

**AIN SHAMS UNIVERSITY**

**FACULTY OF ENGINEERING**

**CREDIT HOURS ENG. PROGRAM**

**Computer engineering and software systems**

**Ain Shams University**

**Faculty of Engineering**

**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** |

# 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

# Communication Protocols:

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

# Request Protocols:

## Login & Registration protocol

A screen shot of a computer

Description automatically generated

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)

## Chatroom protocol

A computer code with black text

Description automatically generated

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.

## Private Messaging protocol

A close-up of a computer screen

Description automatically generated

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.

## Messaging protocol

A black text on a white background

Description automatically generated

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 represents whether the request resulted in success or failure.

Status phrase indicates success or the type of failure if occurred.