University of East London

Ain Shams University

Faculty of Engineering

AIN SHAMS UNIVERSITY

FACULTY OF ENGINEERING

CREDIT HOURS ENG. PROGRAM

Computer engineering and software systems

Phase 1: CSE 351

Project

Peer-to-Peer Multi-User Chatting Application

Submitted to:

Prof. Ayman M. Bahaa-Eldin

Submitted by:

Mohamed Mostafa Bedair El Maghraby	20p7732
Malak Mohamed Mahfouz Mohamed Sadek	20P7813
Mohamed Hesham El Said Zidan	20p7579
Ahmed Saif Elsayed Ibrahim Soliman	20P7668

Contents

Project Goal:

This project aims to develop a robust, user-centric, and scalable Peer-to-Peer Multi-User Chatting Application using Python and sockets. Focused on text-based communication, to emulate features akin to platforms like Clubhouse, emphasizing secure user authentication, efficient client-server communication, and to implement versatile chat functionalities, including group and one-to-one messaging. Additionally, the project aims to implement user-friendly features such as message formatting and hyperlink support, ensuring a seamless and visually appealing user experience.

Project Scope:

Core functionalities include user authentication, basic client-server setup, chat room features, group messaging, one-to-one chat sessions, message formatting, error handling, and UI enhancements. The project will adhere to a command-line interface for simplicity, utilize color-coded messages, and focus on documentation to facilitate easy installation, configuration, and usage. Robust error handling, resilience to network interruptions, and scalability to handle increasing user loads will be integral to the application's success.

List of Functionalities:

1. User Authentication:

- User should be able to create an account using a unique username and password or log in to an existing account if one exists.
- User should have his unique usernames used as his identifier for other users.
- User should be able to change his username if desired.

2. Basic Client-Server Setup:

- Users should be able to connect to the server using a client application.
- Users should be able to see a list of all the currently active users.

3. Chat Room Functionality:

- Users should be able to create or join a chat room.
- Users should be able to see a list of available chat rooms.
- The user that creates the chat room should be the group leader/admin.
- A Chat room admin should be able to remove other participants from the chat.
- When a chat room admin leaves the chat room, the second oldest member in the chat room should become the new admin.
- A chat room admin should be able to make other participants admins.

4. Group Messaging in Chat Rooms:

- Users should be able to send/see a message to/from everyone in the chat room.
- Users should receive notifications for new messages.
- Users can turn off notifications/mute from a chat room.

5. One-to-One Chat Functionality:

- Users can initiate one-to-one chat sessions, in which users can send/receive messages with only one user.
- Users should receive notifications for new private messages.
- Users can turn off notifications/mute from a user in private chat.

6. Message Formatting and Features:

- The application supports basic text formatting (e.g., bold, italics) in messages.
- Requirement: Users can share hyperlinks in messages, which opens a browser and redirects a user that clicked on it to a certain URL.

7. Error Handling and Resilience:

- The application implements robust error handling for unexpected scenarios. (for ex: invalid command or character entered)
- User should receive meaningful error messages for troubleshooting.
- Users are automatically reconnected in case of a network interruption.

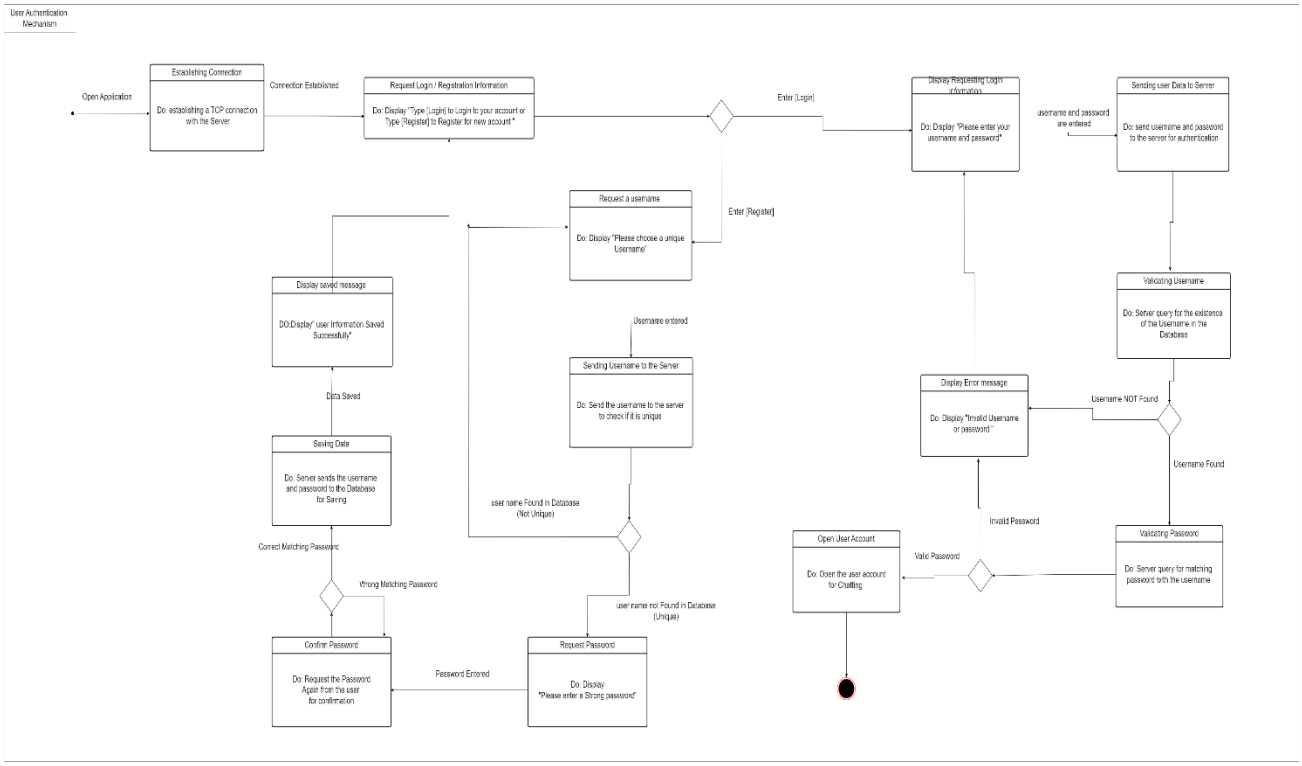
8. User Interface (UI) Enhancements:

- The application implements a clean command line user interface, utilizing color-coded messages.
- User can identify different types of messages, for example usernames appears in a certain colour, notifications appear in a different colour and font, connection errors appear in a third different way. Etc

System Diagrams

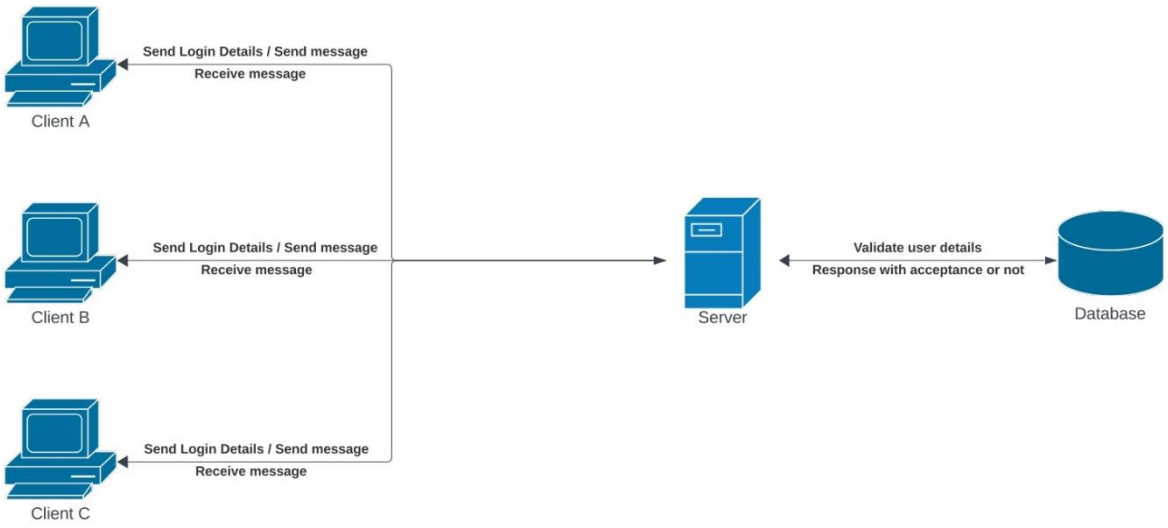
User Authentication Mechanism State Diagram:

Below is a state Diagram showing the states of the system to authenticate the user either by logging in or registration.


User Authentication Mechanism
Establishing Connection
Do: establishing a TCP connection with the Server
Open Application
Connection Established
Request Login / Registration Information
Do: Display "Type [Login] to Login to your account or Type [Register] to Register for new account "
Enter [Login]
Display Requesting Login information
Do: Display "Please enter your username and password"
username and password are entered
Sending user Data to Server
Do: send username and password to the server for authentication
Enter [Register]
Request a username
Do: Display "Please choose a unique Username"
Display saved message
DO:Display" user Information Saved Successfully"
Data Saved
Saving Data
Do: Server sends the username and password to the Database for Saving
Correct Matching Password
Wrong Matching Password
Confirm Password
Do: Request the Password Again from the user for confirmation
Password Entered
Request Password
Do: Display "Please enter a Strong password"
Username entered
Sending Username to the Server
Do: Send the username to the server to check if it is unique
user name Found in Database (Not Unique)
user name not Found in Database (Unique)
Validating Username
Do: Server query for the existance of the Username in the Database
Username NOT Found
Display Error message
Do: Display "Invalid Username or password "
Username Found
Invalid Password
Validating Password
Do: Server query for matching password with the username
Valid Password
Open User Account
Do: Open the user account for Chatting

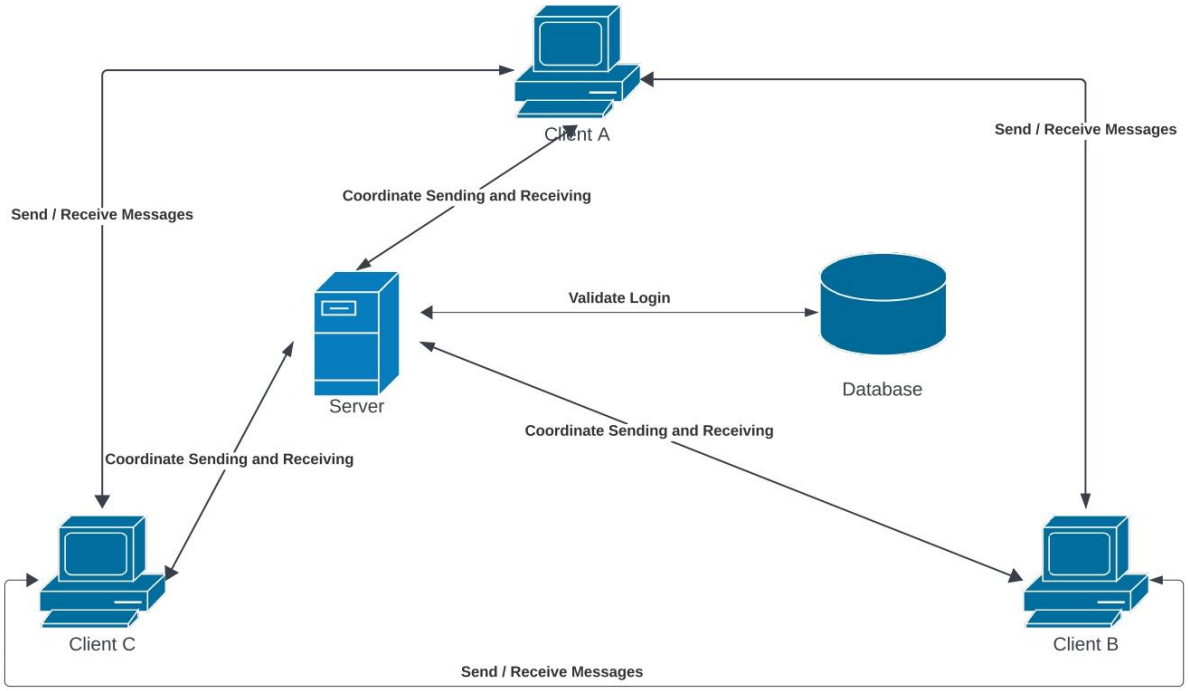
System Architecture Diagram:

1.Client server:



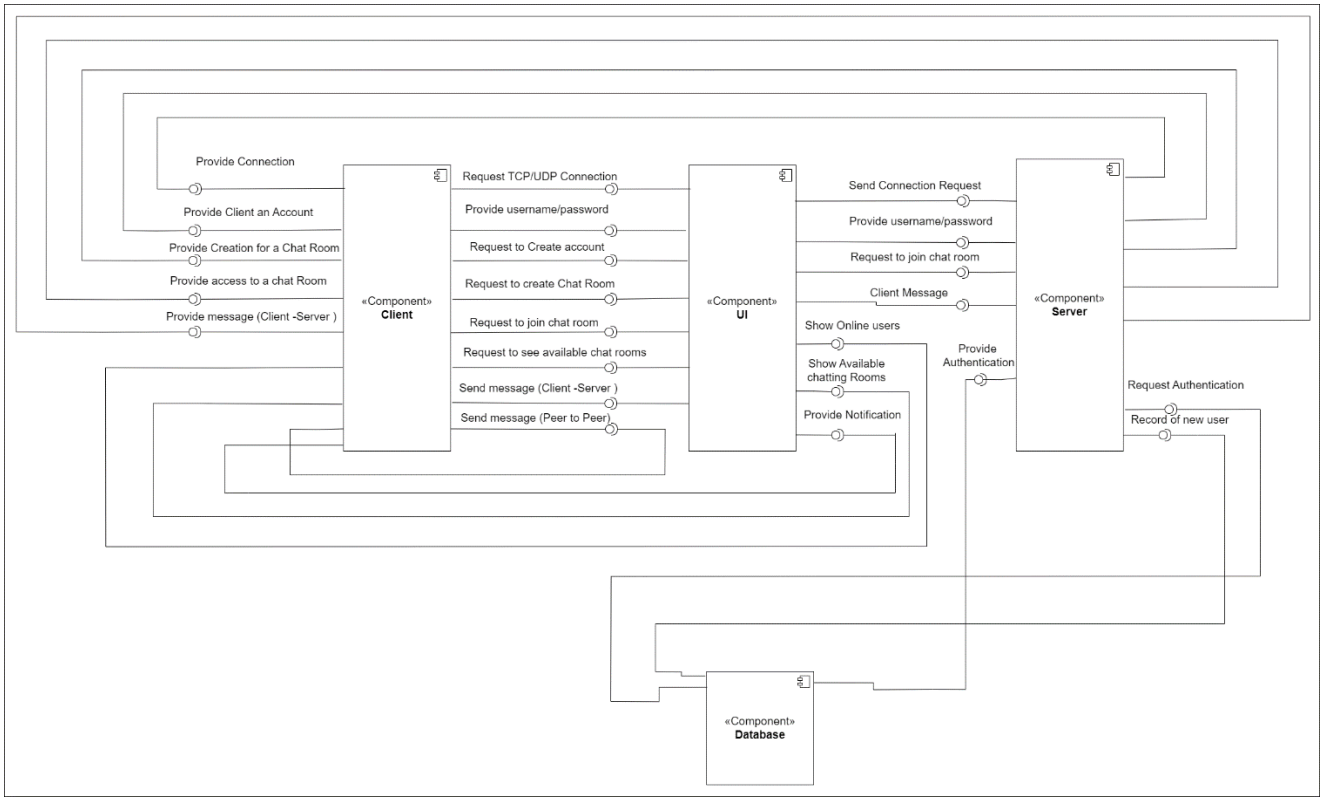
The way a server provides resources and services to one or more clients is described by the client-server model. We have clients (a, b and c) and it validates user details the same way as peer to peer but in client server the user sends the messages to the server with the receiver name that will receive the message. The connection in client server is between them and the server not between them and each other's like peer-to-peer

2.Peer to peer:



A peer-to-peer (P2P) network is created when two or more PCs are connected and share resources without going through a separate server computer. Clients (a, b and c) send requests to connect with the server to validate the user details with database then it will send and receive the messages with each other's and the server can act as a tracker to coordinate the sending and receiving (optional)

Component Diagram:



Communication Protocols:

Note: These protocols are still under development which means it can be added or modified later along the project.

We've created 12 communication protocol, 4 for requests, 4 for successful response messages, and 4 for failure response messages.

Request Protocols:

1.Login & Registration protocol

```
Title:[Login,Registration]
Connection:TCP
send From:[IP address,PortNo]
Send to:[Server IP address, PORT_NO]
Data:[Username,Password]
```

This protocol is for login & registration requests; therefore, login and registration methods are used in the field "title" and the connection used is TCP to make sure credentials are correctly sent.

Data of sender, IP address and port number, are sent to the recipient server. (Identified using its IP address and port number).

The content of the message itself is the user's credentials. (username & password)

2.Chatroom protocol

```
Title:[ChatRoom]
Connection:TCP
send From:[IP address,Port_No]
Send to:[IP address, PORT_NO]
Data:[[CREATE,CHATROOM_NAME],[JOIN,CHATROOM_NAME],SEE,QUIT]
```

This protocol is for chat room operations. Once again, TCP connection is used to make sure credentials are correctly sent.

Data of sender, IP address and port number, are sent to the recipient server. (Identified using its IP address and port number).

The content of the message itself is the desired method on chat room.

3.One-to-One protocol

```
Title:[PRIVATE]
connection:[TCP,UDP]
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
Data:[USERNAME]
```

Connection type for this protocol is chosen by the user as TCP or UDP depending on his needs and preferences.

Data of sender, IP address and port number, are sent to the recipient server. (Identified using its IP address and port number).

The content of the message itself is the desired method on the private chat.

4.Messaging protocol

```
Title:[MESSAGE]
connection:CONNECTION OF PREVIOUS TITLE
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
Data:[MESSAGE_CONTENT,FORMATTING:[BOLD,ITALIC,HYPERLINK]]
```

Connection used in this protocol will depend on the connection established in the chatroom or private messaging protocol.

Data of sender, IP address and port number, are sent to the recipient server. (Identified using its IP address and port number).

The content of the message itself is the text written and its desired formatting.

Response protocols:

Note: **Even** status codes resemble success while **odd** status codes resemble failure.

Status:

10X -> Response to Login & Registration protocol

20X -> Response to Chatroom protocol

30X -> Response to One-to-One protocol

40X -> Response to Messaging protocol

```
STATUS:101
Connection:TCP
send From:[IP address,PortNo]
Send to:[Server IP address, PORT_NO]
STATUS_PHRASE:[LOGIN_FAILED,USERNAME_TAKEN,
               INVALID_COMMAND]
```

```
STATUS:100
Connection:TCP
send From:[IP address,PortNo]
Send to:[Server IP address, PORT_NO]
STATUS_PHRASE:[LOGIN_SUCCESSFUL,REGISTRATION_SUCCESSFUL]
```

```
STATUS:201
Connection:TCP
send From:[IP address,Port_No]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[CREATION_FAILIURE,JOIN_FALIURE
               ,NO_AVAILABLE_CHAT_ROOM,
               INVALID_COMMAND]
```

```
STATUS:200
Connection:TCP
send From:[IP address,Port_No]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[CREATION_S,JOIN_S,ROOM_LISTED_S,QUIT_S]
```

```
STATUS:301
connection:[TCP,UDP]
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[USERNAME_NOT_FOUND,
               USERNAME_OFFLINE,
               INVALID_COMMAND]
```

```
STATUS:300
connection:[TCP,UDP]
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[PRIVATE SESSION INTIATED]
```

```
STATUS:401
connection:CONNECTION OF PREVIOUS TITLE
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[INVALID_FORMAT,INVALID_COMMAND]
```

```
STATUS:400
connection:CONNECTION OF PREVIOUS TITLE
send From:[IP address,PortNo]
Send to:[IP address, PORT_NO]
STATUS_PHRASE:[MESSAGE_SENT]
```

Status represents whether the request resulted in success or failure. Status phrase indicates success or the type of failure if occurred.