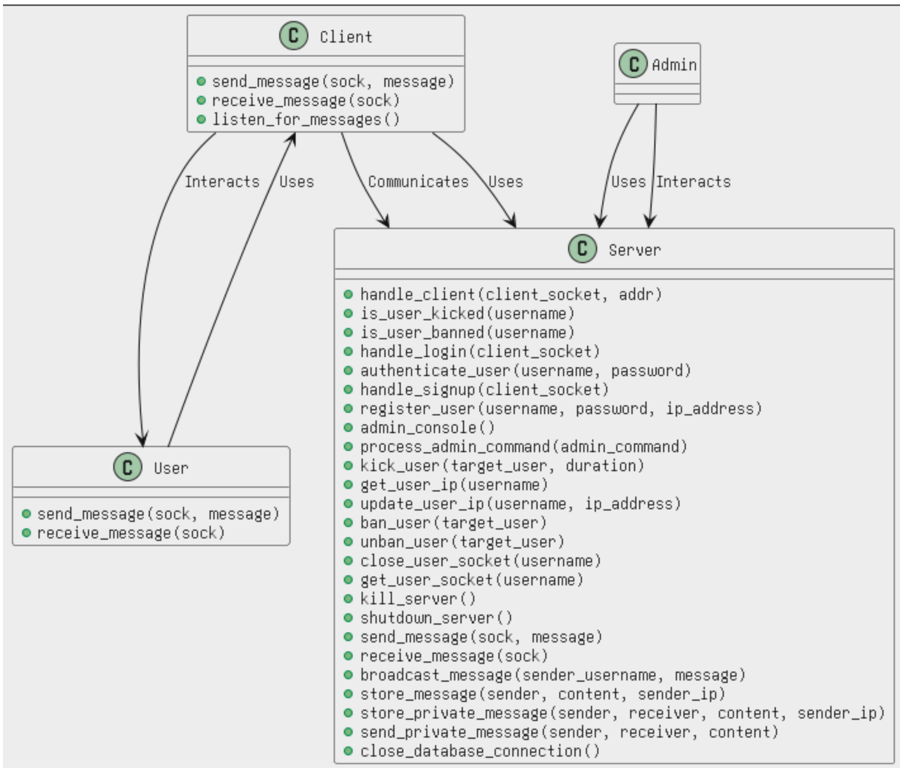
Overview

The chat server follows a client-server architecture, where multiple clients connect to a central server to exchange messages. The server manages user authentication, message handling, and admin commands.

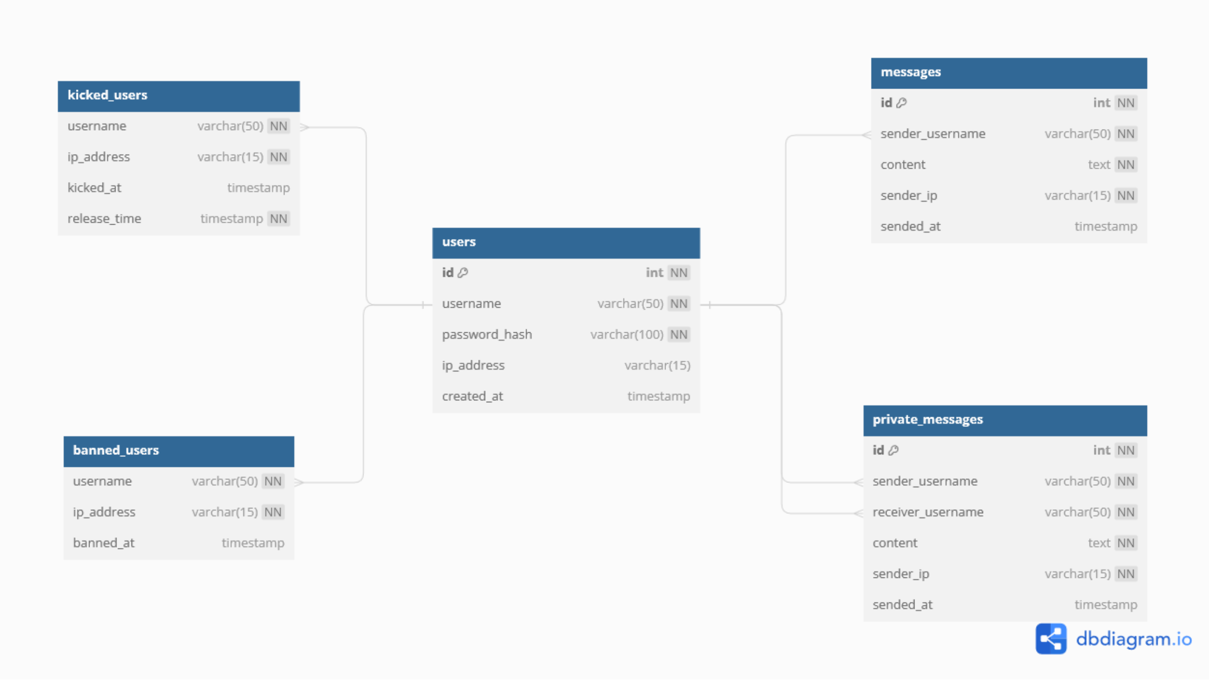
Components

- Server: The main server component responsible for handling client connections, user authentication, message broadcasting, and admin commands.
- User: Represents a user entity with attributes such as username, password, and IP address.
- Client: Represents a connected client with methods for sending and receiving messages.

3. Class Diagram


Client
send_message(sock, message)
receive_message(sock)
listen_for_messages()
Admin
Interacts
Uses
Communicates
Uses
Uses
Interacts
User
send_message(sock, message)
receive_message(sock)
Server
handle_client(client_socket, addr)
is_user_kicked(username)
is_user_banned(username)
handle_login(client_socket)
authenticate_user(username, password)
handle_signup(client_socket)
register_user(username, password, ip_address)
admin_console()
process_admin_command(admin_command)
kick_user(target_user, duration)
get_user_ip(username)
update_user_ip(username, ip_address)
ban_user(target_user)
unban_user(target_user)
close_user_socket(username)
get_user_socket(username)
kill_server()
shutdown_server()
send_message(sock, message)
receive_message(sock)
broadcast_message(sender_username, message)
store_message(sender, content, sender_ip)
store_private_message(sender, receiver, content, sender_ip)
send_private_message(sender, receiver, content)
close_database_connection()

4. Database Schema



kicked_users		
username	varchar(50)	NN
ip_address	varchar(15)	NN
kicked_at	timestamp	
release_time	timestamp	NN

banned_users		
username	varchar(50)	NN
ip_address	varchar(15)	NN
banned_at	timestamp	

users		
id	int	NN
username	varchar(50)	NN
password_hash	varchar(100)	NN
ip_address	varchar(15)	
created_at	timestamp	

messages		
id	int	NN
sender_username	varchar(50)	NN
content	text	NN
sender_ip	varchar(15)	NN
sended_at	timestamp	

private_messages		
id	int	NN
sender_username	varchar(50)	NN
receiver_username	varchar(50)	NN
content	text	NN
sender_ip	varchar(15)	NN
sended_at	timestamp	

5. Server Operations

Handle Client Connection

The server handles incoming client connections, initiates the authentication or registration process, and manages user events.

User Authentication

The server authenticates users based on provided credentials and manages user sessions.

Message Handling

The server processes public and private messages, broadcasts public messages, and sends private messages. To send a private message the user simply have to add /pm *"user"* before the message he want to send

Admin Commands

The server supports admin commands for kicking, banning, and shutting down the server.

```
/kick john 60 → kick the user "john" for 60 seconds
```

```
/ban john → permanently ban the user "john"
```

```
/unban john → remove the ban for the user "john"
```

```
/kill → warn all the connected users of the server shutdown and shut the
server down after a set timer
```

6. Security Considerations

User Authentication

The server uses bcrypt for secure password hashing during user authentication.

Admin Commands

Admin commands are restricted to locally authorized personnel.

7. Server Configuration

Host and Port

The server's host and port configuration can be modified in the code for deployment.

```
python
# Example code snippet
HOST = '127.0.0.1'
```

```
PORT = 12345
```

Database Connection

Database connection details can be configured in the code.

```
python
# Example code snippet
DB_CONFIG = {
    "host": "127.0.0.1",
    "user": "admin",
    "password": "toto",
    "database": "chat_app",
}
```

8. Logging

Server Logs

The server logs important events, such as client connections, authentication, and message handling.

Error Logs

Error logs capture details about unexpected errors, helping in debugging and issue resolution.

9. Future Enhancements

- Add a TLS encryption
- Add a channel system
- Enhance message handling and support multimedia messages.

10. Conclusion

This documentation provides an in-depth understanding of the chat server's architecture, functionality, and key components. It serves as a valuable resource for developers and administrators involved in maintaining and extending the chat application.