

**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** |

# Introduction

Just to Remember from phase 1 we wanted to implement Peer-to-Peer Multi-User Chatting Application that has a lot of functionalities mentioned in Phase1.

In this Phase we will discuss implementing The server and the Client and how they interact with each other and how the server handle multiple clients. In addition, we will show how the server authenticate the Client using A TCP Connection with the client through the database.

All of this will be made using python and sockets. A socket is a communications connection point (endpoint) that you can name and address in a network. Socket programming shows how to use socket APIs to establish communication links between remote and local processes.

The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and network applications. Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralized data. Socket application program interfaces (APIs) are the network standard for TCP/IP. A wide range of operating systems support socket APIs. i5/OS sockets support multiple transport and networking protocols. Socket system functions and the socket network functions are thread safe.[2]

# Server

A server is a computer program or device that provides a service to another computer program and its user, also known as the client. In a data center, the physical computer that a server program runs on is also frequently referred to as a server. That machine might be a dedicated server or it might be used for other purposes.

In the client/server programming model, a server program awaits and fulfils requests from client programs, which might be running in the same, or other computers. A given application in a computer might function as a client with requests for services from other programs and as a server of requests from other programs.[1]

In this case we will implement the server using Socket programming using python programming language, that is capable of handling multiple clients Simultaneously.

## Server functions:

### Main:

First thing we must import the needed libraries for this task:

1. Socket lib -> for establishing socket connection.
2. Threading lib -> for handling multiple clients
3. Hashlib lib -> for hashing the password before sending it to the database.
4. Sqlite3 -> for connecting with the database and write SQL queries.

A black screen with a black background

Description automatically generated

Then we must specify the address of the of the server and its port number. Then we establish a TCP connection with the server using a socket and we bind the host with the port number and make the server start Listening.

A computer screen with white text

Description automatically generated

Then we make an empty dictionary for the clients to store the client object as a key and for each client object their will be a value of List contain the unique username and the nickname of the client.



Then we will print a “server is listening…” to know that the server is up and Running.

Finally a while Loop is made and we will state inside it the **server.accept()** method so we keep accepting clients and we create a thread that will run the **Handle\_Client()** function separately for each accepted client without affecting the server for accepting other clients.

A computer screen with text

Description automatically generated

### Handle\_Client():

1. Function Parameters:
   1. Client object
   2. Address of the client
2. Function Description:

This function is responsible for handling all the client issues starting from logging in and registration ending with allowing the user to chat in a chatting Room or one to one chatting or even if the client suddenly disconnected from the server.

First thing the function calls **Login\_or\_register()** function (navigate to it’s functionality through the table of contents ) , to know the choice of the user whether to login or register , etc. Then it returns a username of the client to be added to the clients dictionary and be able to enjoy the functionalities of the application.

Then it takes a nickname from the client and broadcast to all the other online clients that this client is Online and has joined the server through Broadcast() function (navigate to it’s functionality through the table of contents ).

In Case of, the client disconnected in an inappropriate way the server will print that this client has disconnected and if the client was logged in it will broadcast that the client has left the server and he is now offline and then will remove him from the dictionary of the clients and the server will close the connection with this specific client. However if the client was not logged in the server will only close the connection with this client.

1. Function snippet code:

A screen shot of a computer code

Description automatically generated

### Login\_or\_register():

1. Function Parameters:
   1. Client Object
2. Function Description:

This function Displays the options for the client to either login or register or Close the Application.

In Case of choosing to login, the server will request the username and password of the client and then will authenticate them using **Client\_authentication()** function (navigate to it’s functionality through the table of contents ) , if one of them is wrong it will display a message “Wrong user name or password ” and will let the user to choose the command again , however if the username and password are correct it will login the client to server and show the client the list of features that could be done in the application.

In Case of choosing to Register, The server will request a unique Username from the Client and it will check whether it is unique or not through **is\_unique()** function (navigate to its functionality through the table of contents ). If the username is unique, the server will the let the user to choose his password however if the password is weak, the server will request a strong password, and if it is not weak the new username and password will be saved in the database and the client will be redirected to the Login Options again to choose from it. However, if the username is not unique the server, the server will display “This username has been taken” and will let the user to enter a unique username again.

In Case of choosing an invalid command, the server will display “Please enter A valid command” and will take another command from the client.

1. Function snippet code:

A computer screen with colorful text

Description automatically generated A computer screen shot of a program code

Description automatically generated

### Client\_authentication():

1. Function Parameters:
   1. Username of the Client
   2. Password of the Client
2. Function Description:

This Function connects with the Database and hash the password and then query for the existence of this username with that specific hashed password.

In case it is found it returns True, Otherwise Returns False.

1. Function snippet code:

A screen shot of a computer code

Description automatically generated

### Client\_Registration():

1. Function Parameters:
   1. Client object
   2. Unique Username
2. Function Description:

This function is used after we made sure that the username the client entered is unique using **is\_unique()** function (navigate to it’s functionality through the table of contents ) then the server requests from the user a strong password and the server validates whether the password is strong or not through **is\_strong()** function (navigate to it’s functionality through the table of contents ) after the validation, the server saves the information of the new Account to the Database using **add\_new\_user()** function (navigate to it’s functionality through the table of contents ) and then the server takes the next command from the user and return it to **Login\_or\_Registration()** function .

1. Function snippet code:

A computer screen shot of a program code

Description automatically generated

### add\_new\_user():

1. Function Parameters:
   1. Unique Username
   2. Hashed Password
2. Function Description:

This function connects to the Database and makes an Insert Query to it for the username and password.

1. Function snippet code:

A computer screen shot of a person

Description automatically generated

### is\_unique():

1. Function Parameters:
   1. Username
2. Function Description:

This Function connects with the database and query for the existence of such username in the database, if it is Found the function returns True , otherwise it returns False.

1. Function snippet code:

A screen shot of a computer

Description automatically generated

### is\_strong():

1. Function Parameters:
   1. Client Object
   2. password
2. Function Description:

This function see the length of the password, if it is less than 5 characters, the server notifies the client that this is a weak password , and the server will loop on the password until the client enter a strong one and the function then return the strong password.

1. Function snippet code:

A black screen with white text

Description automatically generated

### Broadcast():

1. Function Parameters:
   1. Message
2. Function Description:

This function aims on looping upon the clients a send them all the message sent to this function.

1. Function snippet code:

A black screen with white text

Description automatically generated

# Client

## Client Functions:

### Main():

First thing we will import the necessary libraries for running the client code

1. Socket
2. threading

A screenshot of a computer program

Description automatically generated

Then we make a client object using socket library establishing a TCP connection, then we specify the address and the port number of the server to be connected with.

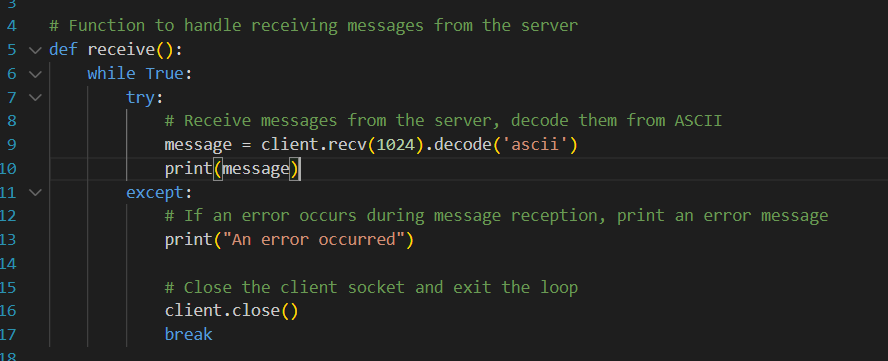
Finally, we will make two threads, one for the **receive** function and the other one for the **write** function. Noticing that we will put all of this code inside try except statement in case the client code is running without a server to catch this exception.

### Receive():

1. Function Parameters:
   1. NULL
2. Function Description:

This function consists of a while loop containing the receiving function from the server to keep receive messages or commands from the server and print this message being Received for the Client.

1. Function snippet code:



### Write():

1. Function Parameters:
   1. NULL
2. Function Description:
3. Function snippet code:

# Repository Link:

<https://github.com/MHZDN/-P2P-Multi-User-Chatting>

# References

1. [https://www.techtarget.com/whatis/definition/server [1](https://www.techtarget.com/whatis/definition/server%20%5b1)]
2. <https://www.ibm.com/docs/en/i/7.1?topic=communications-socket-programming> [2]