

Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering



Computer Network Lab (CO3094)

Assignment Report

P2P Chat Application

Lecturer: Nguyễn Lê Duy Lai

Group name: Ignorant Spider

Team members: Trương Tân Hào Hiệp – 2011211

Ho Chi Minh city, 10 December, 2022



Table of Contents

PHASE 1	3
1.1. Architectural Approach	3
1.1.1. <i>Conceptual Solution.....</i>	<i>3</i>
1.1.2. <i>Architectural Design.....</i>	<i>3</i>
1.2. Requirement Elicitation	3
1.2.1. <i>Server – Client.....</i>	<i>3</i>
1.2.2. <i>Peer – Peer.....</i>	<i>4</i>
1.3. Protocol Design.....	4
1.3.1. <i>Sign-up</i>	<i>4</i>
1.3.2. <i>Login.....</i>	<i>5</i>
1.3.3. <i>Logout.....</i>	<i>6</i>
1.3.4. <i>Create chat session</i>	<i>7</i>
PHASE 2	8
2.1. Setting up.....	8
2.2. Overview	8
2.3. Implementation.....	9
2.3.1. <i>Start the central server</i>	<i>9</i>
2.3.2. <i>Initiate the connection from a client to the central server.....</i>	<i>9</i>
2.3.3. <i>Login/ Signup.....</i>	<i>10</i>
2.3.4. <i>Initiate a chat session between peers.....</i>	<i>11</i>
END OF REPORT!!!.....	12

PHASE 1

1.1. Architectural Approach

1.1.1. Conceptual Solution

In this application, **every client** (with a unique username) must connect to a **main server** before establishing any connection to another peer **in their friend list**. The connection between both Client – Server and Peer – Peer will be using TCP/IP protocol.

1.1.2. Architectural Design

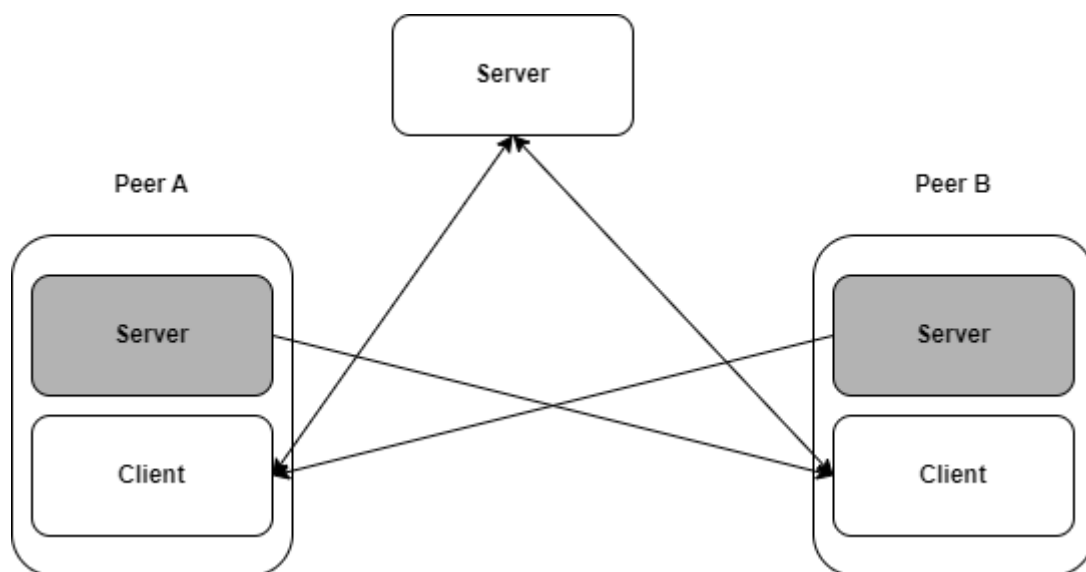


Figure 1. A three-way handshake protocol.

1.2. Requirement Elicitation

1.2.1. Server – Client

– Server:

- + Receive login/ sign-up request from client.
- + Manage the list of users and their status.
- + Update the status of recently online users to their friends.
- + Return the friend list of the user after successful login attempt.

– Client:

- + Send login/ sign-up request to server.
- + Create a chat session with a friend using the IP address in the friend list received from server.

1.2.2. Peer – Peer

- Send message to another peer.
- Transfer one or many files to another peer.

1.3. Protocol Design

1.3.1. Sign-up

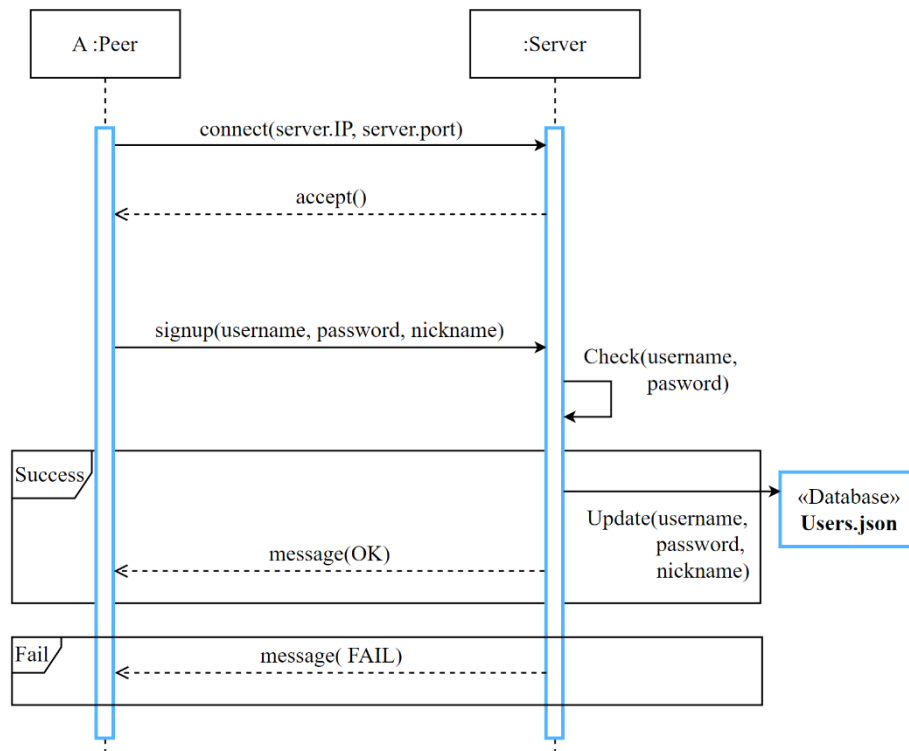


Figure 2. Sign-up protocol.

Use Case Name	Sign-up
Description	A client wants to register with the main server as a new user.
Actor(s)	Peer A, Server.
Trigger	Peer A sent the signup request to the Server.
Precondition(s)	Peer A must be connected to the Server.
Postcondition(s)	Peer A successfully create an account in the server.
Main flow	<ol style="list-style-type: none"> 1. Server check the validity of the info sent by Peer A (whether the username is unique or not). 2. The username is unique; the Server will acknowledge Peer A as a new user and update the information to the database (hardcoded in .json file in this case). 3. Server send a “success” notification to Peer A.

Exception flow	2a. The username is not unique , Server send a “failed” notification to Peer A. <i>End use case.</i>
----------------	---

1.3.2. Login

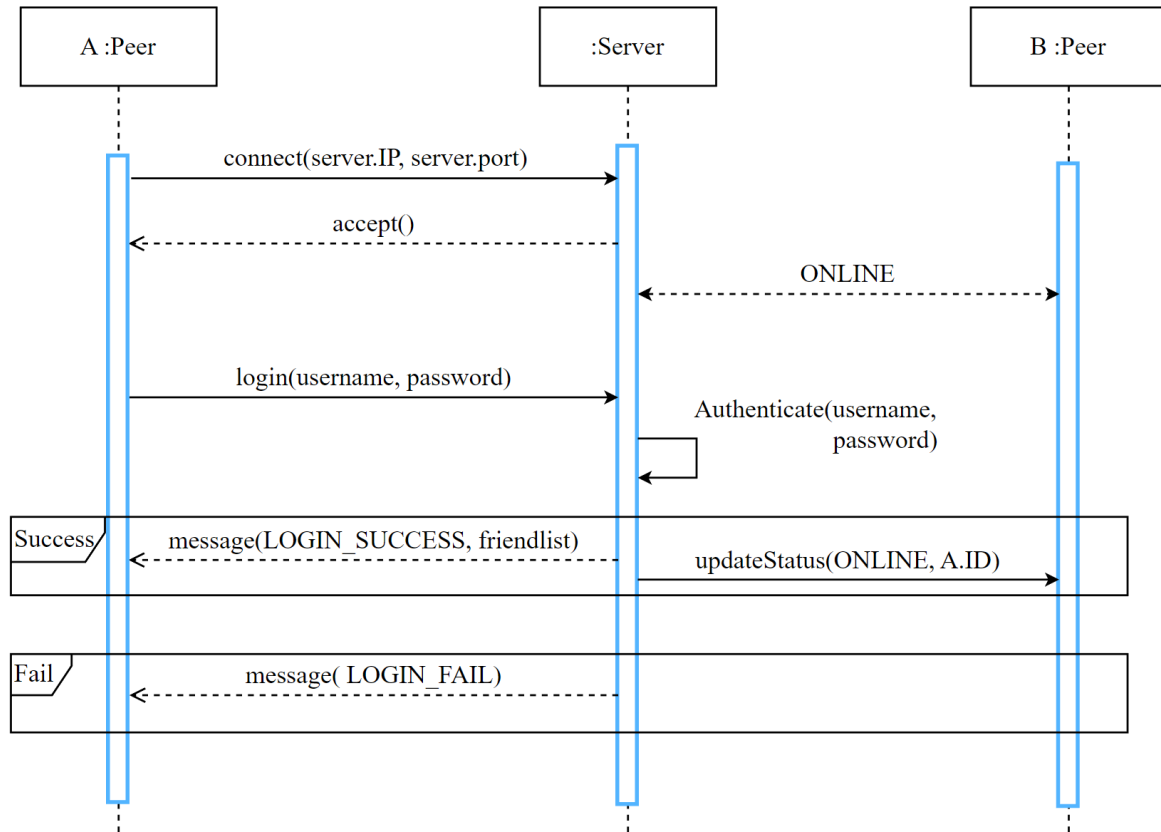


Figure 3. Login protocol.

Use Case Name	Login
Description	A client wants to login to the main server.
Actor(s)	Peer A, Server, Peer B.
Trigger	Peer A sent the login request to the Server.
Precondition(s)	Peer A must be connected to the Server. Peer B is friends of A and must be ONLINE.
Postcondition(s)	Peer A successfully login to the Server and retrieve the friend list.
Main flow	<ol style="list-style-type: none"> 1. Server authenticate the info sent by Peer A (whether the username and password is correct or not). 2. The info is eligible; the Server will send a “success” notification to Peer A together with the user’s friend list. 3. Server will update the status (as “ONLINE”) and user’s address to every ONLINE peer in the user’s friend list.

Exception flow	2a. The info is <i>ineligible</i> , Server send a “failed” notification to Peer A. End use case.
----------------	---

1.3.3. Logout

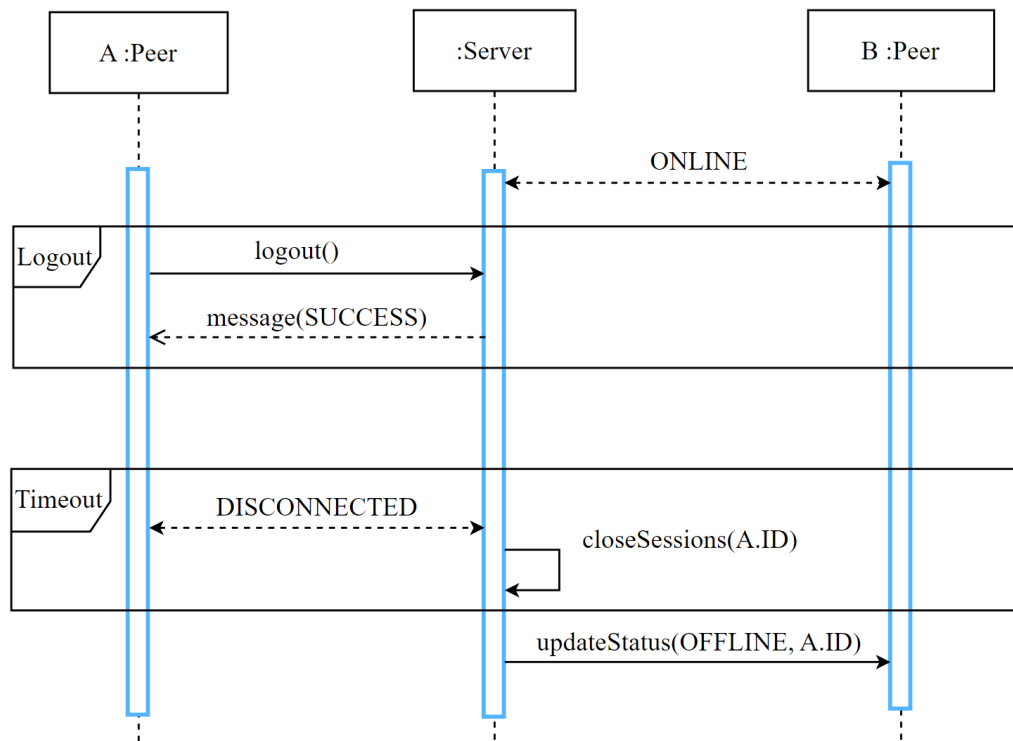


Figure 4. Logout protocol.

Use Case Name	Logout
Description	A client wants to logout from the main server.
Actor(s)	Peer A, Server, Peer B.
Trigger	Peer A sent the logout request to the Server or Session timeout.
Precondition(s)	Peer A must login to the Server. Peer B is friends of A and must be ONLINE.
Postcondition(s)	Peer A successfully logout and close the chat session.
Main flow	<ol style="list-style-type: none"> 1. Peer A send the logout request to the Server. 2. Server accept and send “success” notification to Peer A. 3. Server close all threads and sessions of the user. 4. Server update status “OFFLINE” of Peer A to other peers in friend list.
Alternative flow	<ol style="list-style-type: none"> 1a. After the TTL (time to live) is up (there is no response from Peer A), all chat sessions will be unavailable. 2a. Server close all threads and sessions of the user. 3a. Server update status “OFFLINE” of Peer A to other peers.

1.3.4. Create chat session

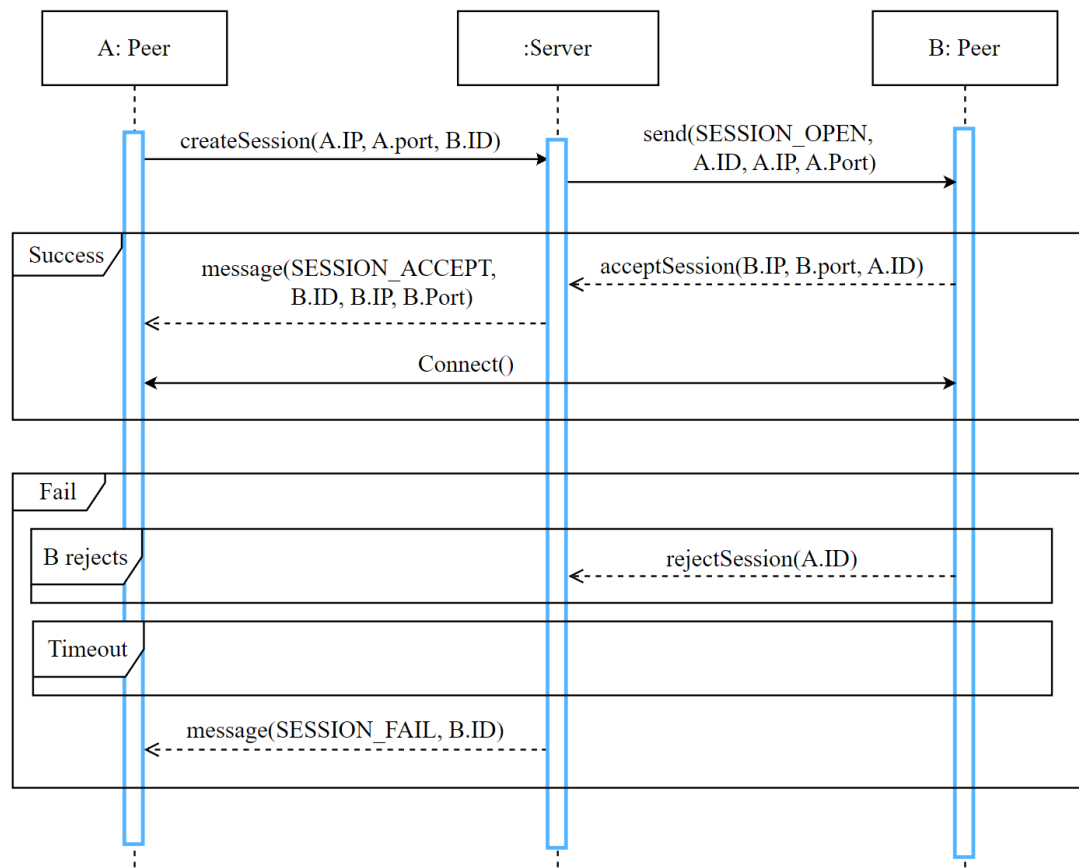


Figure 5. Create chat session protocol.

Use Case Name	Create chat session
Description	A peer wants to create a chat session with another peer.
Actor(s)	Peer A, Server, Peer B.
Trigger	Peer A sent the create chat session request to the Server.
Precondition(s)	Peer A must login to the Server.
Postcondition(s)	Peer A successfully create chat session with Peer B.
Main flow	<ol style="list-style-type: none"> 1. Server sent the request notification from peer A to peer B. 2. Peer B sends response (YES, <i>B.IP</i>, <i>B.port</i>). 3. Server send "session success" notification to peer A. 4. Initiate a chat session between Peer A and Peer B.
Exception flow	<ol style="list-style-type: none"> 2a. Peer B sends response (NO). 3a. Server sends "session fail" notification to peer A. <i>End use case.</i> 2b. There is no response from Peer B (Timeout). 3b. Server sends "session fail" notification to peer A. <i>End use case.</i>

PHASE 2

2.1. Setting up

The source code can be retrieved from our GitHub repository via the following [link](#).

***Note:** It's recommended to use PyCharm to avoid any problems regarding the relative path to open any files used in the code or manually changes the relative path to the absolute path according to your computer directory.

After successfully retrieving the source code, open up the terminal to install the required packages using python “*pip install*” command shown below:

```
pip install -r . / Server / requirements.txt
```

After successfully installing all of the required packages, the application is ready to be launched; however, let's go briefly through all the stuffs in these folders.

2.2. Overview

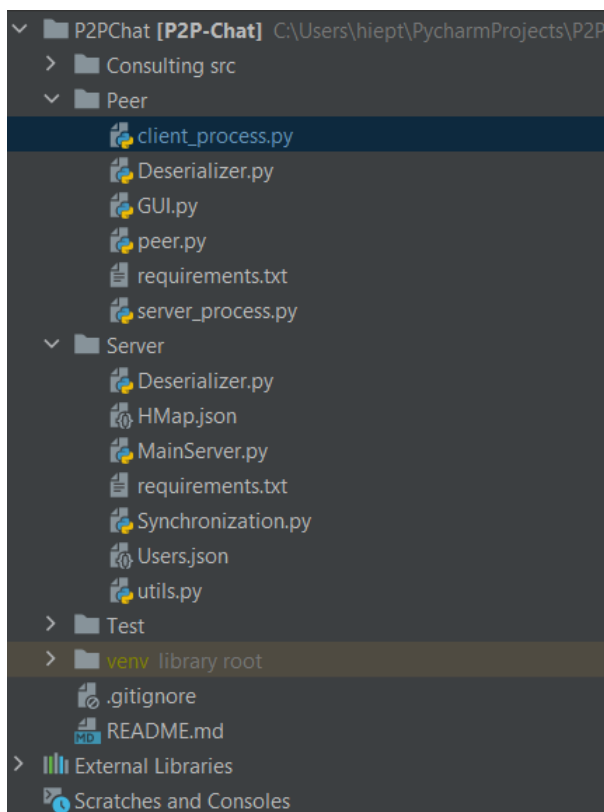


Figure 6. P2P Chat folder Overview

The P2P Chat folder consist of two main parts: **Peer** and **Server**.

In **Peer** folder:

- client_process is used to execute Client–Server and Peer–Peer operations.
- GUI is used to implement the Graphical User Interface for the application.
- server_process acts as a mediator.

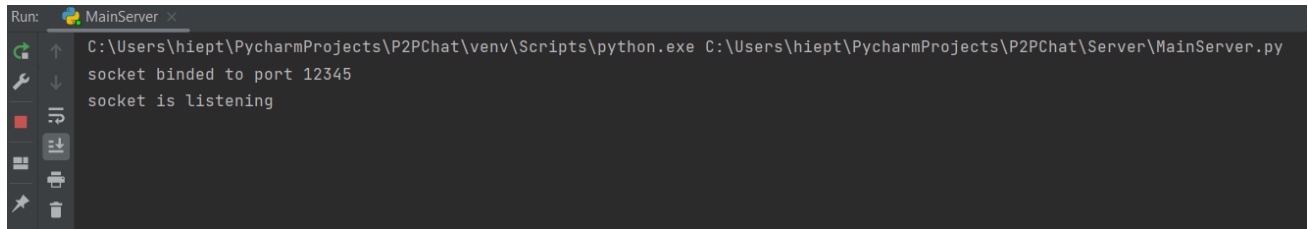
In the **Server** folder:

- MainServer is in charge of all the operations of the central server.
- HMap and Users are the hardcoded data file for storing users' information updated via utils.py.

2.3. Implementation

2.3.1. Start the central server

The first step into this process is to go into the **Server** folder and run **MainServer.py** which will start the central server. Successfully start the server will result in the following prompt.



```
Run: MainServer x
C:\Users\hiept\PycharmProjects\P2PChat\venv\Scripts\python.exe C:\Users\hiept\PycharmProjects\P2PChat\Server\MainServer.py
socket binded to port 12345
socket is listening
```

Figure 7. MainServer started.

2.3.2. Initiate the connection from a client to the central server

Having a central server up and running, the next step is to go into **Peer** folder and run **peer.py** which will initiate a client connection to the central server. A successful connection attempt will result in showing the application interface and the connection prompt from MainServer.

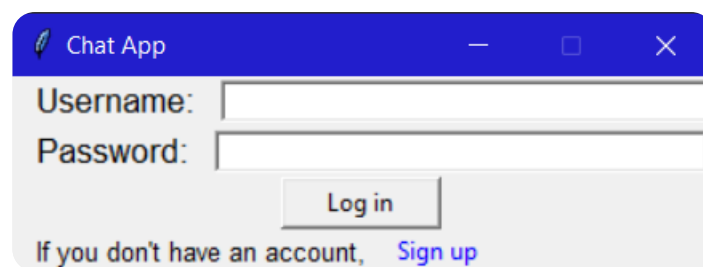
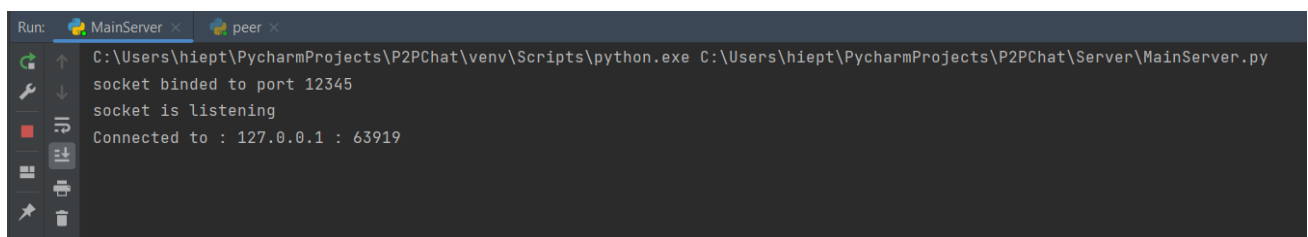


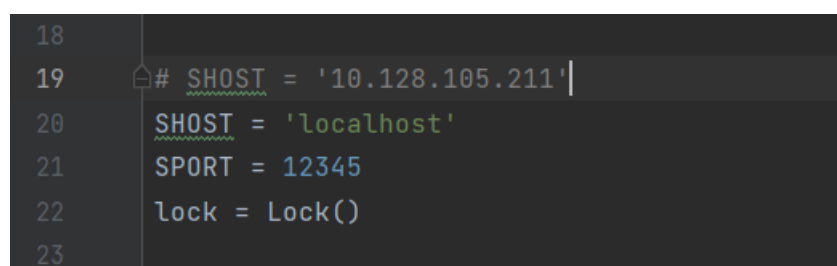
Figure 8. Application Interface.



```
Run: MainServer x peer x
C:\Users\hiept\PycharmProjects\P2PChat\venv\Scripts\python.exe C:\Users\hiept\PycharmProjects\P2PChat\Server\MainServer.py
socket binded to port 12345
socket is listening
Connected to : 127.0.0.1 : 63919
```

Figure 9. Client connected to MainServer.

To connect from another device using the same network, the SHOST must be changed to the IP address of the device hosting the MainServer.



```
18
19 # SHOST = '10.128.105.211'
20 SHOST = 'localhost'
21 SPORT = 12345
22 lock = Lock()
23
```

Figure 10. Change the SHOST value.

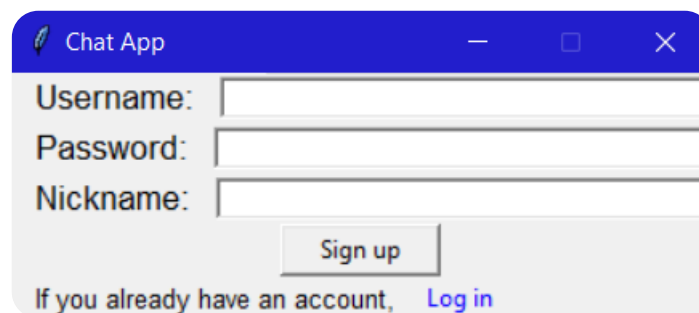
2.3.3. Login/ Signup

Before logging into the server, let's have a look at the **Users.json** file, which contains all the user's registered information, in the **Server** folder.

```
{
  "nickname": "Apple",
  "username": "aaa",
  "password": "aaa",
  "friends": [
    1,
    2
  ]
},
```

Figure 11. Structure of a user's info.

In case of registering as a new user, click on the **Signup** highlighted text that will redirect to the below figure.



The image shows a web interface for a 'Chat App'. It has a blue header bar with the text 'Chat App' and standard window controls. Below the header, there are three input fields labeled 'Username:', 'Password:', and 'Nickname:'. To the right of these fields is a 'Sign up' button. Below the 'Sign up' button, there is a link that says 'If you already have an account, Log in'.

Figure 12. Signup Interface.

After providing the correct username and password, a login success will pop up redirect you to the main interface of the chat application.

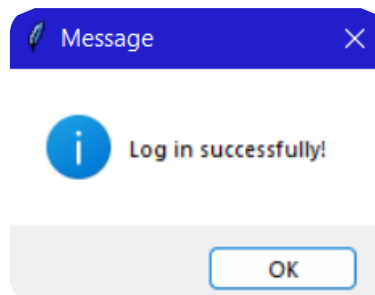


Figure 13. Login success.

In the Main interface, there will be a list of friends and their status, a chat box and log out button. The friend status will be automatically updated if that friend just got online and the same case for offline.

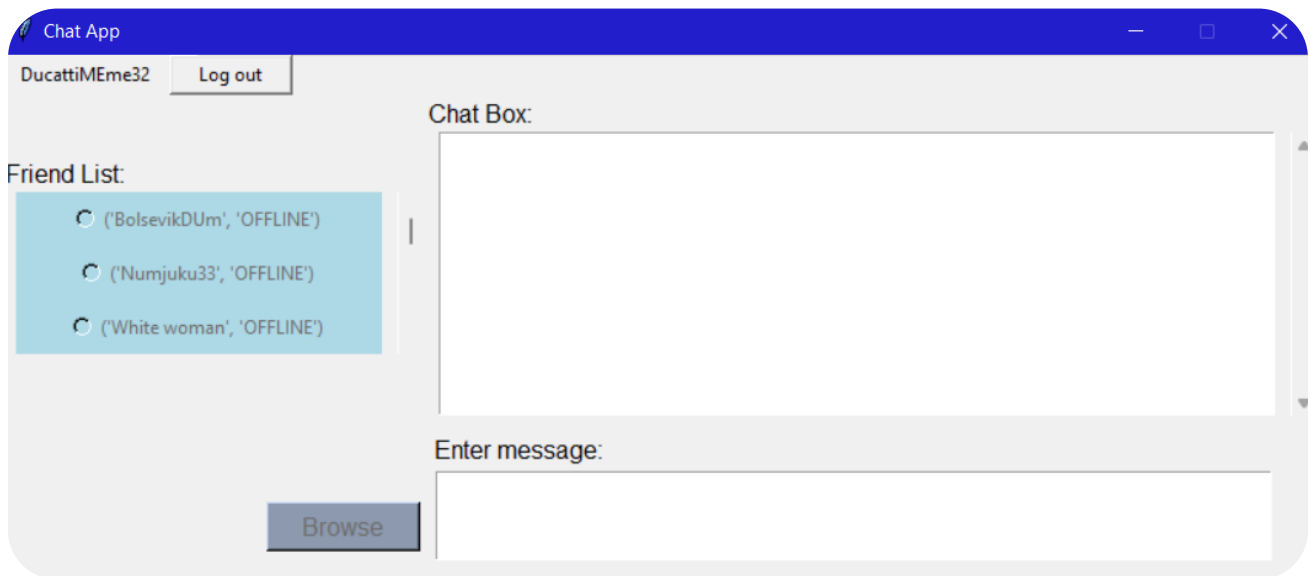


Figure 14. Main Interface.

2.3.4. Initiate a chat session between peers

Choose an online friend to create a chat session with. In this scenario, Peer A is trying to connect with Peer B. On successful attempt, user will be prompted “SESSION INITIATED” and can start sending messages to each other.

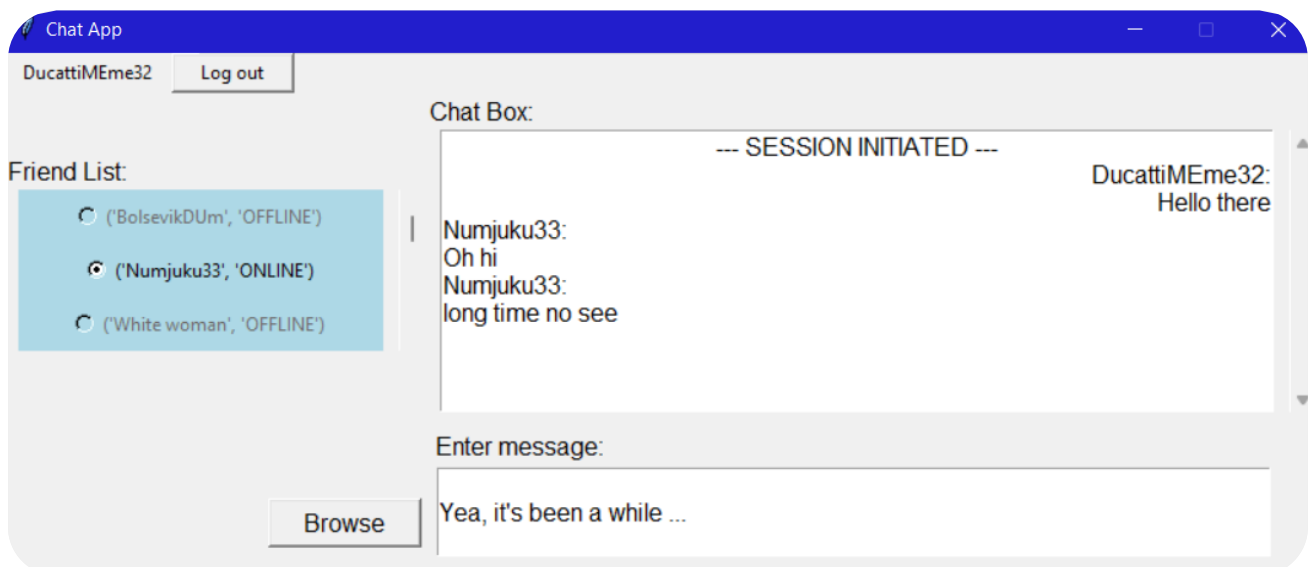


Figure 15. Apple connected to Banana.

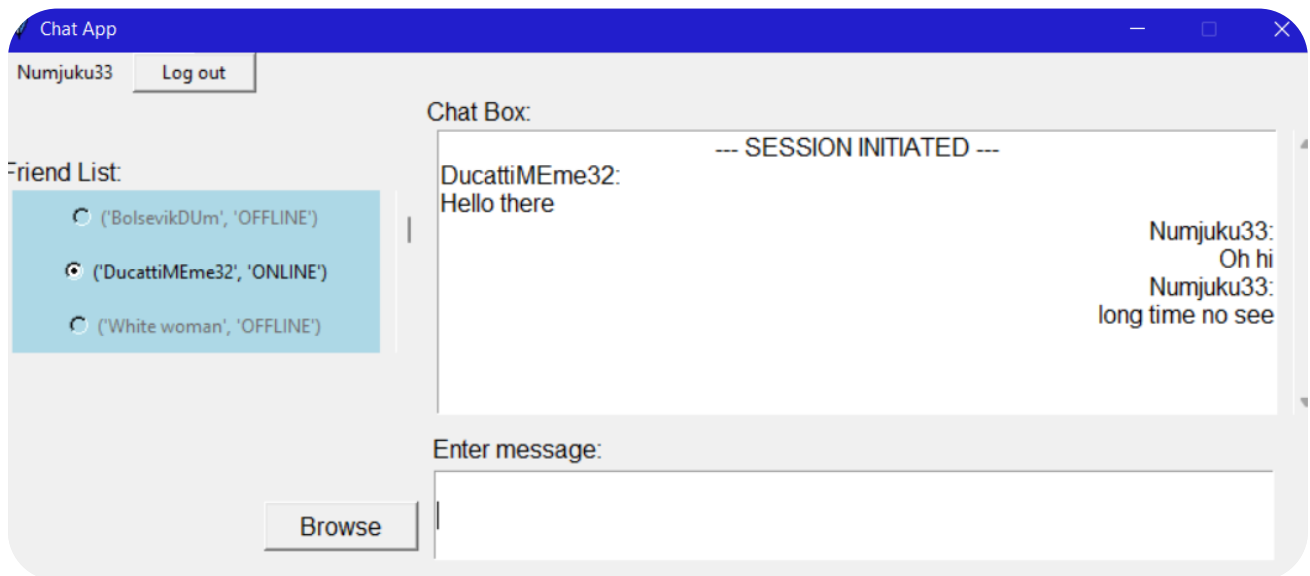
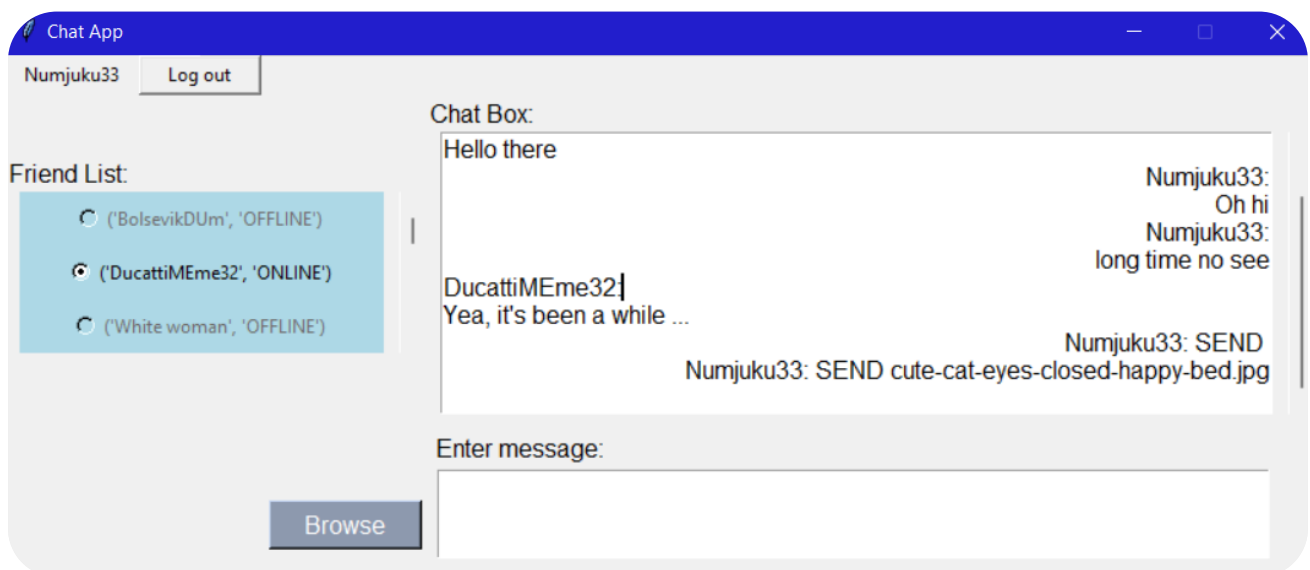


Figure 16. Banana connected to Apple.

Logging out will terminate the chat session but will not delete the chat history (unless one of the users disconnected to the server). Login again with another account will not provide the previous chat history.

There is also the File Transferring feature to transfer a file from one Peer to another just by clicking on the Browse Button and choose any file you want to send.



END OF REPORT!!!