

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Mạng máy tính - CO3094

Báo cáo

DEVELOP A SIMPLE FILE-SHARING APPLICATION

Giảng viên hướng dẫn: Nguyễn Phương Duy

Sinh viên thực hiện: 2110527 - Nguyễn Hoàng Duy Tân
2112594 - Trần Nguyễn Minh Tuệ
2110342 - Nguyễn Minh Lộc

Mục lục

1. Thành viên và phân chia công việc	6
2. Phân tích yêu cầu	7
2.1. Functional requirements	7
2.1.1. Client Functions	7
2.1.2. Server Functions	8
2.2. Non-functional requirements	8
2.3. Phân tích kiến trúc	8
2.3.1. Kiến trúc Peer-to-Peer (P2P)	8
2.3.2. Kiến trúc client-server	8
3. Giới thiệu Protocol	10
3.1. HyperText Transfer Protocol	10
3.2. Transmission Control Protocol	11
4. Socket programming with Python	11
5. Application design	13
5.1. Architecture design	13
5.2. Flow protocol design	14
5.2.1. Register	14
5.2.2. Login	14
5.2.3. Ping	14
5.2.4. Discover	14
5.2.5. Publish	15
5.2.6. Fetch	15
5.3. Activity diagram	16
5.3.1. Register	16
5.3.2. Login	16
5.3.3. Ping	17
5.3.4. Discover	18
5.3.5. Publish	19
5.3.6. Fetch	19
5.4. Status Code	20
6. Design UI	21
6.1. Register and Login	21
6.2. Publish Page	22
6.3. Fetch Page	23

7. Tutorial - Guides	25
7.1. Server	25
7.1.1. ping	26
7.1.2. Discover	28
7.2. Client	30
7.2.1. Register	30
7.2.2. Login	32
7.2.3. Publish	33
7.2.4. Fetch	34
8. Error Handling	37
8.1. Server not running	37
8.2. Connection error	37
8.3. Login & Register: Wrong input format	39
8.4. Register: User already exist	39
8.5. Login: User not found	40
8.6. Login: Wrong password	41
8.7. Login: User already online	41
8.8. Publish: File not found	43
8.9. Publish: File already published	43
8.10. Fetch: File not found	45
8.11. Fetch: File already fetched	45

Danh mục hình vẽ

Hình 1: 8 giao thức mạng máy tính phổ biến (Nguồn Bytebytego)	10
Hình 2: Mô hình socket	11
Hình 3: Quy trình một kết nối socket để gửi nhận dữ liệu (Nguồn ResearchGate)	12
Hình 4: Kiến trúc tổng quan của hệ thống	13
Hình 5: Sơ đồ luồng của chức năng đăng ký	14
Hình 6: Sơ đồ luồng của chức năng đăng nhập	14
Hình 7: Sơ đồ luồng của chức năng ping	14
Hình 8: Sơ đồ luồng của chức năng discover	14
Hình 9: Sơ đồ luồng của chức năng publish	15
Hình 10: Sơ đồ luồng của chức năng fetch	15
Hình 11: Sơ đồ hoạt động của chức năng đăng ký	16
Hình 12: Sơ đồ hoạt động của chức năng đăng nhập	16
Hình 13: Sơ đồ hoạt động của chức năng ping	17

Hình 14: Sơ đồ hoạt động của chức năng discover	18
Hình 15: Sơ đồ hoạt động của chức năng publish	19
Hình 16: Sơ đồ hoạt động của chức năng fetch	19
Hình 17: Trao đổi status code giữa client và server	20
Hình 18: UI Register and Login	21
Hình 19: UI Publish File	22
Hình 20: UI Fetch File	23
Hình 21: UI Profile	24
Hình 22: Start Server	25
Hình 23: Start Server	26
Hình 24: Ping	27
Hình 25: Ping No Online	28
Hình 26: Discover	29
Hình 27: Start Client	30
Hình 28: Client Register	30
Hình 29: Client Register Success	31
Hình 30: Server Register Success	32
Hình 31: Client Login	33
Hình 32: Login Success	33
Hình 33: Publish	34
Hình 34: Publish Success	34
Hình 35: Fetch	35
Hình 36: Fetch Success	35
Hình 37: Fetch Success	36
Hình 38: Server not running	37
Hình 39: Connection error	38
Hình 40: Wrong input format	39
Hình 41: User already exist	40
Hình 42: User not found	40
Hình 43: Wrong password	41
Hình 44: User already online	42
Hình 45: File not found	43
Hình 46: File already published	44
Hình 47: File not found	45
Hình 48: File already fetched	46



Danh mục bảng biểu

Bảng 1: Các status code được định nghĩa	20
---	----



1. Thành viên và phân chia công việc

Họ và tên	Nhiệm vụ	Mức độ hoàn thành
Nguyễn Hoàng Duy Tân		100%
Trần Nguyễn Minh Tuệ		100%
Nguyễn Minh Lộc		100%

2. Phân tích yêu cầu

2.1. Functional requirements

Xây dựng một ứng dụng chia sẻ file đơn giản với giao thức được định nghĩa sẵn, sử dụng những giao thức trong TCP/IP stack.

2.1.1. Client Functions

Basic functions

Đăng ký trong kho lưu trữ

- Máy khách có thể gửi yêu cầu đăng ký file có trong kho lưu trữ cho máy chủ.
- Thông điệp đăng ký file: “publish “. File trên máy khách sẽ được thêm vào kho lưu trữ dưới tên .
- Các file sau khi đăng ký sẽ được lưu trữ trong kho lưu trữ của tài khoản được liên kết với máy chủ.

Gửi yêu cầu tải file cho server

- Máy khách có thể gửi yêu cầu tải file không có sẵn trong kho lưu trữ của mình. Lúc này máy chủ sẽ phản hồi lại danh sách các máy khách khác có file được yêu cầu.

Tải file trực tiếp từ nguồn muốn chọn

- Máy khách sau khi nhận được phản hồi từ máy chủ danh sách máy khách có sẵn file được yêu cầu có thể chọn một nguồn thích hợp và gửi yêu cầu tải file tới đó.
- Các máy khách được cung cấp một danh sách yêu cầu tải file từ các máy khách khác, máy khách có thể chọn 1 file trong danh sách và gửi yêu cầu tải file tới máy khách đó.
- Thông điệp tải file: “fetch “. Trong đó fname là 1 trong những tên file muốn chọn sau khi server đã phản hồi.

Extended functions

Đăng ký tài khoản

- Người dùng đăng ký địa chỉ của máy vào hệ thống của máy chủ.

Đăng nhập tài khoản và xác thực

- Người dùng đăng nhập tài khoản để sử dụng các chức năng của hệ thống.

Liệt kê danh sách lưu trữ

- Máy khách có thể kiểm tra danh sách các file có trong kho lưu trữ của mình.

Tìm kiếm bằng từ khóa

- Server sẽ hỗ trợ người dùng tìm kiếm file theo từ khóa.

2.1.2. Server Functions

Basic functions

Kiểm tra trạng thái máy chủ

- Máy chủ có thể kiểm tra trạng thái của máy chủ khác thông qua lệnh “ping <hostname>”

Xem danh sách file của máy khách khác

- Máy chủ có thể xem danh sách file trong kho lưu trữ của các máy khách thông qua lệnh “discover <hostname>”

Gửi thông tin cần thiết sau khi nhận yêu cầu tải file từ clients

- Sau khi nhận được yêu cầu tìm file từ người dùng, server sẽ tiến hành theo dõi và tìm kiếm để trả về các thông tin nơi đang lưu trữ các file đó cho clients: ID peer, thời gian file được cập nhật.

Extended functions

Xem file log

- Máy chủ có thể xem file log của máy khách khác thông qua.

2.2. Non-functional requirements

Giao diện người dùng – Cung cấp giao diện người dùng để sử dụng cho máy khách để nhập các lệnh và theo dõi quá trình tải tệp. **Multi-threading** – Triển khai đa luồng trong máy khách để có thể xử lý nhiều tải xuống cùng lúc. **Hiệu năng và tích hợp** – Đảm bảo rằng hệ thống hoạt động hiệu quả và có khả năng tích hợp với các mạng internet và hệ thống người dùng khác nhau.

2.3. Phân tích kiến trúc

2.3.1. Kiến trúc Peer-to-Peer (P2P)

- Kiến trúc Peer-to-Peer (P2P) là một mô hình mạng máy tính trong đó các máy tính (được gọi là nút hoặc “peers”) kết nối trực tiếp với nhau để chia sẻ tài nguyên và thông tin mà không cần sự tương tác trung tâm từ máy chủ. Trong kiến trúc P2P, mỗi máy tính có thể đồng thời hoạt động như máy khách và máy chủ, có nghĩa là chúng có khả năng yêu cầu tài nguyên từ các máy tính khác và chia sẻ tài nguyên với người khác.

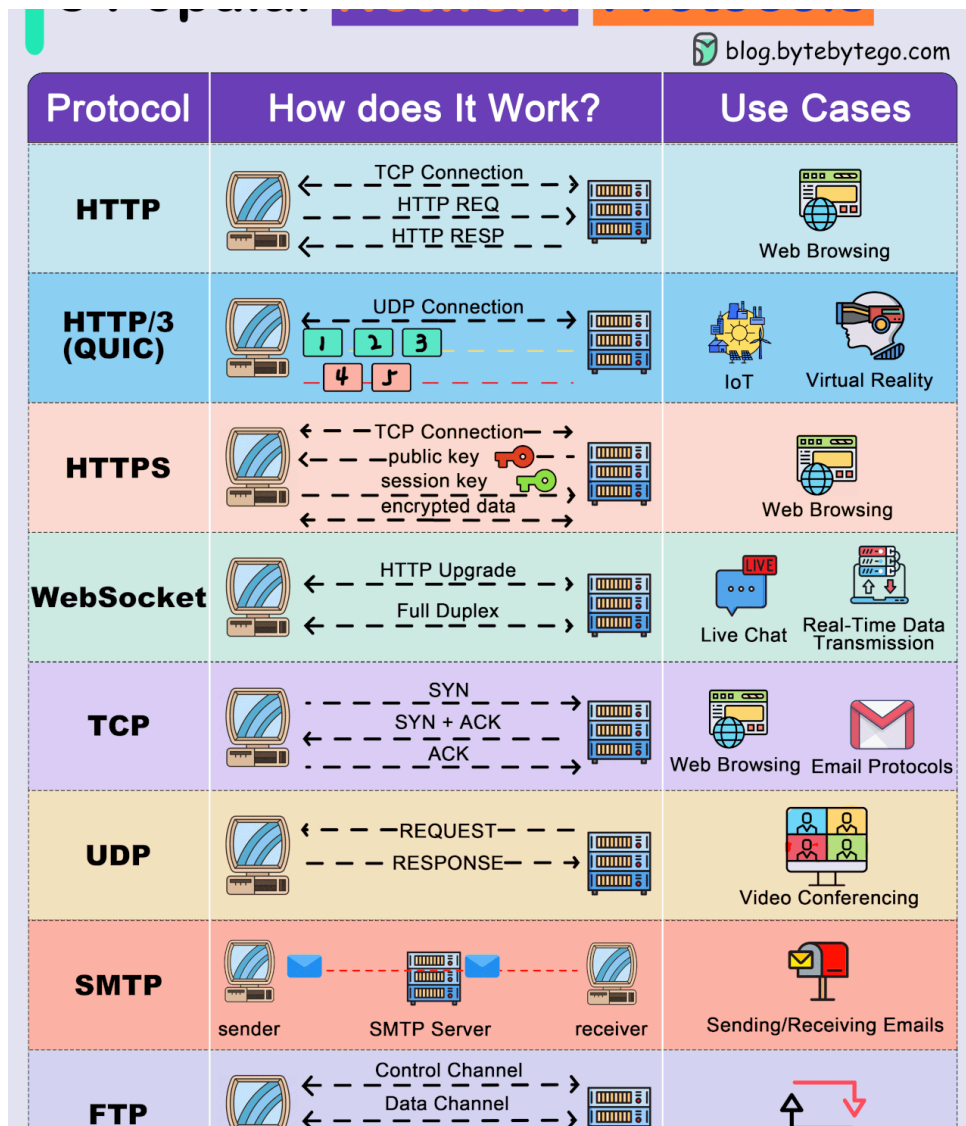
2.3.2. Kiến trúc client-server

- Kiến trúc client-server (còn được gọi là mô hình client-server) là một kiến trúc máy tính phổ biến được sử dụng trong việc tổ chức và quản lý các dịch vụ và tài nguyên trên mạng. Nó dựa trên sự phân chia các vai trò chính trong hệ thống thành hai phần: máy khách (client) và máy chủ (server). Hai phần này tương tác với nhau để cung cấp các dịch vụ, ứng dụng, và tài nguyên cho người dùng.

3. Giới thiệu Protocol

3.1. HyperText Trànner Protocol

HyperText Transfer Protocol (HTTP) là một giao thức ở tầng ứng dụng trong mô hình OSI để gửi và nhận tài liệu, hình ảnh, văn bản như HTML document. Về cơ bản, giao thức HTTP xây dựng trên cơ chế request-response trong mô hình client-server. Trong mô hình này, client có thể là một process thuộc máy tính này, server có thể là process thuộc máy tính khác, hai process thuộc hai phần cứng khác nhau khi muốn giao tiếp với nhau thì sẽ thông qua HTTP để giao tiếp. Ai là người request thì đó là client, người nhận request để response sẽ là server.



Hình 1: 8 giao thức mạng máy tính phổ biến (Nguồn Bytebytego)

3.2. Transmission Control Protocol

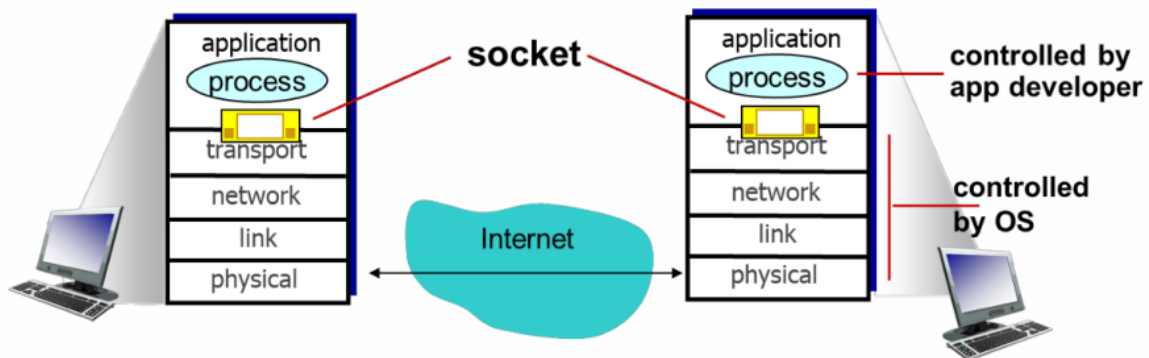
Transmission Control Protocol (TCP) là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Nhờ có TCP, các ứng dụng trên các host được nối mạng có thể tạo các kết nối với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự.

Cách đặc tính cơ bản của TCP:

- Point-to-point: Trong một giao thức TCP, chỉ có một sender và một server được kết nối với nhau bằng 3-way handshaking.
- Reliable, in-order bit stream: Hỗ trợ truyền tin cậy và đúng thứ tự.
- Pipelined: Truyền song song nhằm tăng hiệu quả gửi nhận
- Flow control: Receiver kiểm soát tốc độ gửi của sender để tránh làm quá tải receiver.
- Congestion control: Tự động điều chỉnh tốc độ gửi ở mức tối đa mà không làm tắc nghẽn hệ thống.
- Full-duplex connection: hỗ trợ truyền hai chiều trong cùng một thời điểm trong một kết nối.

4. Socket programming with Python

Socket là cánh cổng ngăn cách giữa Application Layer với Transport Layer.



Hình 2: Mô hình socket

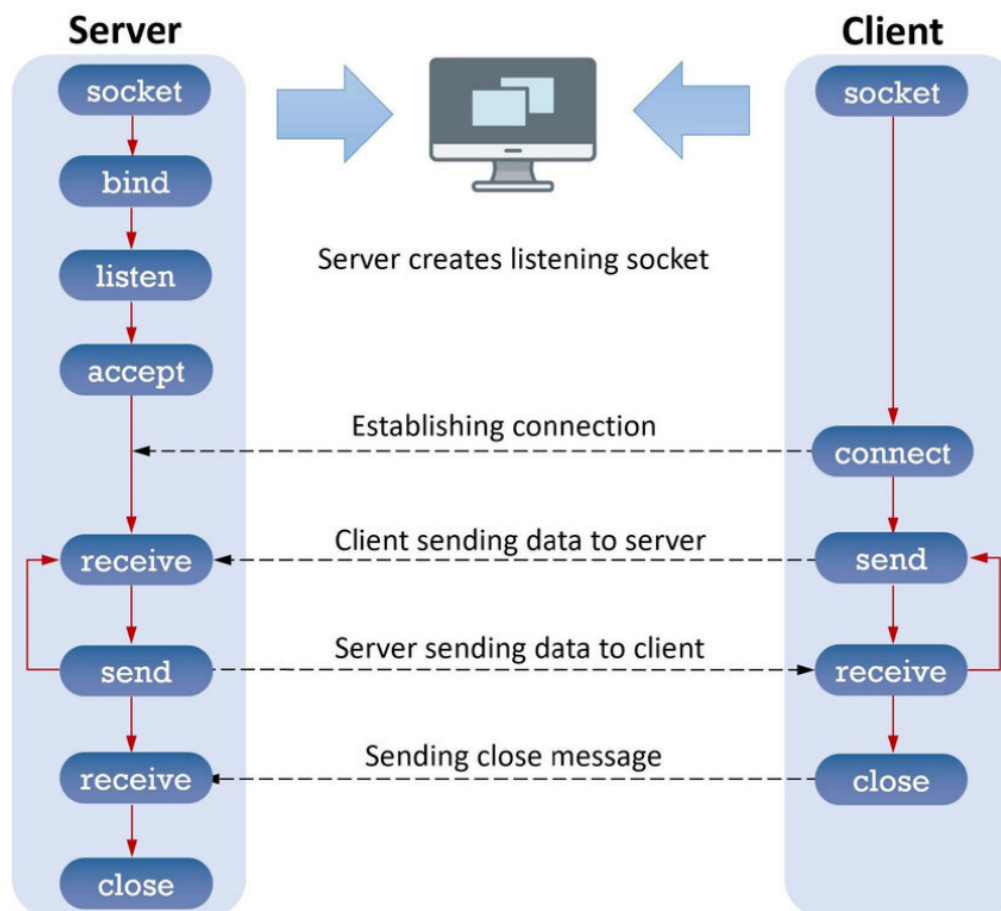
Thư viện socket của Python hỗ trợ một số API sau để thiết lập kết nối socket TCP/UDP:

- socket()
- bind()
- listen()
- accept()
- connect()

- connect_ex()
- send()
- recv()
- close()

Thư viện socket của Python hỗ trợ cả TCP socket lẫn UDP socket. Trong project này, nhóm dự định sử dụng TCP socket để hiện thực dự án.

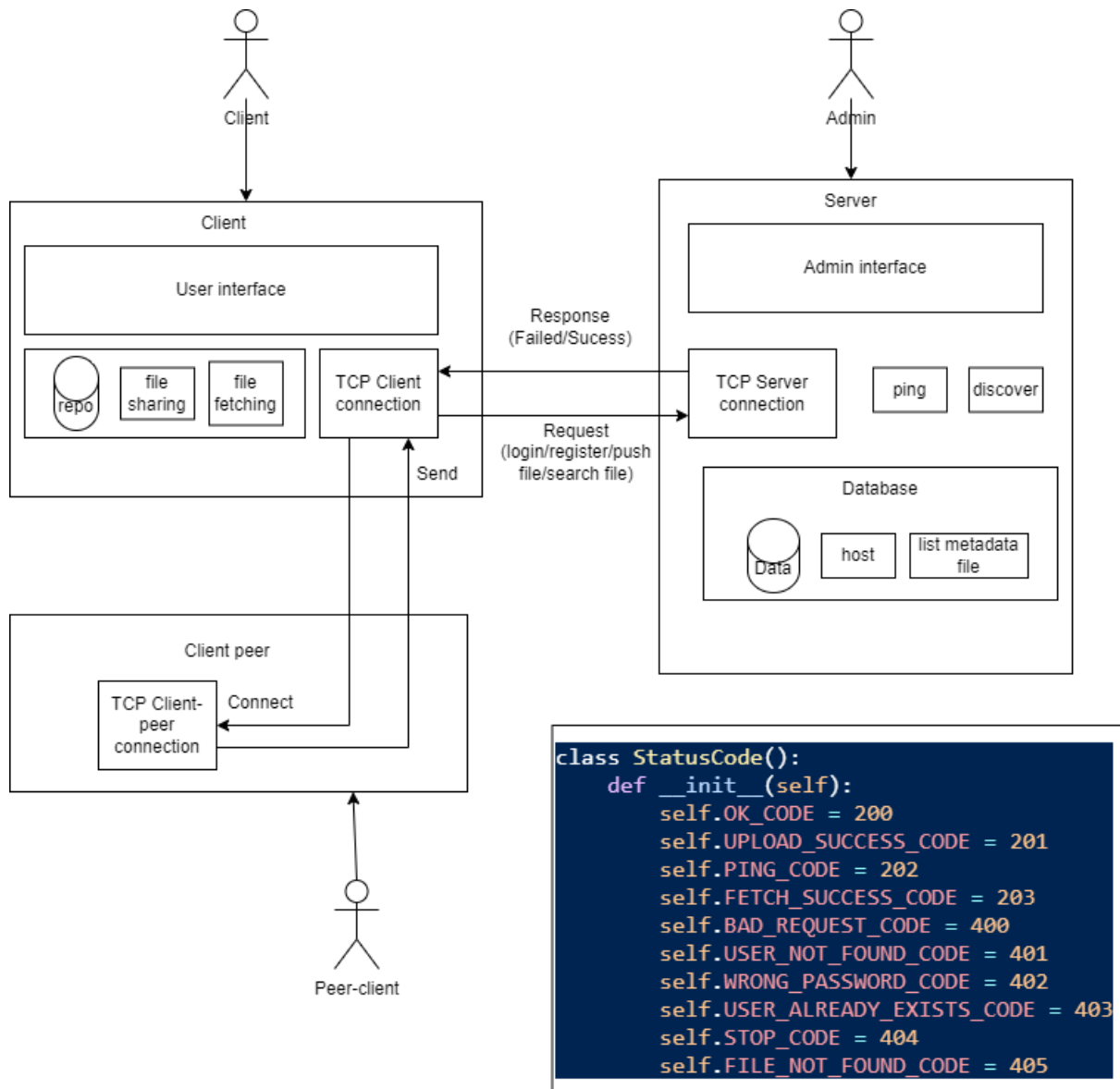
Flow của một TCP socket connection có thể được thể hiện như hình dưới đây:



Hình 3: Quy trình một kết nối socket để gửi nhận dữ liệu (Nguồn ResearchGate)

5. Application design

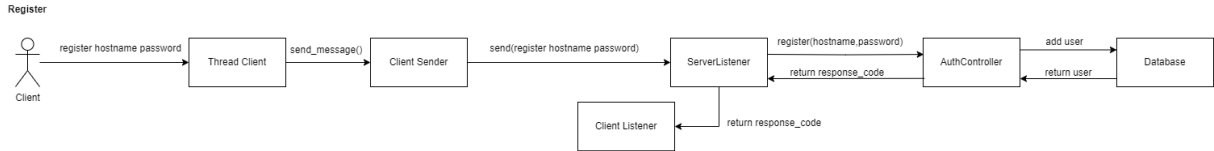
5.1. Architecture design



Hình 4: Kiến trúc tổng quan của hệ thống

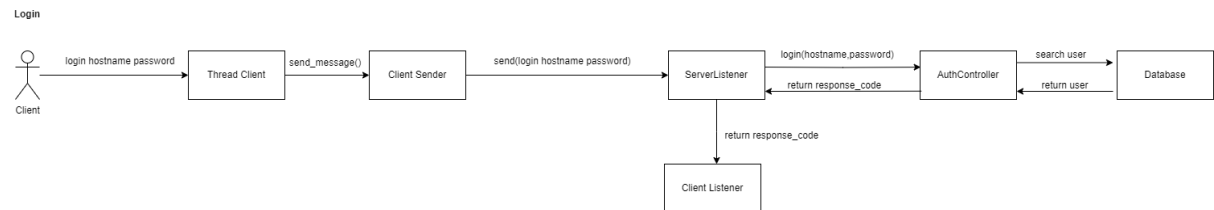
5.2. Flow protocol design

5.2.1. Register



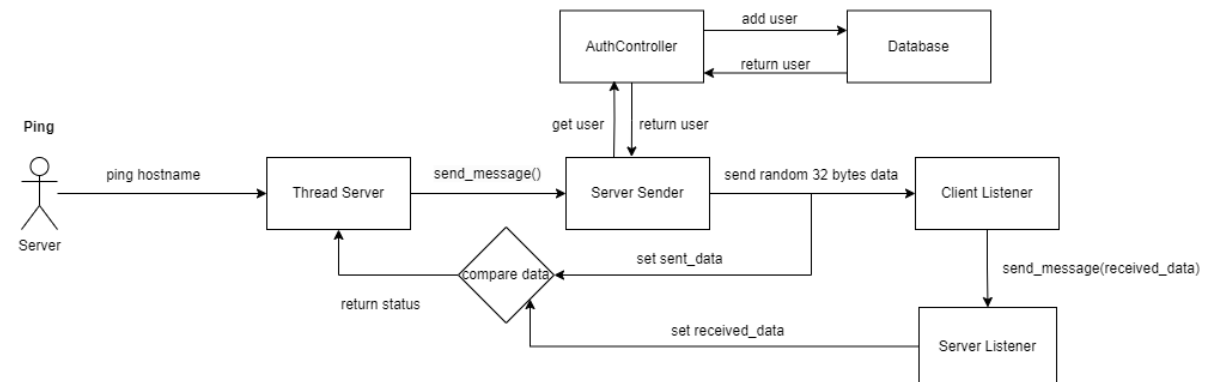
Hình 5: Sơ đồ luồng của chức năng đăng ký

5.2.2. Login



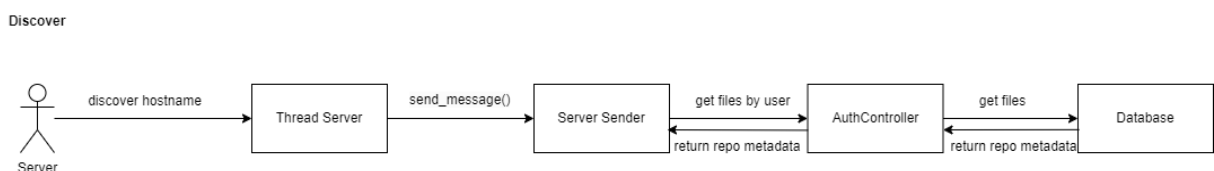
Hình 6: Sơ đồ luồng của chức năng đăng nhập

5.2.3. Ping



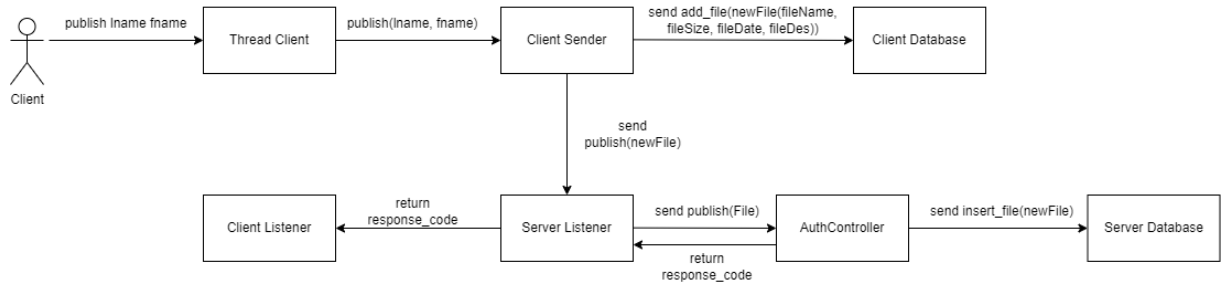
Hình 7: Sơ đồ luồng của chức năng ping

5.2.4. Discover



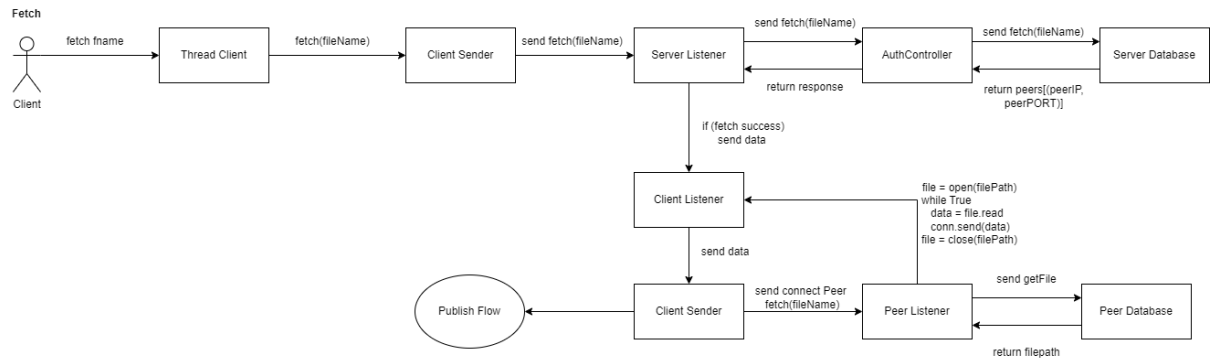
Hình 8: Sơ đồ luồng của chức năng discover

5.2.5. Publish



Hình 9: Sơ đồ luồng của chức năng publish

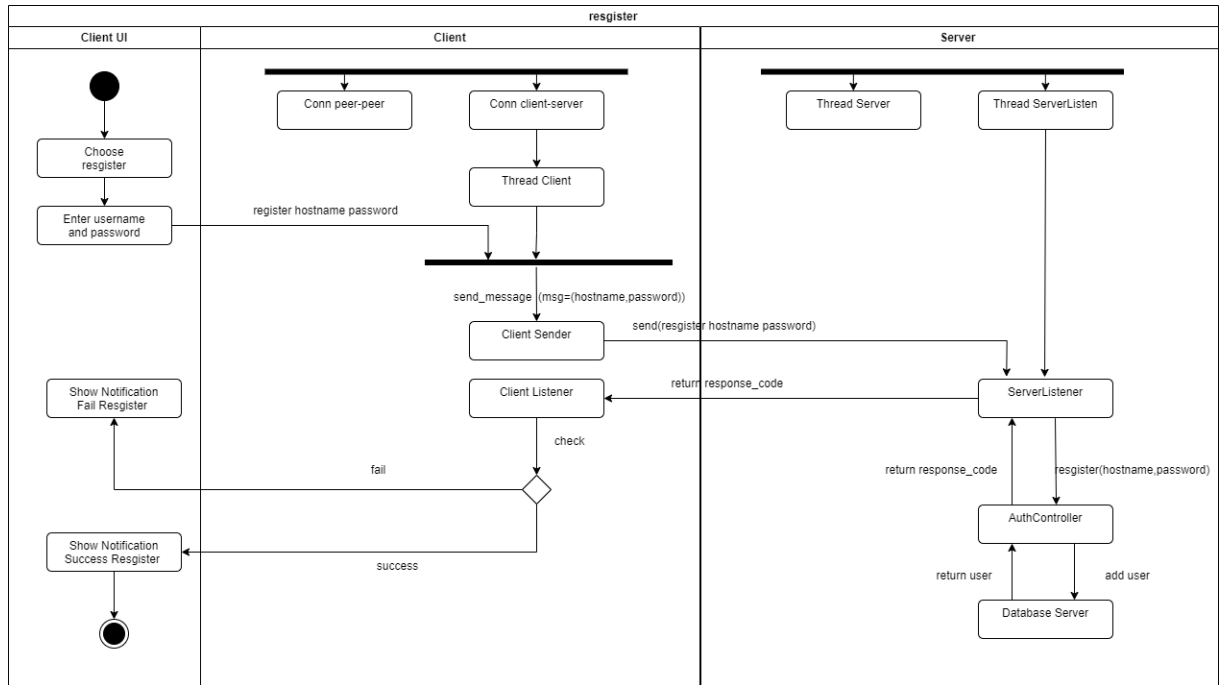
5.2.6. Fetch



Hình 10: Sơ đồ luồng của chức năng fetch

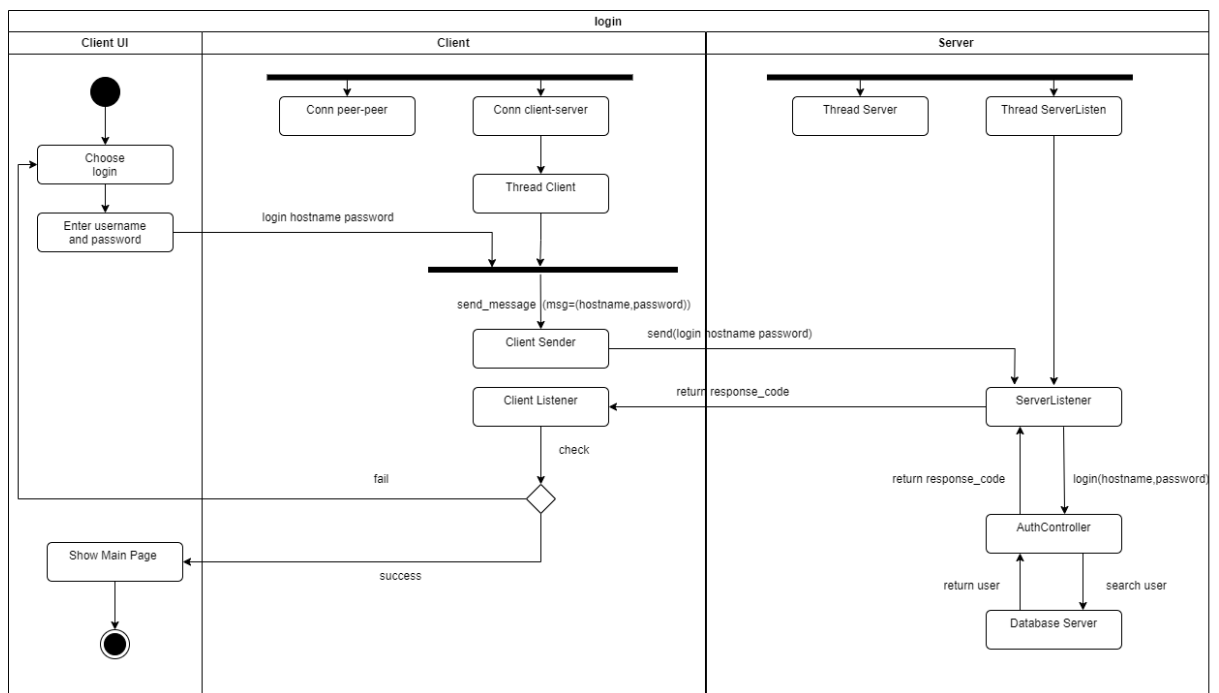
5.3. Activity diagram

5.3.1. Register



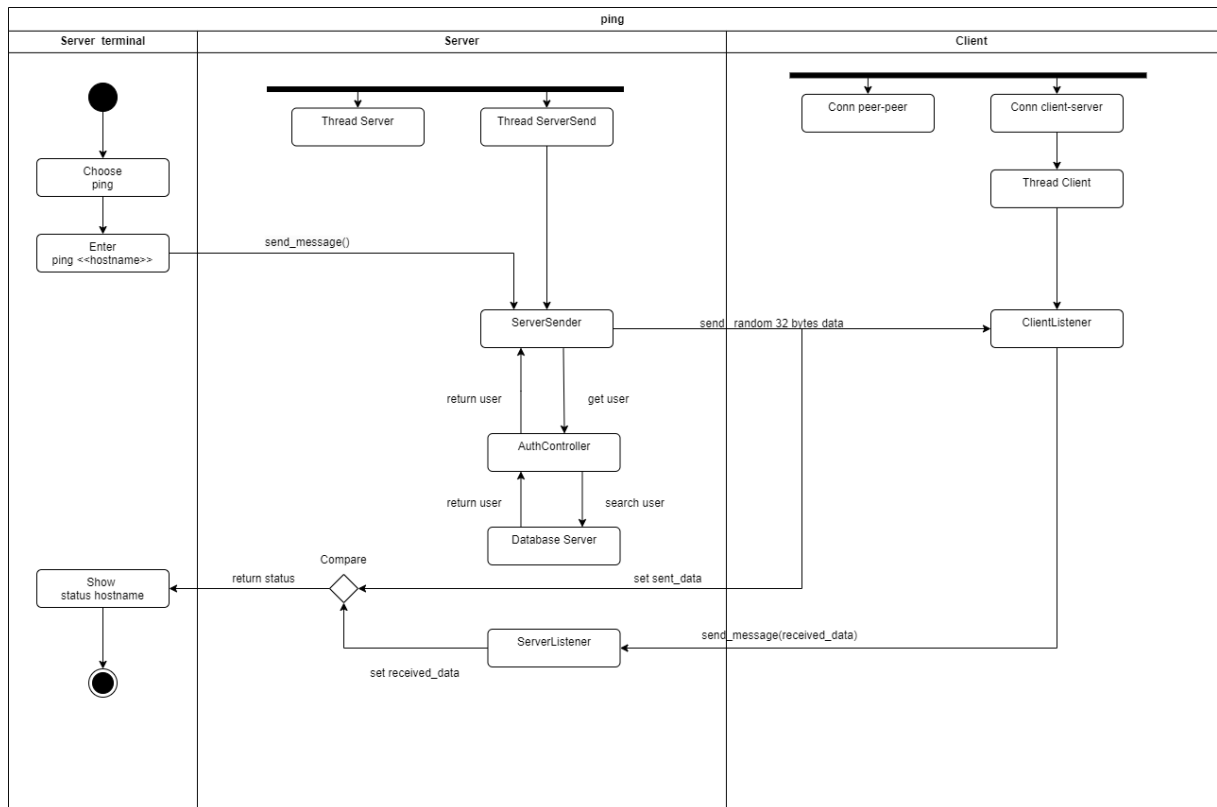
Hình 11: Sơ đồ hoạt động của chức năng đăng ký

5.3.2. Login



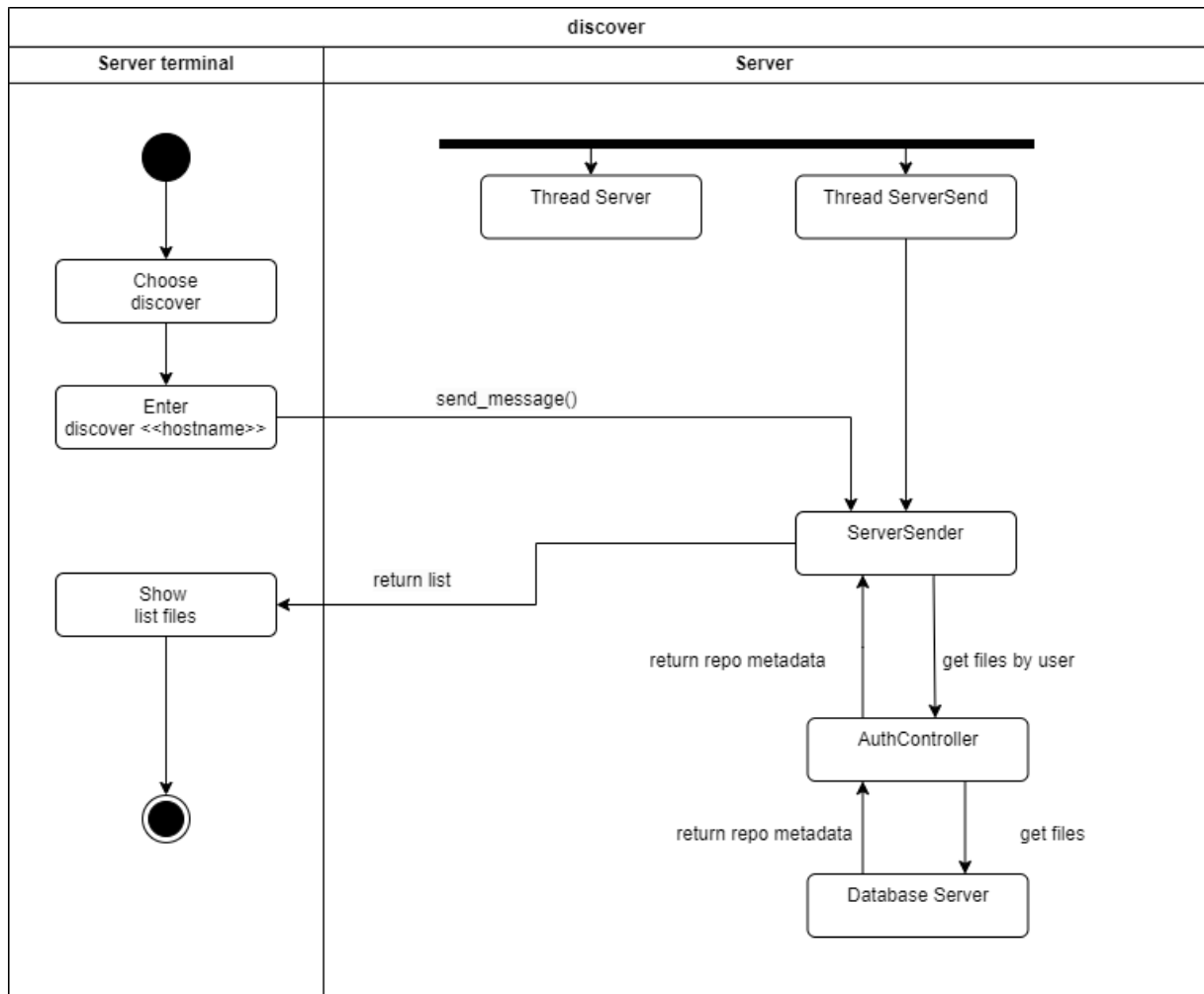
Hình 12: Sơ đồ hoạt động của chức năng đăng nhập

5.3.3. Ping



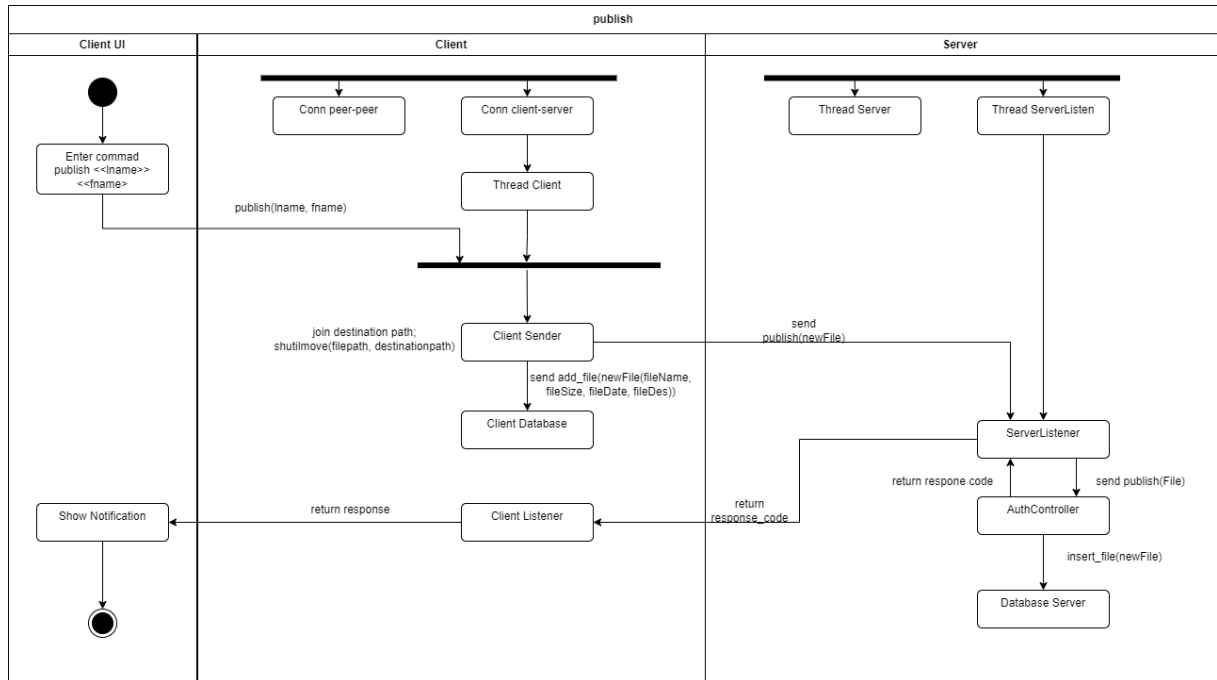
Hình 13: Sơ đồ hoạt động của chức năng ping

5.3.4. Discover



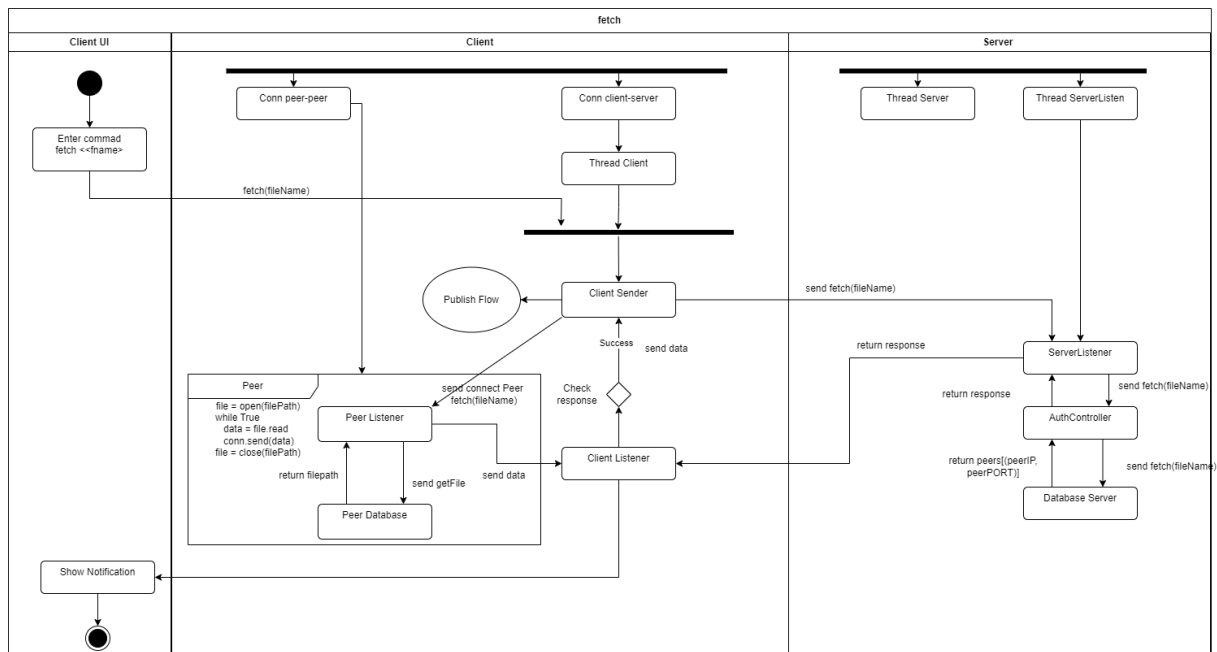
Hình 14: Sơ đồ hoạt động của chức năng discover

5.3.5. Publish



Hình 15: Sơ đồ hoạt động của chức năng publish

5.3.6. Fetch



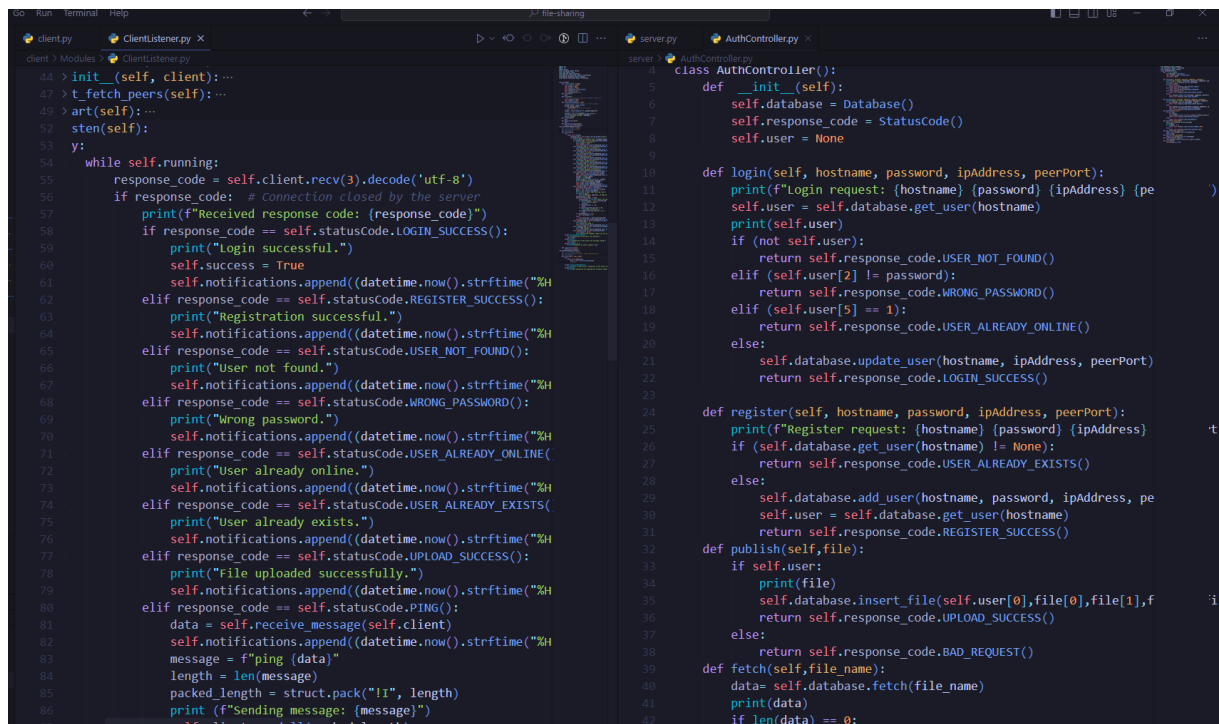
Hình 16: Sơ đồ hoạt động của chức năng fetch

5.4. Status Code

Ngoài ra, nhóm còn định nghĩa thêm một số status code để phục vụ cho việc trao đổi thông tin giữa server đến client và ngược lại.

Status Code	Mô tả
200	Đăng nhập thành công
201	Đăng ký thành công
202	Publish thành công
203	Server muốn ping client
204	Fetch thành công
205	Download file thành công
400	Yêu cầu không hợp lệ
401	Không tìm thấy tài khoản
402	Sai mật khẩu
403	Tài khoản đã đăng nhập
404	Tài khoản đã tồn tại
405	Client thoát khỏi server
406	File không tồn tại

Bảng 1: Các status code được định nghĩa



```

client.py
class ClientListener:
    def __init__(self, client):
        self.client = client
    def t_fetch_peers(self):
        # ...
    def art(self):
        # ...
    def sten(self):
        # ...
    def while self.running:
        response_code = self.client.recv(3).decode('utf-8')
        if response_code == self.statusCode.CONNECTION_CLOSED():
            print(f"Received response code: {response_code}")
            if response_code == self.statusCode.LOGIN_SUCCESS():
                print("Login successful.")
                self.success = True
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "Login successful."))
            elif response_code == self.statusCode.REGISTER_SUCCESS():
                print("Registration successful.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "Registration successful."))
            elif response_code == self.statusCode.USER_NOT_FOUND():
                print("User not found.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "User not found."))
            elif response_code == self.statusCode.WRONG_PASSWORD():
                print("Wrong password.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "Wrong password."))
            elif response_code == self.statusCode.USER_ALREADY_ONLINE():
                print("User already online.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "User already online."))
            elif response_code == self.statusCode.USER_ALREADY_EXISTS():
                print("User already exists.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "User already exists."))
            elif response_code == self.statusCode.UPLOAD_SUCCESS():
                print("File uploaded successfully.")
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), "File uploaded successfully."))
            elif response_code == self.statusCode.PING():
                data = self.receive_message(self.client)
                self.notifications.append((datetime.now().strftime("%H:%M:%S"), f"ping {data}"))
                length = len(message)
                packed_length = struct.pack("i", length)
                print(f"Sending message: {message}")
                self.client.send(packed_length + message)

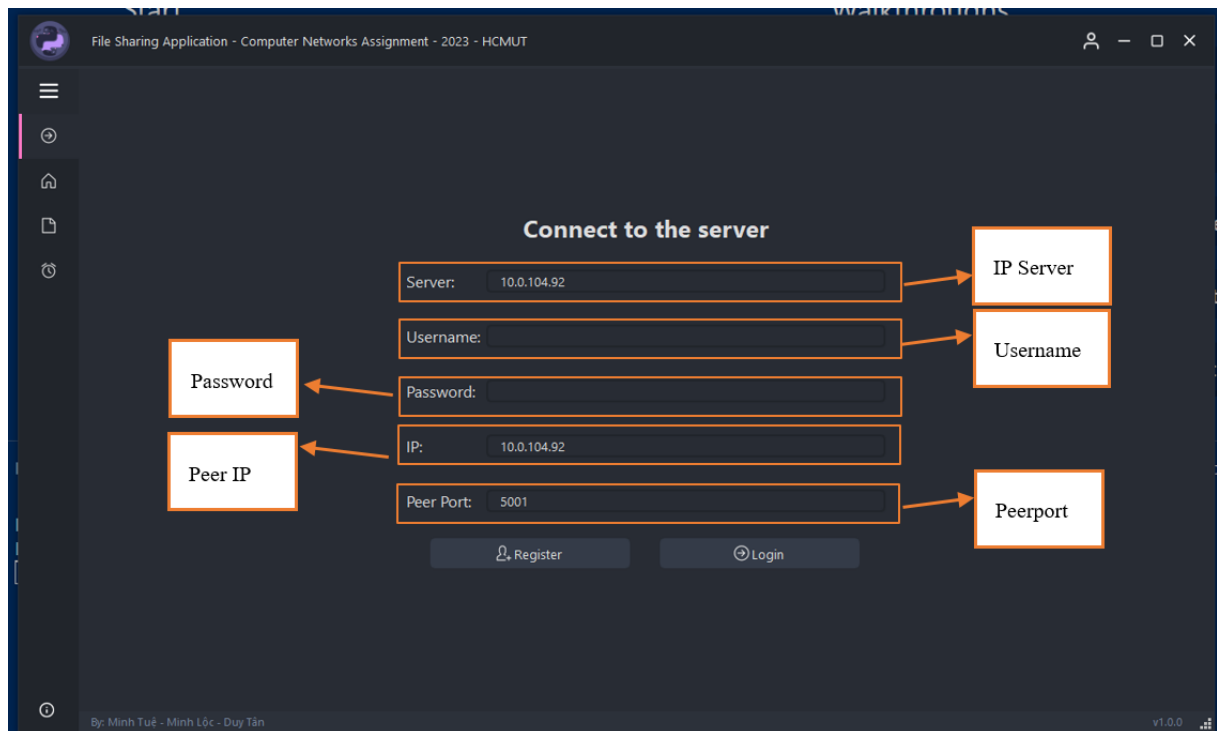
AuthController.py
class AuthController:
    def __init__(self):
        self.database = Database()
        self.response_code = StatusCode()
        self.user = None
    def login(self, hostname, password, ipAddress, peerPort):
        print(f"Login request: {hostname} {password} {ipAddress} {peerPort}")
        self.user = self.database.get_user(hostname)
        print(self.user)
        if not self.user:
            return self.response_code.USER_NOT_FOUND()
        elif self.user[2] != password:
            return self.response_code.WRONG_PASSWORD()
        elif self.user[5] == 1:
            return self.response_code.USER_ALREADY_ONLINE()
        else:
            self.database.update_user(hostname, ipAddress, peerPort)
            return self.response_code.LOGIN_SUCCESS()
    def register(self, hostname, password, ipAddress, peerPort):
        print(f"Register request: {hostname} {password} {ipAddress} {peerPort}")
        if self.database.get_user(hostname) != None:
            return self.response_code.USER_ALREADY_EXISTS()
        else:
            self.database.add_user(hostname, password, ipAddress, peerPort)
            self.user = self.database.get_user(hostname)
            return self.response_code.REGISTER_SUCCESS()
    def publish(self, file):
        if self.user:
            print(file)
            self.database.insert_file(self.user[0], file[0], file[1], file[2])
            return self.response_code.UPLOAD_SUCCESS()
        else:
            return self.response_code.BAD_REQUEST()
    def fetch(self, file_name):
        data = self.database.fetch(file_name)
        print(data)
        if len(data) == 0:
            return self.response_code.FILE_NOT_FOUND()

```

Hình 17: Trao đổi status code giữa client và server

6. Design UI

6.1. Register and Login



The screenshot shows a web application window titled "File Sharing Application - Computer Networks Assignment - 2023 - HCMUT". The main content area is titled "Connect to the server". It contains five input fields with labels: "Server:" (value: 10.0.104.92), "Username:", "Password:", "IP:" (value: 10.0.104.92), and "Peer Port:" (value: 5001). Below these fields are two buttons: "Register" and "Login". Arrows point from labels to their respective input fields: "IP Server" points to the Server field, "Username" points to the Username field, "Password" points to the Password field, "Peer IP" points to the IP field, and "Peerport" points to the Peer Port field. The footer of the application shows the text "Bç Minh Tuệ - Minh Lộc - Duy Tân" and the version "v1.0.0".

Hình 18: UI Register and Login

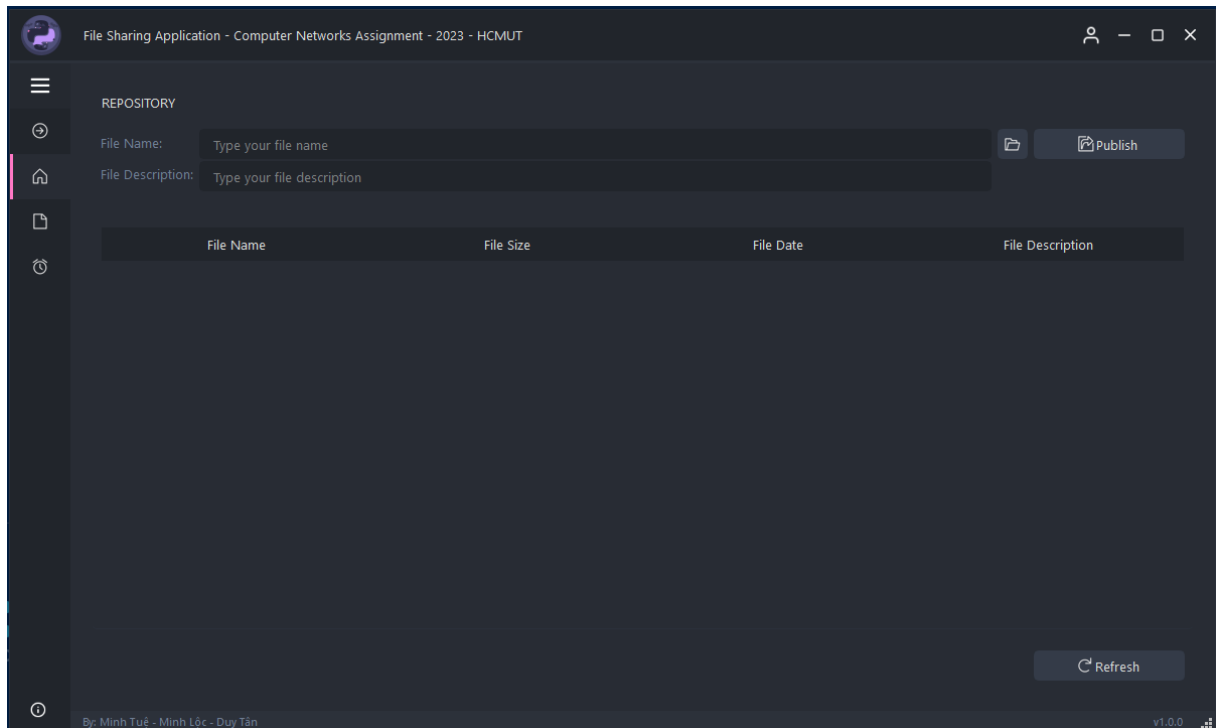
Đây là trang đăng ký, đăng nhập vào hệ thống bao gồm: địa chỉ IP Server, username, password, địa chỉ IP Peer, địa chỉ Peer Port.

Phía dưới có 2 nút: Register và Login.

Khi nhấn nút Register, hệ thống sẽ gửi thông tin đăng ký lên Server.

Khi nhấn nút Login, hệ thống sẽ gửi thông tin đăng nhập lên Server.

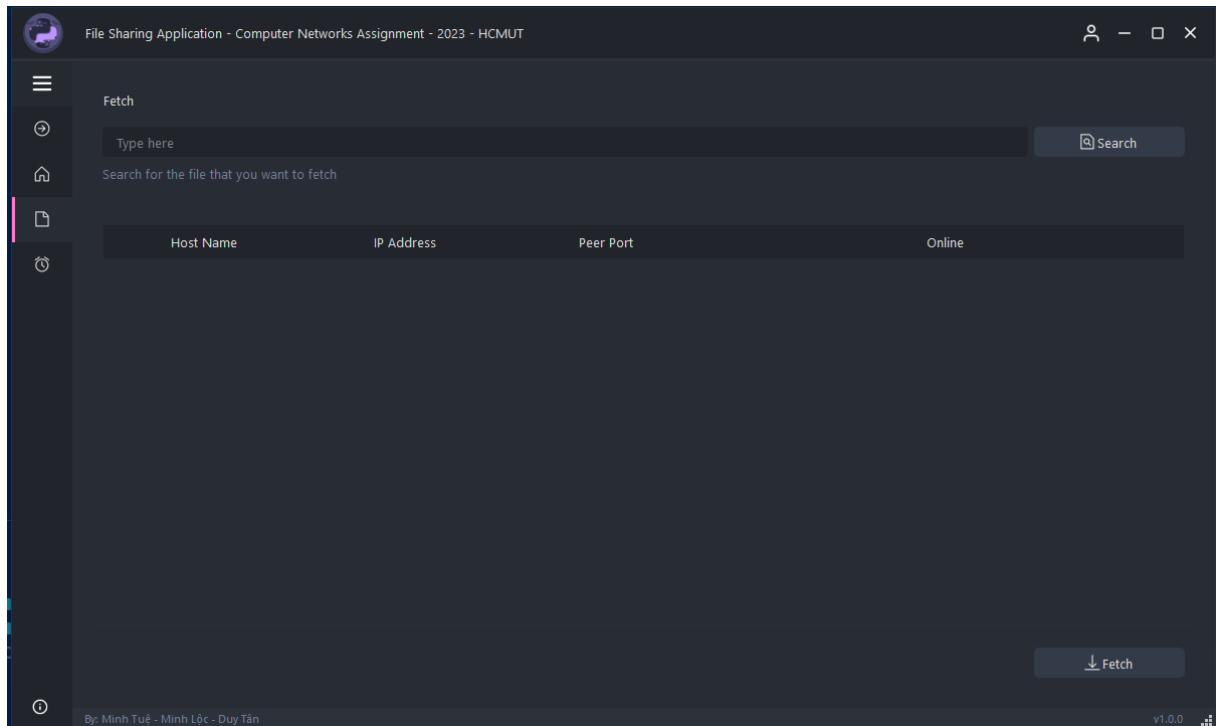
6.2. Publish Page



Hình 19: UI Publish File

Đây là trang Publish File, bao gồm: đường dẫn file, tên file fname sau khi publish, mô tả file, nút Publish. Khi nhấn nút Publish, hệ thống sẽ gửi thông tin Publish lên Server. Phía dưới là lưới hiển thị danh sách các file đã Publish.

6.3. Fetch Page



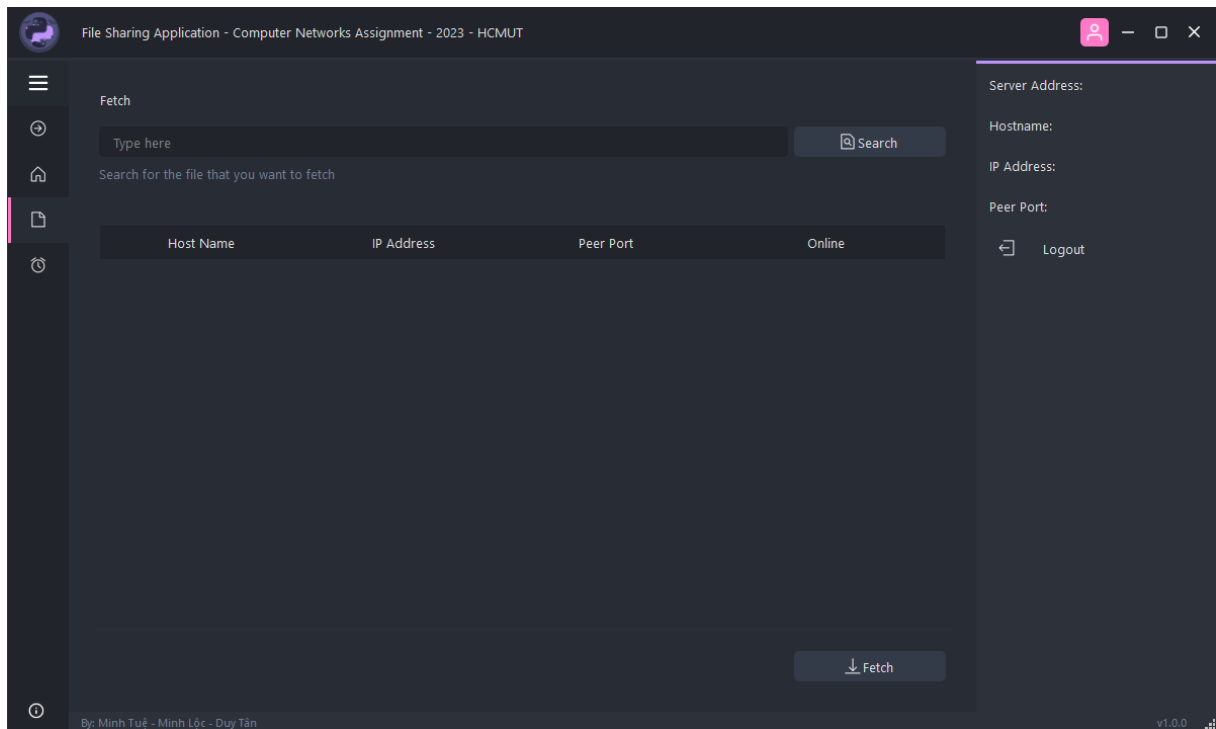
Hình 20: UI Fetch File

Đây là trang Fetch File, bao gồm: tên file fname cần Fetch, nút Search.

Sau khi Search, giao diện sẽ hiển thị danh sách các file có tên fname.

Người dùng có thể chọn 1 file trong danh sách và nhấn nút Fetch để Fetch file về.

Ngoài ra còn hiển thị thông tin người dùng: username, địa chỉ IP Peer, địa chỉ Peer Port.

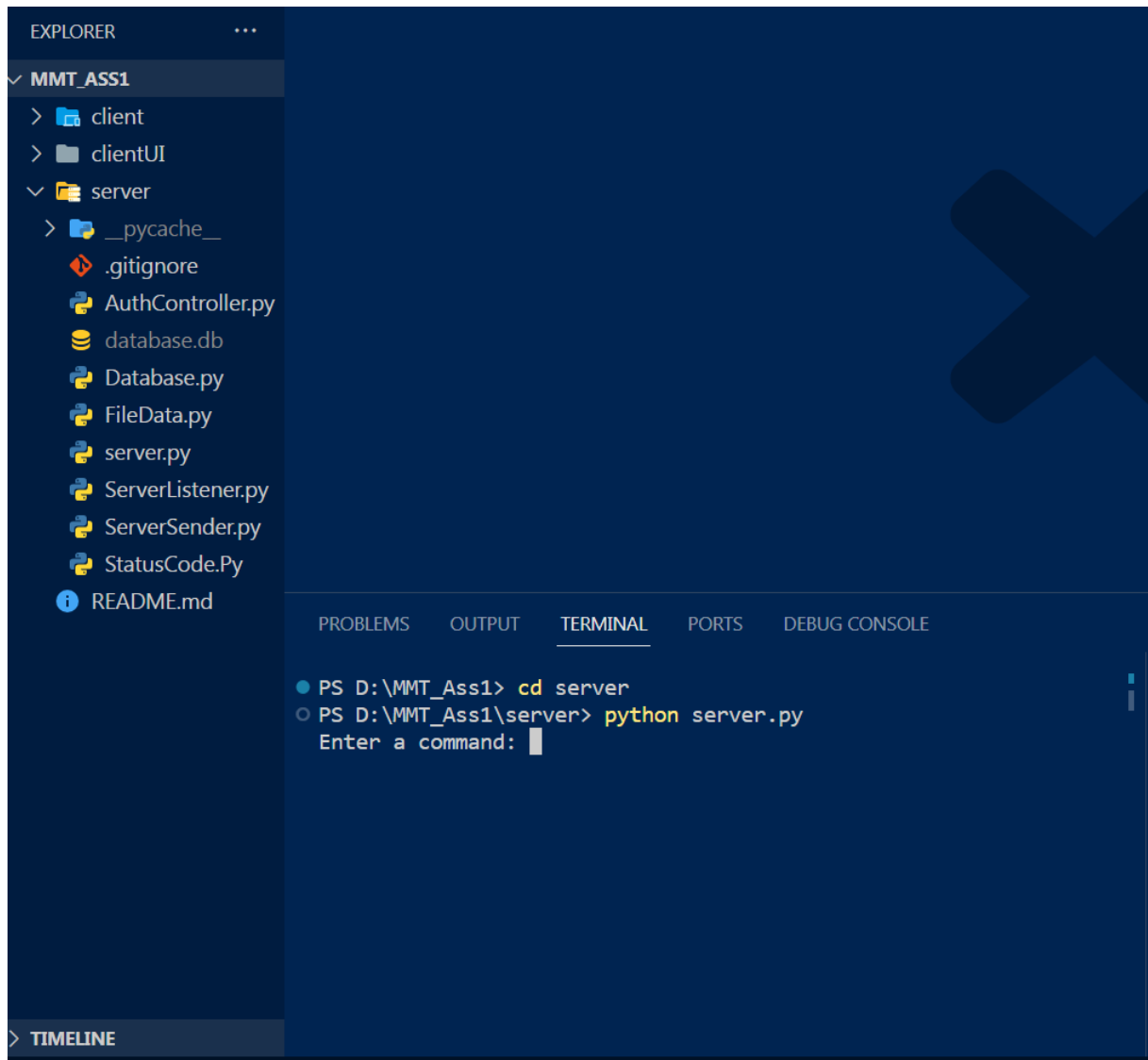


Hình 21: UI Profile

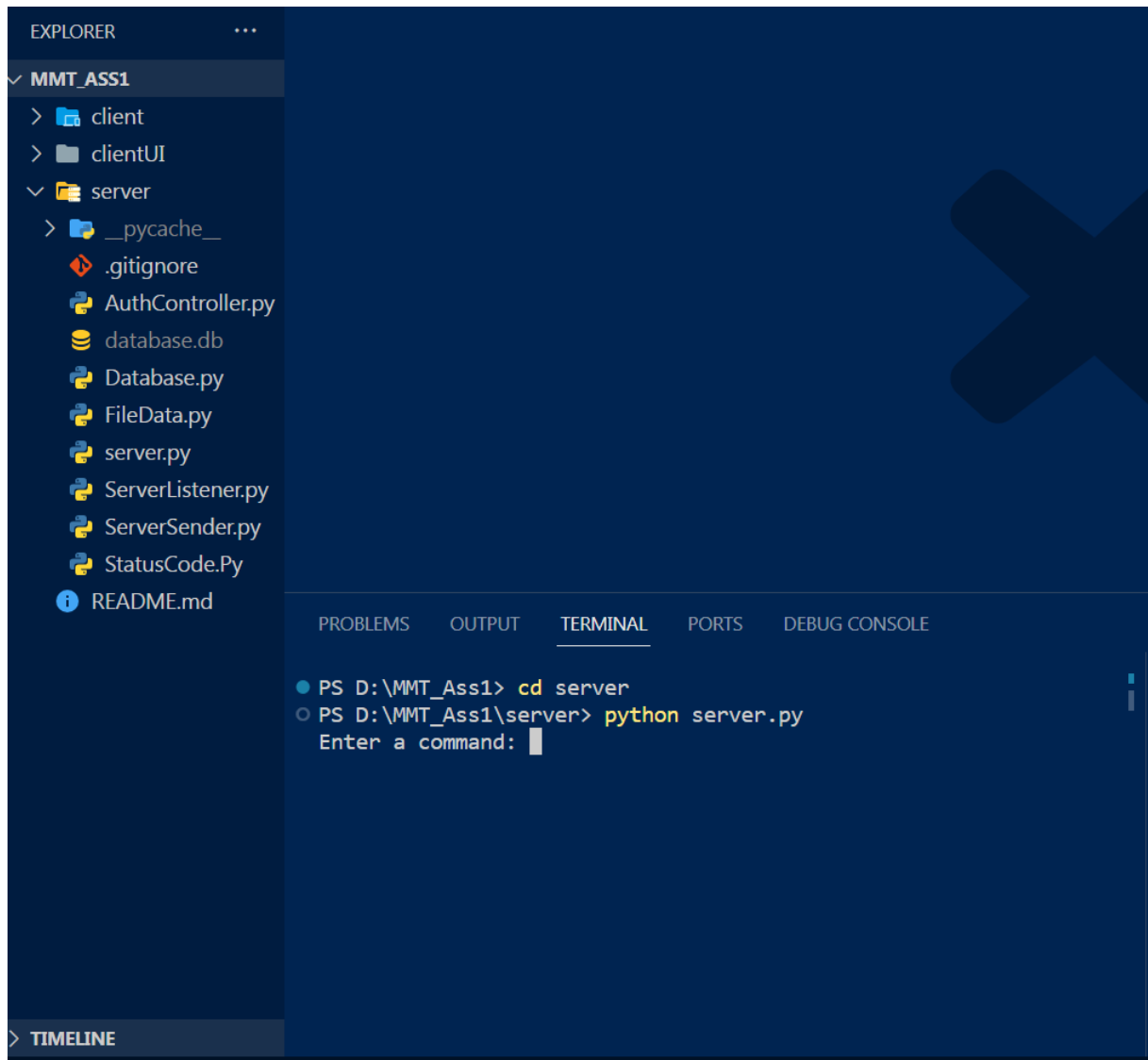
7. Tutorial - Guides

7.1. Server

Người dùng có thể start server bằng cách chạy file server.py



Hình 22: Start Server

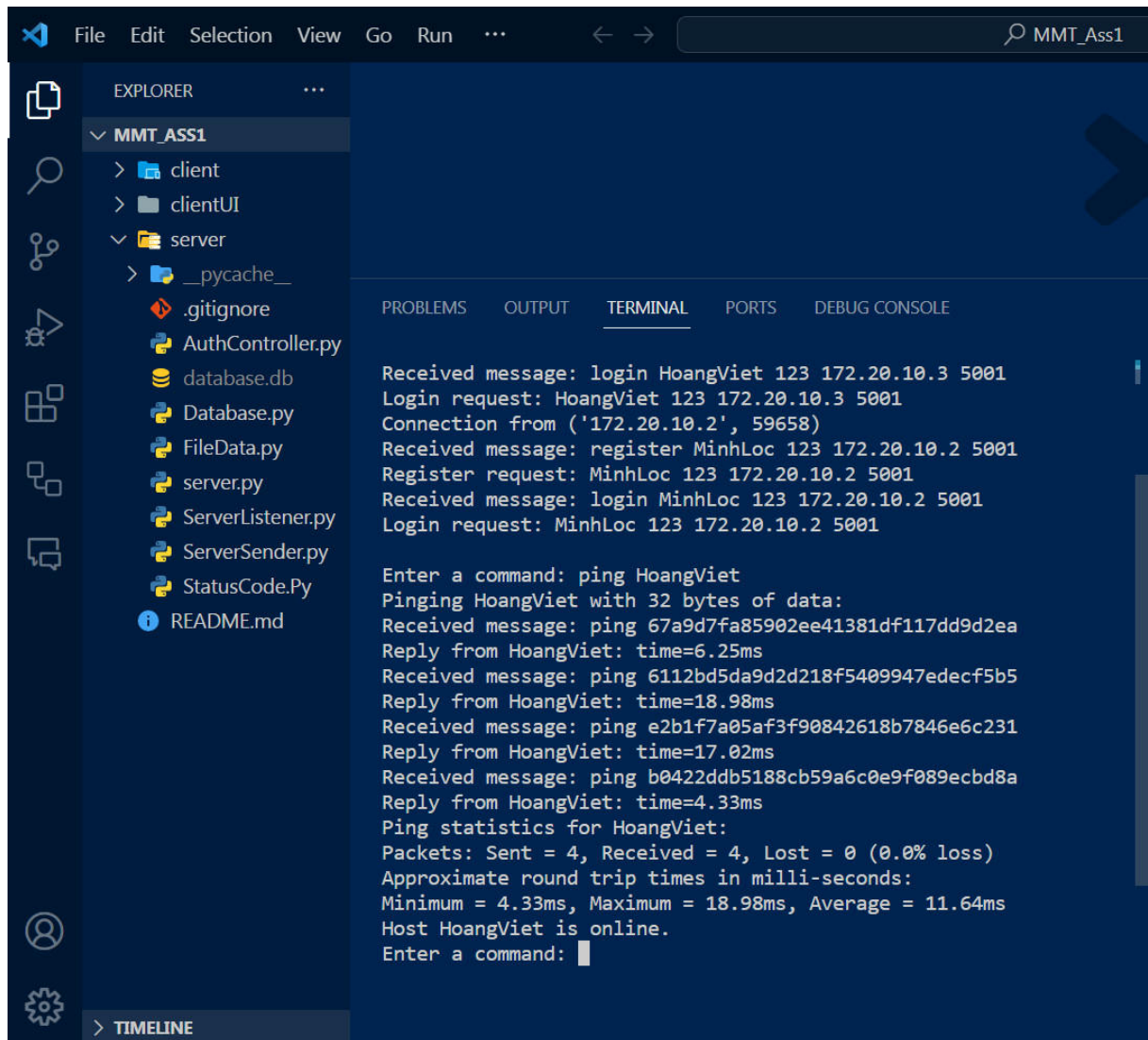


Hình 23: Start Server

7.1.1. ping

Server có thể ping tới các Peer khác bằng cách nhập ping <>

Nếu ping thành công, sẽ hiển thị thông báo client đang hoạt động



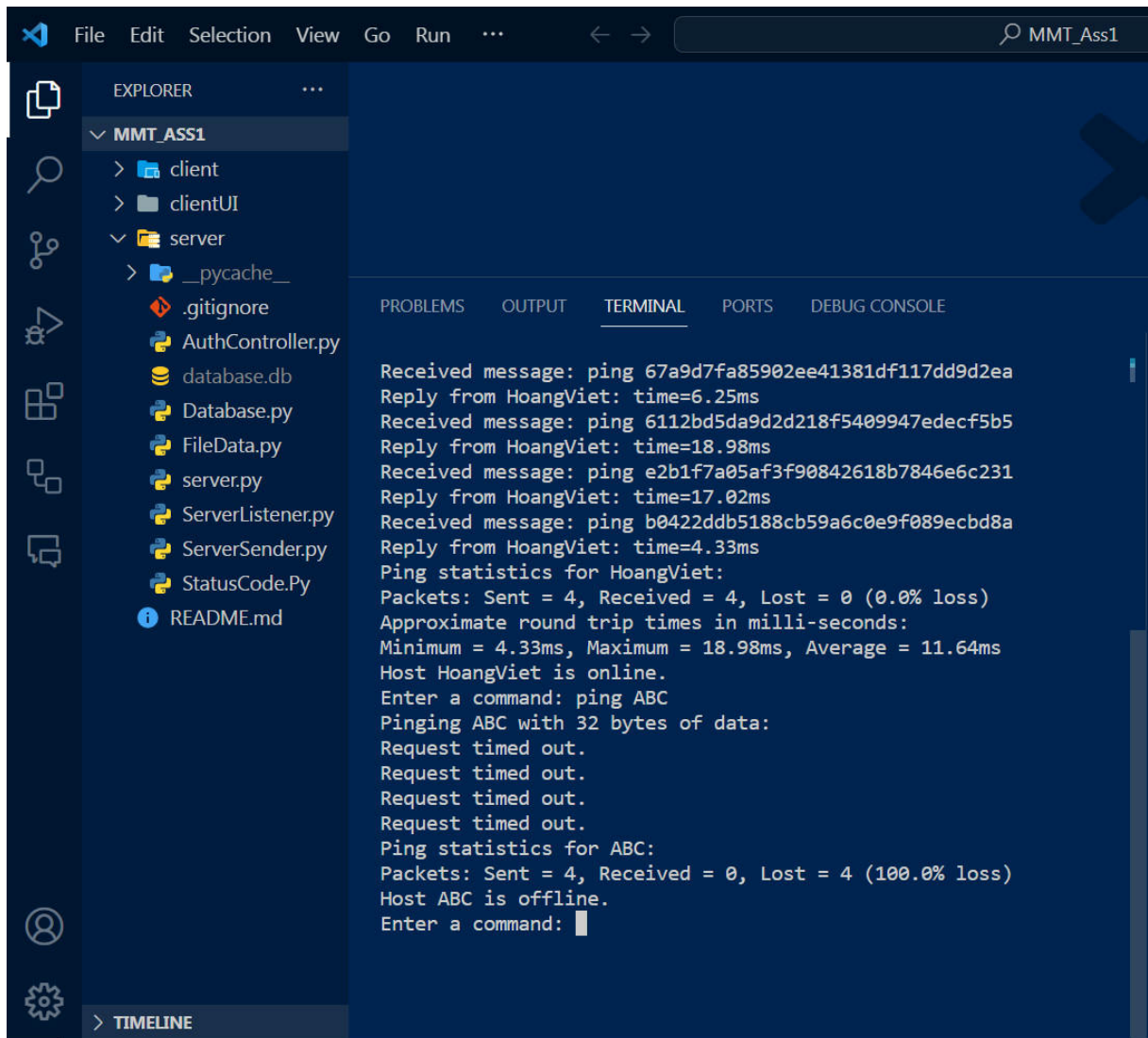
The screenshot shows the Visual Studio Code interface with a project named 'MMT_ASS1'. The Explorer sidebar on the left lists the following files and folders: 'client', 'clientUI', 'server', '__pycache__', '.gitignore', 'AuthController.py', 'database.db', 'Database.py', 'FileData.py', 'server.py', 'ServerListener.py', 'ServerSender.py', 'StatusCode.py', and 'README.md'. The Terminal panel on the right is active and displays the following output:

```
Received message: login HoangViet 123 172.20.10.3 5001
Login request: HoangViet 123 172.20.10.3 5001
Connection from ('172.20.10.2', 59658)
Received message: register MinhLoc 123 172.20.10.2 5001
Register request: MinhLoc 123 172.20.10.2 5001
Received message: login MinhLoc 123 172.20.10.2 5001
Login request: MinhLoc 123 172.20.10.2 5001

Enter a command: ping HoangViet
Pinging HoangViet with 32 bytes of data:
Received message: ping 67a9d7fa85902ee41381df117dd9d2ea
Reply from HoangViet: time=6.25ms
Received message: ping 6112bd5da9d2d218f5409947edecf5b5
Reply from HoangViet: time=18.98ms
Received message: ping e2b1f7a05af3f90842618b7846e6c231
Reply from HoangViet: time=17.02ms
Received message: ping b0422ddb5188cb59a6c0e9f089ecbd8a
Reply from HoangViet: time=4.33ms
Ping statistics for HoangViet:
Packets: Sent = 4, Received = 4, Lost = 0 (0.0% loss)
Approximate round trip times in milli-seconds:
Minimum = 4.33ms, Maximum = 18.98ms, Average = 11.64ms
Host HoangViet is online.
Enter a command: 
```

Hình 24: Ping

Nếu ping thất bại, sẽ hiển thị thông báo client không hoạt động

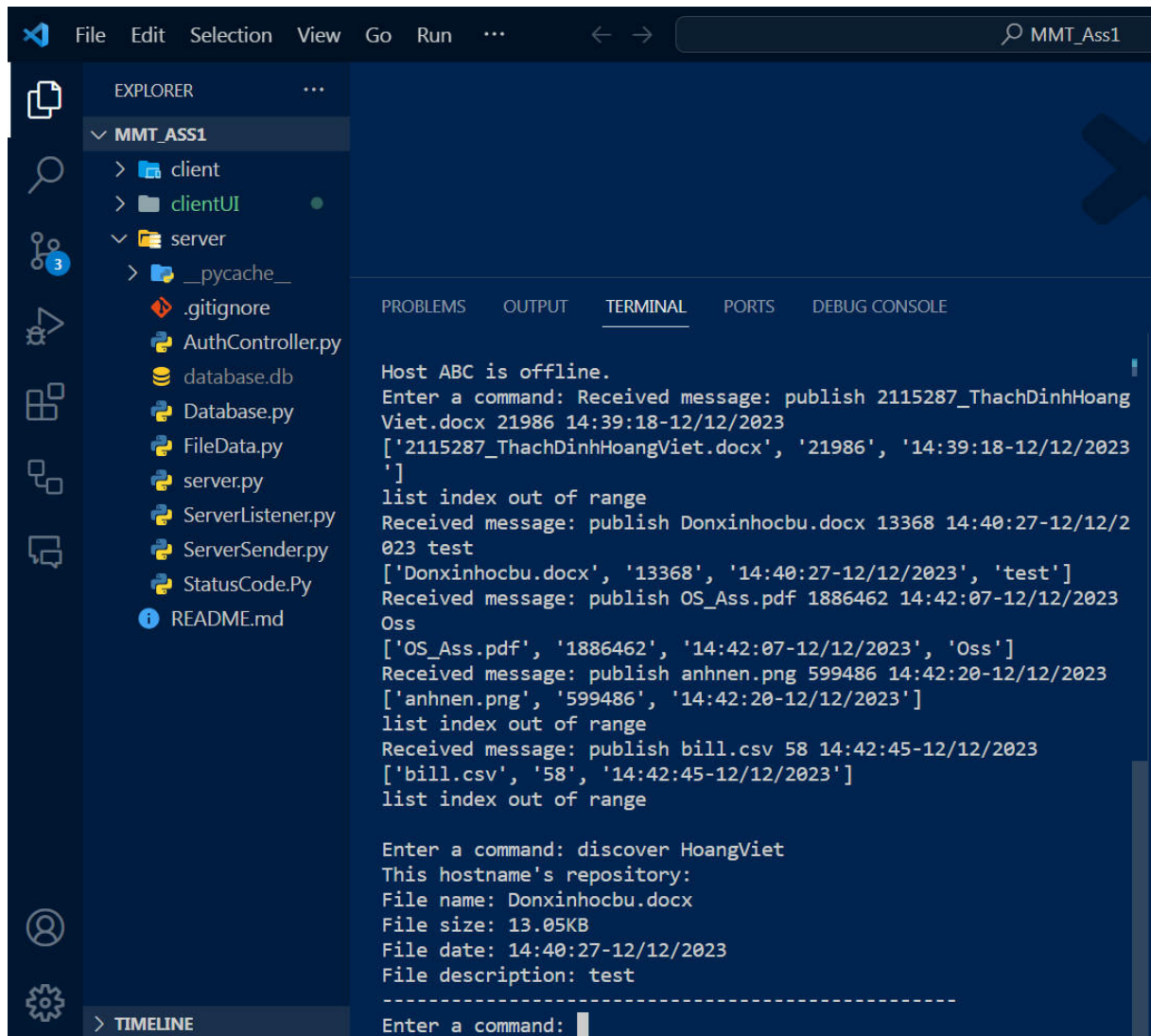


```
Received message: ping 67a9d7fa85902ee41381df117dd9d2ea
Reply from HoangViet: time=6.25ms
Received message: ping 6112bd5da9d2d218f5409947edecf5b5
Reply from HoangViet: time=18.98ms
Received message: ping e2b1f7a05af3f90842618b7846e6c231
Reply from HoangViet: time=17.02ms
Received message: ping b0422ddb5188cb59a6c0e9f089ecbd8a
Reply from HoangViet: time=4.33ms
Ping statistics for HoangViet:
Packets: Sent = 4, Received = 4, Lost = 0 (0.0% loss)
Approximate round trip times in milli-seconds:
Minimum = 4.33ms, Maximum = 18.98ms, Average = 11.64ms
Host HoangViet is online.
Enter a command: ping ABC
Pinging ABC with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for ABC:
Packets: Sent = 4, Received = 0, Lost = 4 (100.0% loss)
Host ABC is offline.
Enter a command:
```

Hình 25: Ping No Online

7.1.2. Discover

Server có thể discover các Peer khác bằng cách nhập discover



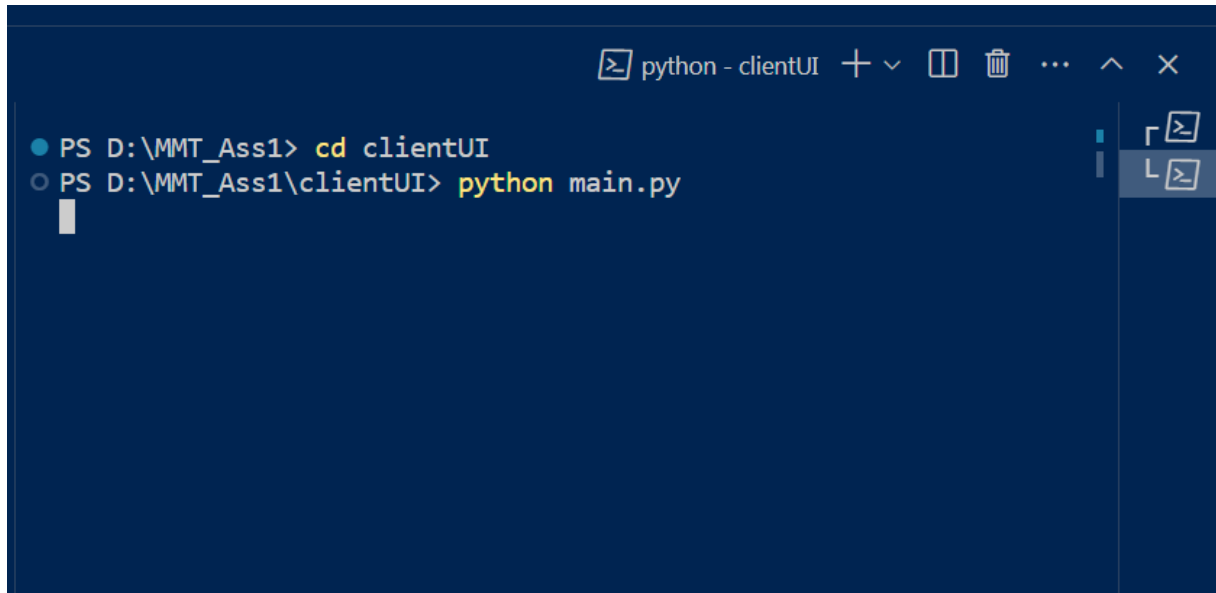
```
Host ABC is offline.
Enter a command: Received message: publish 2115287_ThachDinhHoangViet.docx 21986 14:39:18-12/12/2023
['2115287_ThachDinhHoangViet.docx', '21986', '14:39:18-12/12/2023']
list index out of range
Received message: publish Donxinhocbu.docx 13368 14:40:27-12/12/2023
test
['Donxinhocbu.docx', '13368', '14:40:27-12/12/2023', 'test']
Received message: publish OS_Ass.pdf 1886462 14:42:07-12/12/2023
Oss
['OS_Ass.pdf', '1886462', '14:42:07-12/12/2023', 'Oss']
Received message: publish anhnen.png 599486 14:42:20-12/12/2023
['anhnen.png', '599486', '14:42:20-12/12/2023']
list index out of range
Received message: publish bill.csv 58 14:42:45-12/12/2023
['bill.csv', '58', '14:42:45-12/12/2023']
list index out of range

Enter a command: discover HoangViet
This hostname's repository:
File name: Donxinhocbu.docx
File size: 13.05KB
File date: 14:40:27-12/12/2023
File description: test
-----
Enter a command: 
```

Hình 26: Discover

7.2. Client

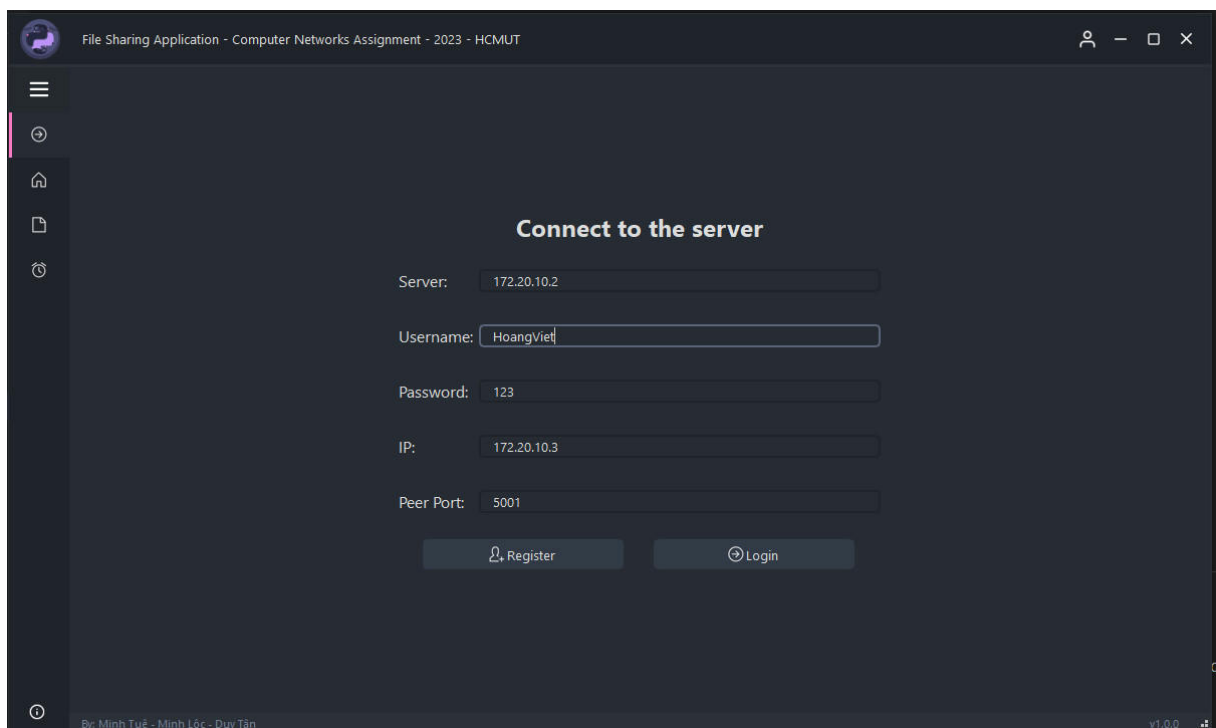
Người dùng có thể start client bằng cách chạy file main.py



Hình 27: Start Client

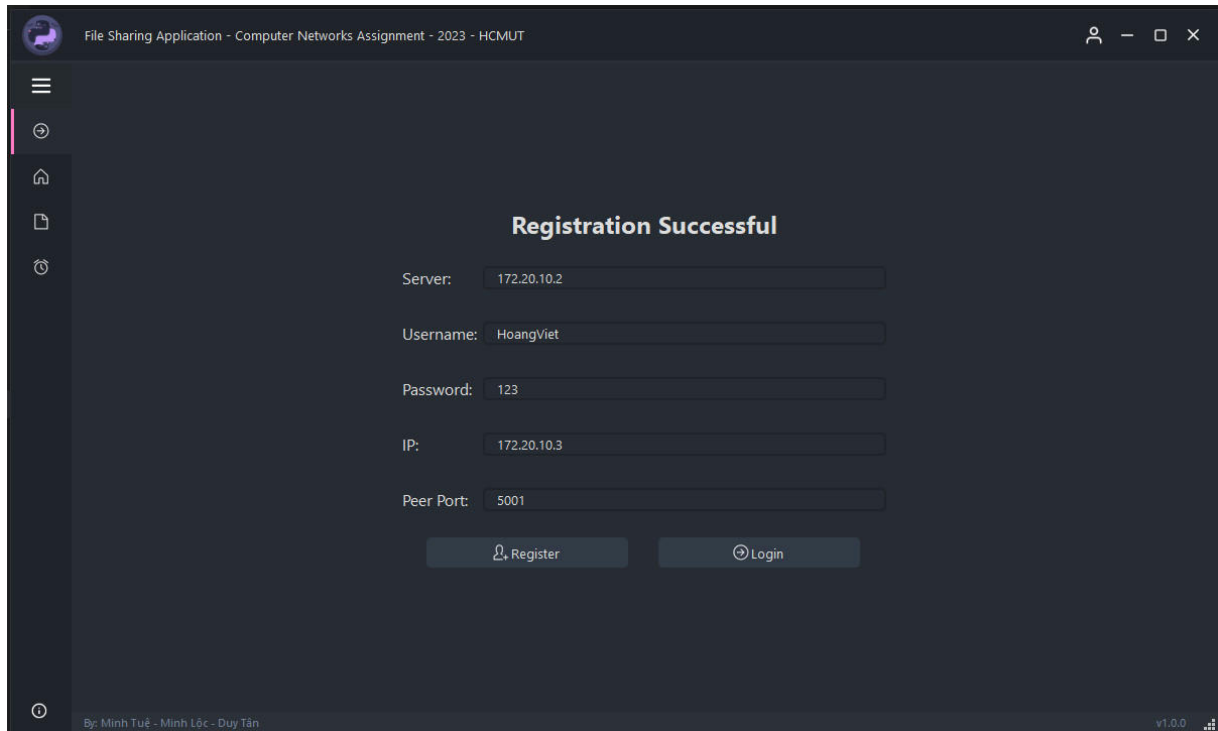
7.2.1. Register

Người dùng sẽ nhập IP Server, username, password, địa chỉ IP Peer, địa chỉ Peer Port để đăng ký tài khoản trên hệ thống.



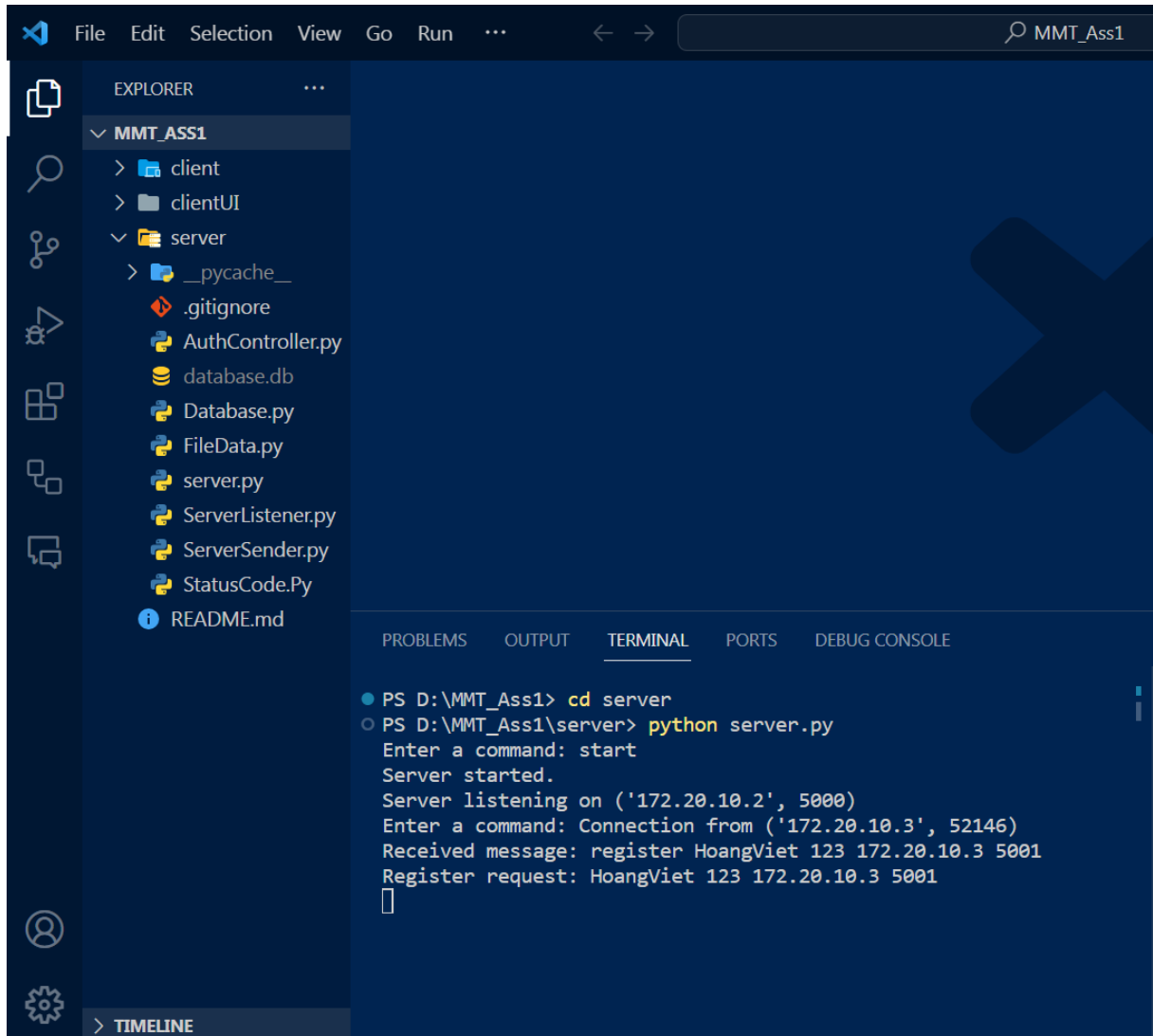
Hình 28: Client Register

Giao diện sẽ hiển thị thông báo đăng ký thành công



Hình 29: Client Register Success

Phía Server sẽ hiển thị thông báo đăng ký thành công



```
File Edit Selection View Go Run ... MMT_Ass1

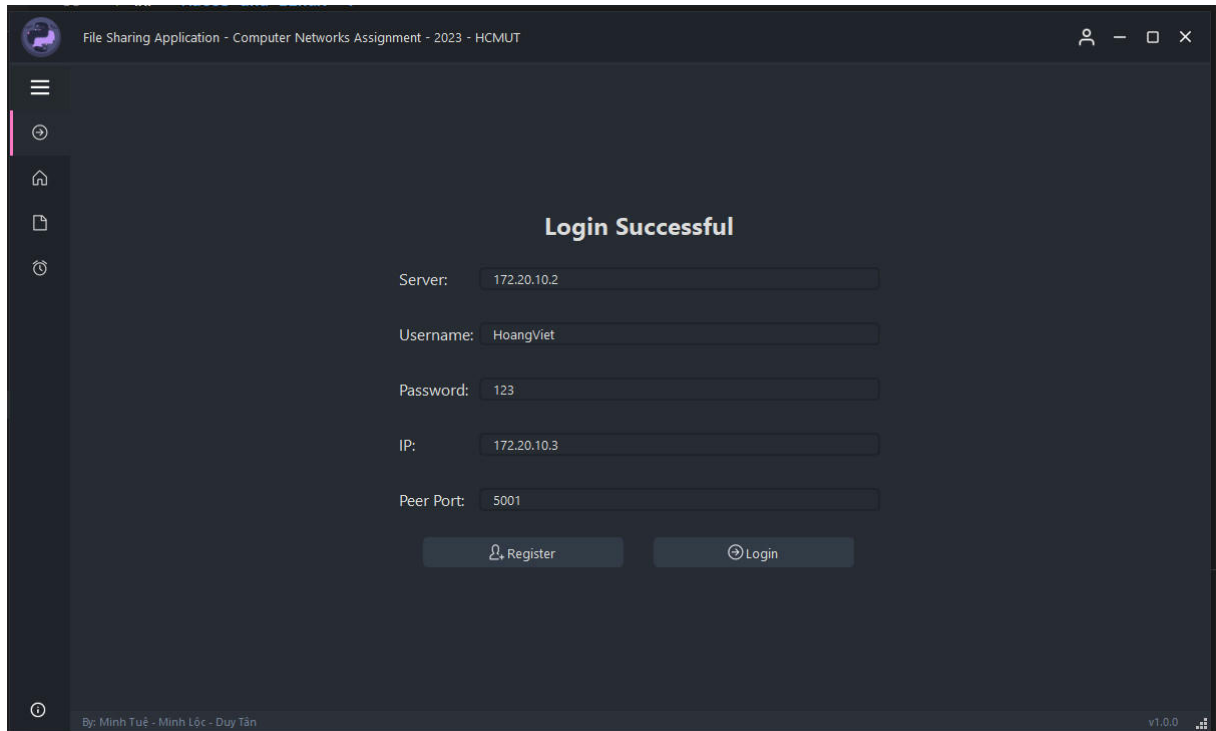
EXPLORER
  MMT_Ass1
    client
    clientUI
    server
      __pycache__
      .gitignore
      AuthController.py
      database.db
      Database.py
      FileData.py
      server.py
      ServerListener.py
      ServerSender.py
      StatusCode.py
      README.md

TERMINAL
  PS D:\MMT_Ass1> cd server
  PS D:\MMT_Ass1\server> python server.py
  Enter a command: start
  Server started.
  Server listening on ('172.20.10.2', 5000)
  Enter a command: Connection from ('172.20.10.3', 52146)
  Received message: register HoangViet 123 172.20.10.3 5001
  Register request: HoangViet 123 172.20.10.3 5001
  [ ]
```

Hình 30: Server Register Success

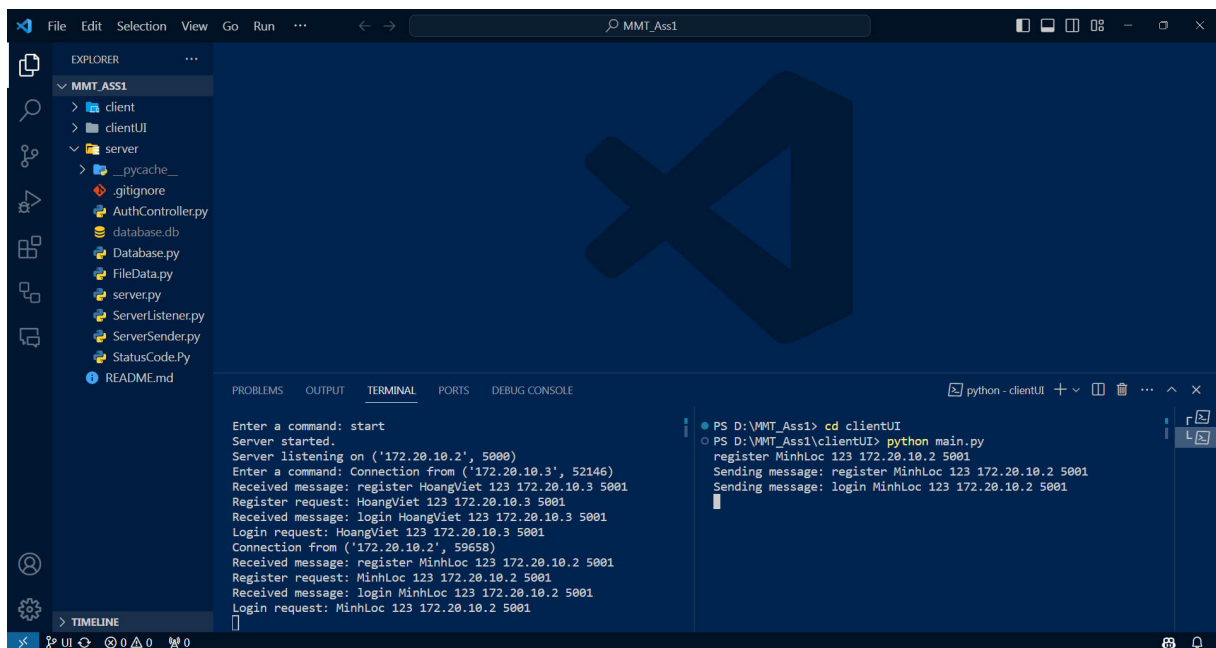
7.2.2. Login

Người dùng sẽ nhập IP Server, username, password, địa chỉ IP Peer, địa chỉ Peer Port để đăng nhập vào hệ thống.



Hình 31: Client Login

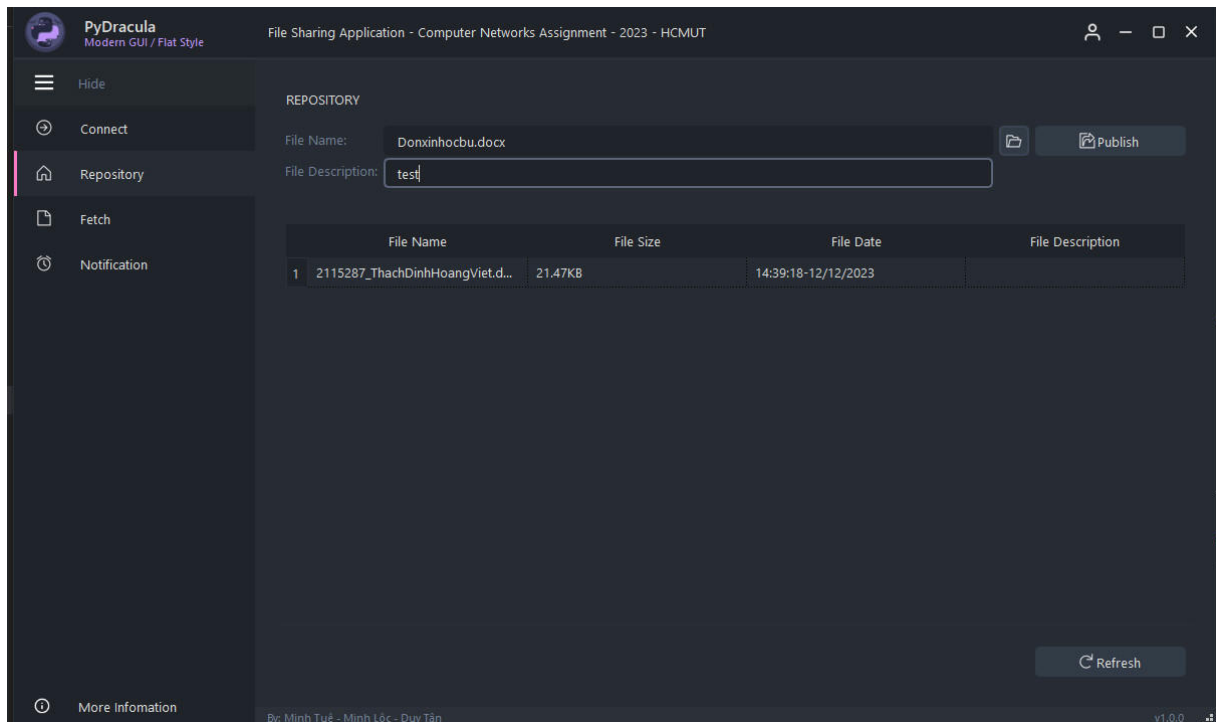
Phía Server sẽ hiển thị thông báo đăng nhập thành công



Hình 32: Login Success

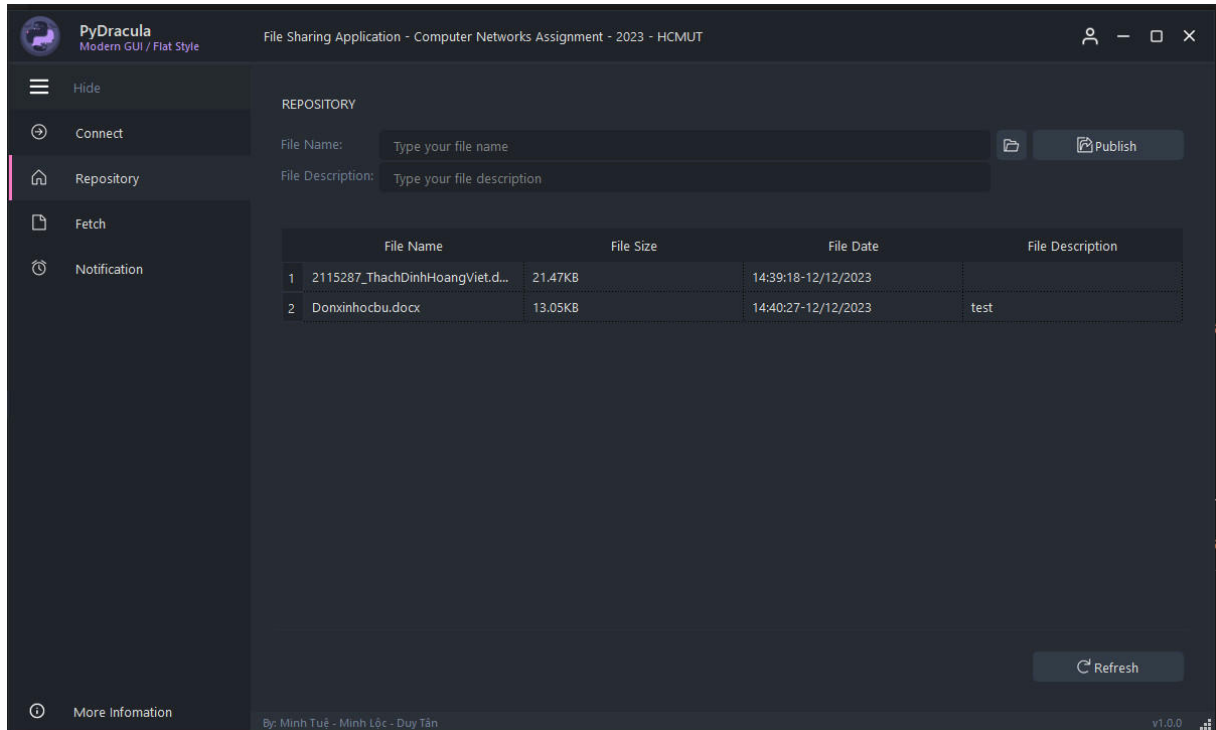
7.2.3. Publish

Người dùng sẽ nhập đường dẫn file, mô tả file và nhấn nút Publish để Publish file lên hệ thống.



Hình 33: Publish

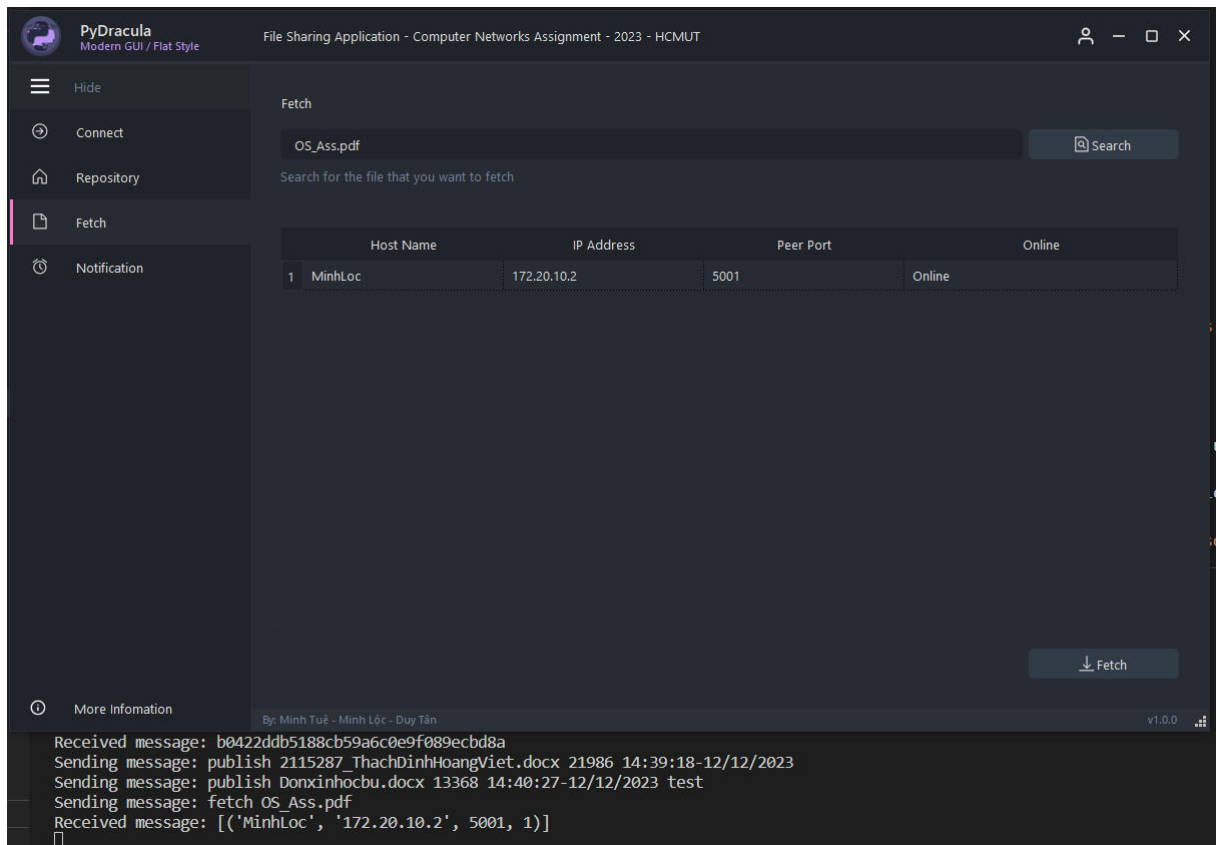
Giao diện sẽ hiển thị file đã Publish vào danh sách



Hình 34: Publish Success

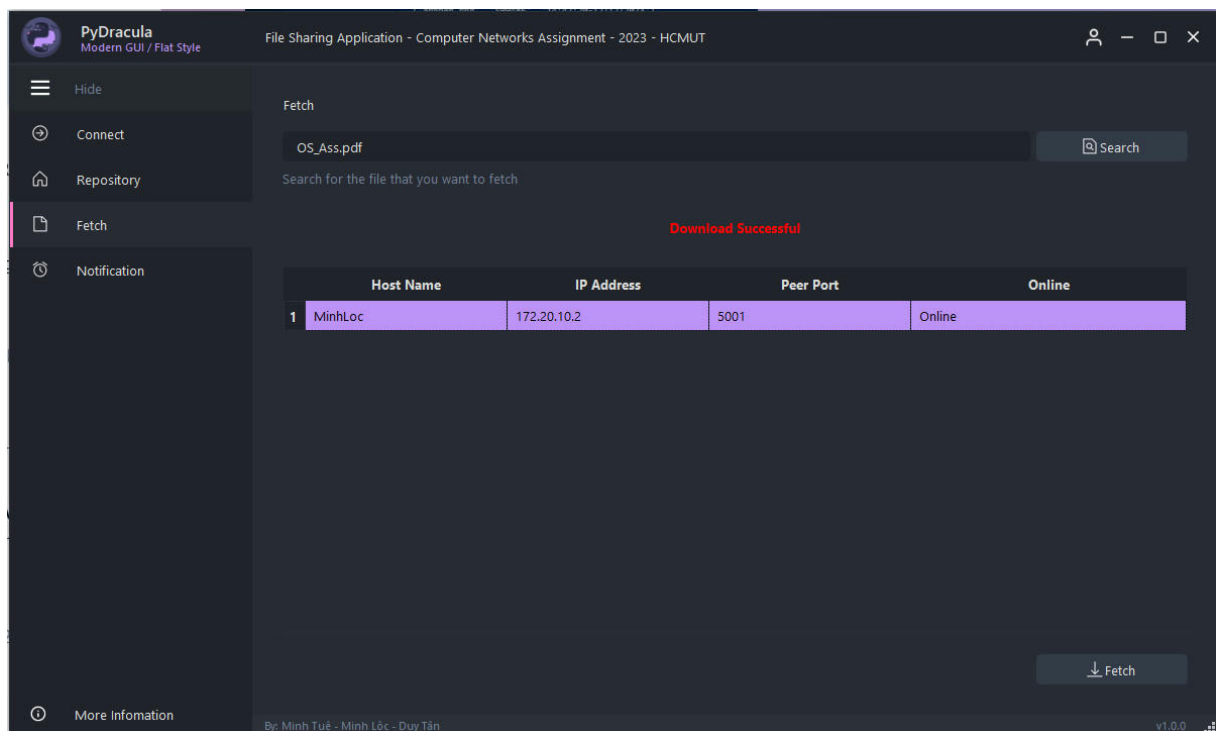
7.2.4. Fetch

Người dùng sẽ nhập tên file cần Fetch và nhấn nút Search để tìm kiếm file.



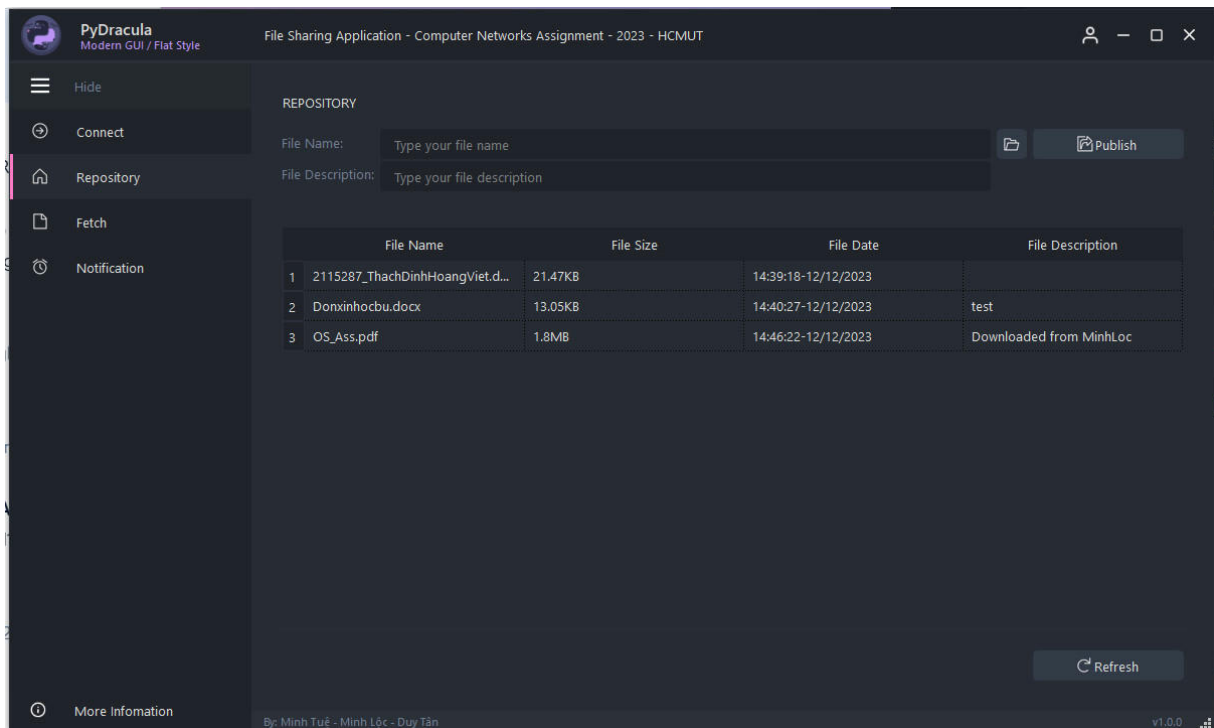
Hình 35: Fetch

Sau khi chọn file cần Fetch và nhấn nút Fetch, file sẽ được Fetch về.



Hình 36: Fetch Success

Lưới danh sách file sẽ hiển thị file đã Fetch về.

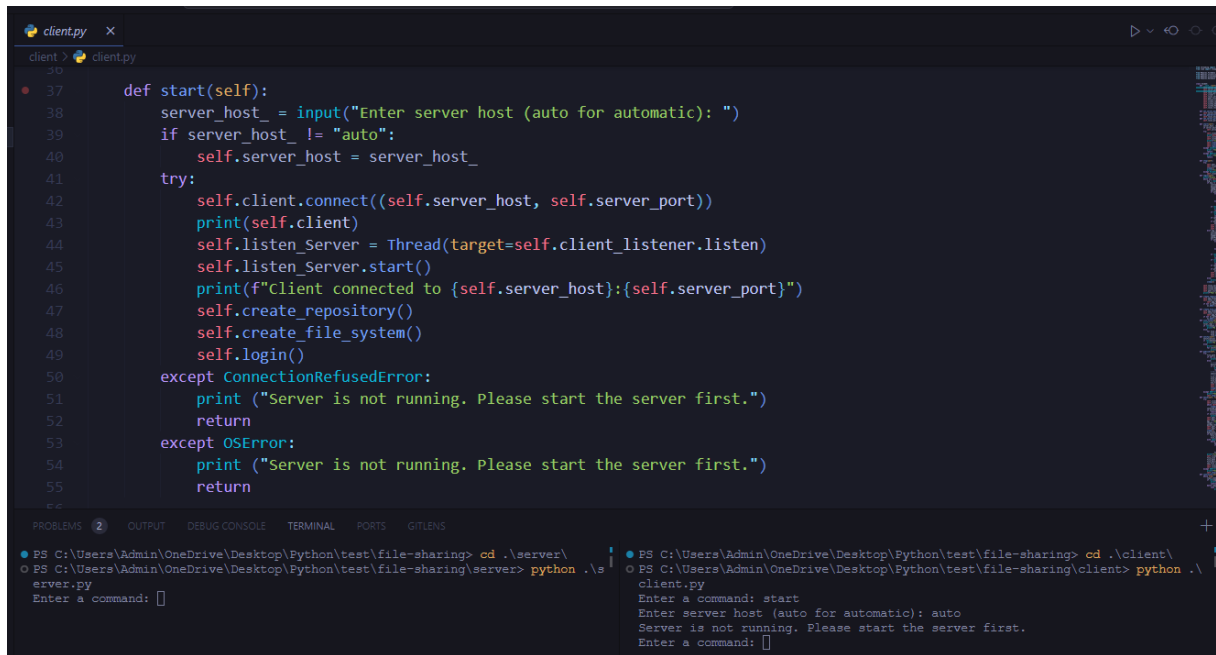


Hình 37: Fetch Success

8. Error Handling

8.1. Server not running

Khi Client kết nối tới Server, nếu Server không hoạt động, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu khởi động lại Server.

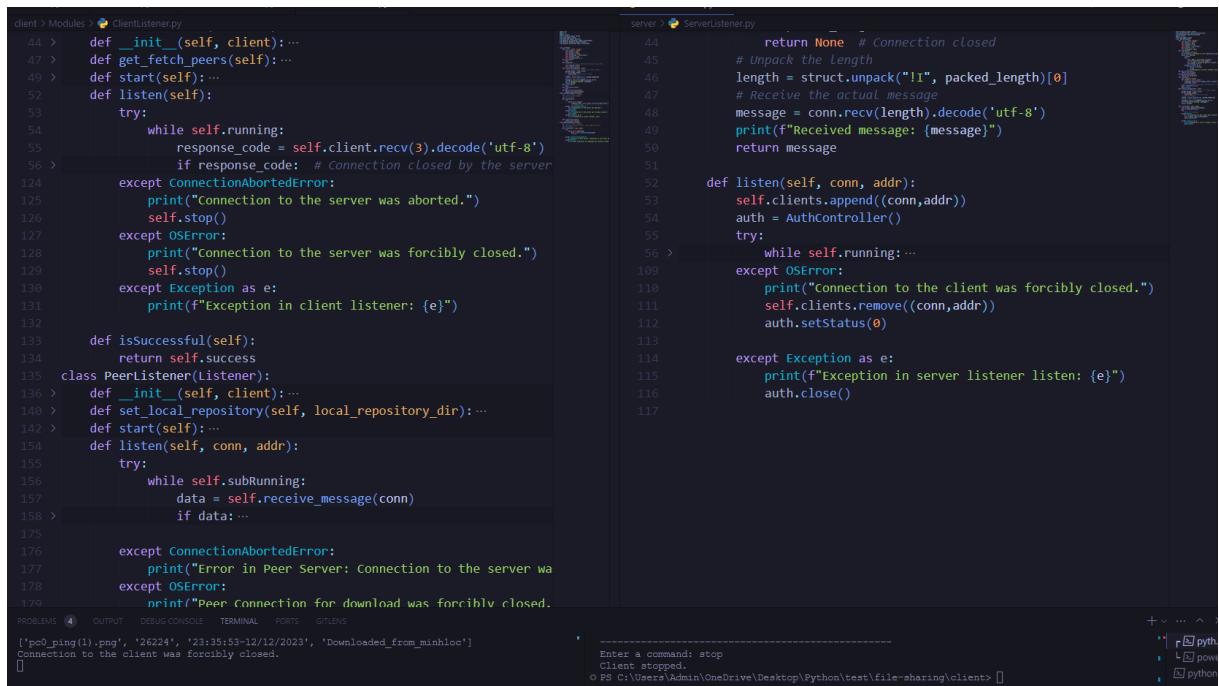


```
client.py
client > client.py
37 def start(self):
38     server_host_ = input("Enter server host (auto for automatic): ")
39     if server_host_ != "auto":
40         self.server_host = server_host_
41     try:
42         self.client.connect((self.server_host, self.server_port))
43         print(self.client)
44         self.listen_Server = Thread(target=self.client_listener.listen)
45         self.listen_Server.start()
46         print(f"Client connected to {self.server_host}:{self.server_port}")
47         self.create_repository()
48         self.create_file_system()
49         self.login()
50     except ConnectionRefusedError:
51         print("Server is not running. Please start the server first.")
52         return
53     except OSError:
54         print("Server is not running. Please start the server first.")
55         return
56
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing> cd .\server\
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\server> python .\server.py
Enter a command:
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing> cd .\client\
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\client> python .\client.py
Enter a command: start
Enter server host (auto for automatic): auto
Server is not running. Please start the server first.
Enter a command:
```

Hình 38: Server not running

8.2. Connection error

Khi một trong hai kết nối giữa Client và Server bị lỗi, hệ thống sẽ hiển thị thông báo lỗi ở mỗi phía. Nếu là lỗi ảnh hưởng đến hệ thống, hệ thống sẽ tự động tắt. Nếu là lỗi không ảnh hưởng đến hệ thống, hệ thống sẽ hiển thị thông báo lỗi và tiếp tục hoạt động.



```
client.py
44 > def __init__(self, client):...
47 > def get_fetch_peers(self):...
49 > def start(self):...
52 def listen(self):
53     try:
54         while self.running:
55             response_code = self.client.recv(3).decode('utf-8')
56             if response_code: # Connection closed by the server
124         except ConnectionAbortedError:
125             print("Connection to the server was aborted.")
126             self.stop()
127         except OSError:
128             print("Connection to the server was forcibly closed.")
129             self.stop()
130         except Exception as e:
131             print(f"Exception in client listener: {e}")
132
133     def isSuccessful(self):
134         return self.success
135
136 class PeerListener(Listener):
137     def __init__(self, client):...
140     def set_local_repository(self, local_repository_dir):...
142     def start(self):...
144     def listen(self, conn, addr):
145         try:
146             while self.subRunning:
147                 data = self.receive_message(conn)
148                 if data: ...
175
176         except ConnectionAbortedError:
177             print("Error in Peer Server: Connection to the server wa
178         except OSError:
179             print("Peer connection for download was forcibly closed.

server.py
44         return None # Connection closed
45         # Unpack the length
46         length = struct.unpack("I", packed_length)[0]
47         # Receive the actual message
48         message = conn.recv(length).decode('utf-8')
49         print(f"Received message: {message}")
50         return message
51
52     def listen(self, conn, addr):
53         self.clients.append((conn,addr))
54         auth = AuthController()
55         try:
56             while self.running:...
109         except OSError:
110             print("Connection to the client was forcibly closed.")
111             self.clients.remove((conn,addr))
112             auth.setStatus(0)
113
114         except Exception as e:
115             print(f"Exception in server listener listen: {e}")
116             auth.close()
117
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[{"peo_ping(1).png", "26224", "23:35:53-12/12/2023", "Downloaded_from_minhloc"}]
Connection to the client was forcibly closed.
[]
```

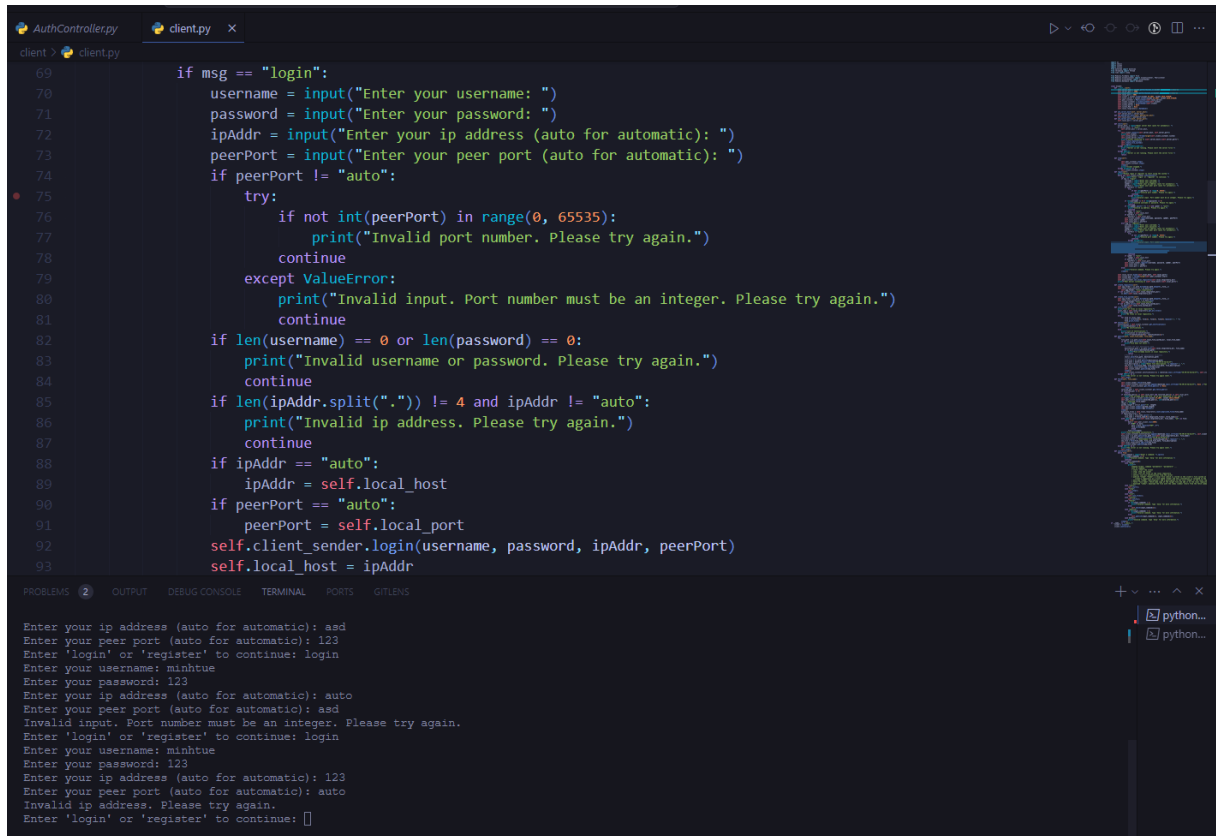
Enter a command: stop
Client stopped.
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\client>

Hình 39: Connection error

8.3. Login & Register: Wrong input format

Hệ thống sẽ kiểm tra các trường nhập vào có đúng định dạng hay không.

Nếu không đúng định dạng, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.



```
client.py
69     if msg == "login":
70         username = input("Enter your username: ")
71         password = input("Enter your password: ")
72         ipAddr = input("Enter your ip address (auto for automatic): ")
73         peerPort = input("Enter your peer port (auto for automatic): ")
74         if peerPort != "auto":
75             try:
76                 if not int(peerPort) in range(0, 65535):
77                     print("Invalid port number. Please try again.")
78                     continue
79             except ValueError:
80                 print("Invalid input. Port number must be an integer. Please try again.")
81                 continue
82         if len(username) == 0 or len(password) == 0:
83             print("Invalid username or password. Please try again.")
84             continue
85         if len(ipAddr.split(".")) != 4 and ipAddr != "auto":
86             print("Invalid ip address. Please try again.")
87             continue
88         if ipAddr == "auto":
89             ipAddr = self.local_host
90         if peerPort == "auto":
91             peerPort = self.local_port
92         self.client_sender.login(username, password, ipAddr, peerPort)
93         self.local_host = ipAddr

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS
Enter your ip address (auto for automatic): asd
Enter your peer port (auto for automatic): 123
Enter 'login' or 'register' to continue: login
Enter your username: minhtue
Enter your password: 123
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): asd
Invalid input. Port number must be an integer. Please try again.
Enter 'login' or 'register' to continue: login
Enter your username: minhtue
Enter your password: 123
Enter your ip address (auto for automatic): 123
Enter your peer port (auto for automatic): auto
Invalid ip address. Please try again.
Enter 'login' or 'register' to continue: []
```

Hình 40: Wrong input format

8.4. Register: User already exist

Hệ thống sẽ kiểm tra xem username đã tồn tại hay chưa.

Nếu đã tồn tại, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.

```
9
10 def login(self, hostname, password, ipAddress, peerPort):
11     print(f"Login request: {hostname} {password} {ipAddress} {peerPort}")
12     self.user = self.database.get_user(hostname)
13     print(self.user)
14     if (not self.user):
15         return self.response_code.USER_NOT_FOUND()
16     elif (self.user[2] != password):
17         return self.response_code.WRONG_PASSWORD()
18     elif (self.user[5] == 1):
19         return self.response_code.USER_ALREADY_ONLINE()
20     else:
21         self.database.update_user(hostname, ipAddress, peerPort)
22         return self.response_code.LOGIN_SUCCESS()
23
24 def register(self, hostname, password, ipAddress, peerPort):
25     print(f"Register request: {hostname} {password} {ipAddress} {peerPort}")
26     if (self.database.get_user(hostname) != None):
27         return self.response_code.USER_ALREADY_EXISTS()
28     else:
```

Enter 'login' or 'register' to continue: register
Enter your username: minhtue
Enter your password: 123
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): auto
Sending message: register minhtue 123 192.168.1.8 5001
Received response code: 201
Registration successful
Enter 'login' or 'register' to continue: register
Enter your username: minhtue
Enter your password: 123
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): auto
Sending message: register minhtue 123 192.168.1.8 5001
Received response code: 404
User already exists.
Enter 'login' or 'register' to continue:

Hình 41: User already exist

8.5. Login: User not found

Hệ thống sẽ kiểm tra xem username đã tồn tại hay chưa.

Nếu chưa tồn tại, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.

```
9
10 def login(self, hostname, password, ipAddress, peerPort):
11     print(f"Login request: {hostname} {password} {ipAddress} {peerPort}")
12     self.user = self.database.get_user(hostname)
13     print(self.user)
14     if (not self.user):
15         return self.response_code.USER_NOT_FOUND()
16     elif (self.user[2] != password):
17         return self.response_code.WRONG_PASSWORD()
18     elif (self.user[5] == 1):
19         return self.response_code.USER_ALREADY_ONLINE()
20     else:
21         self.database.update_user(hostname, ipAddress, peerPort)
22         return self.response_code.LOGIN_SUCCESS()
23
```

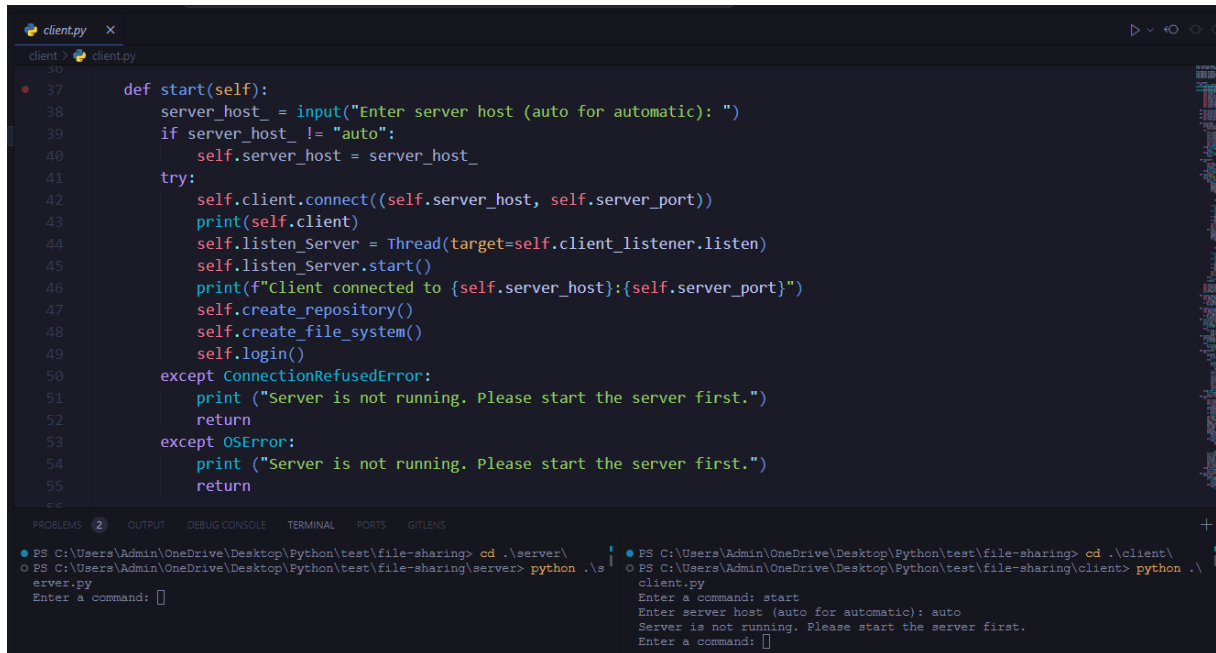
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing> cd .\client\
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\client> python .\client.py
Enter a command: start
Enter server host (auto for automatic): auto
csocket.socket fd=544, family=2, type=1, proto=0, laddr=('192.168.1.8', 63000), raddr=('192.168.1.8', 5000)>
Client connected to 192.168.1.8:5000
Please login or register to start using the system.
Enter 'login' or 'register' to continue: login
Enter your username: minhtue
Enter your password: 123
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): auto
Sending message: login minhtue 123 192.168.1.8 5001
Received response code: 401
User not found.
Enter 'login' or 'register' to continue:

Hình 42: User not found

8.6. Login: Wrong password

Hệ thống sẽ kiểm tra xem password có đúng hay không.

Nếu không đúng, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.



```
client.py
def start(self):
    server_host_ = input("Enter server host (auto for automatic): ")
    if server_host_ != "auto":
        self.server_host = server_host_
    try:
        self.client.connect((self.server_host, self.server_port))
        print(self.client)
        self.listen_Server = Thread(target=self.client_listener.listen)
        self.listen_Server.start()
        print(f"Client connected to {self.server_host}:{self.server_port}")
        self.create_repository()
        self.create_file_system()
        self.login()
    except ConnectionRefusedError:
        print("Server is not running. Please start the server first.")
        return
    except OSError:
        print("Server is not running. Please start the server first.")
        return
```

```
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing> cd .\server\
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\server> python .\server.py
Enter a command:

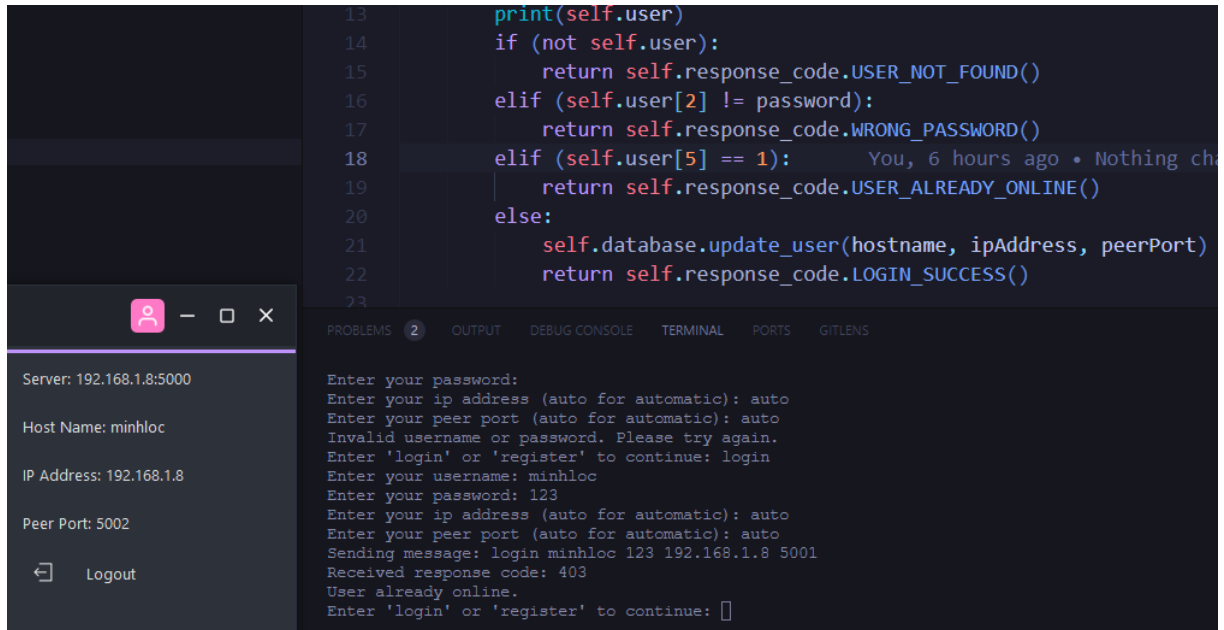
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing> cd .\client\
PS C:\Users\Admin\OneDrive\Desktop\Python\test\file-sharing\client> python .\client.py
Enter a command: start
Enter server host (auto for automatic): auto
Server is not running. Please start the server first.
Enter a command:
```

Hình 43: Wrong password

8.7. Login: User already online

Hệ thống sẽ kiểm tra xem username đã đăng nhập hay chưa.

Nếu đã đăng nhập, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.



```
13     print(self.user)
14     if (not self.user):
15         return self.response_code.USER_NOT_FOUND()
16     elif (self.user[2] != password):
17         return self.response_code.WRONG_PASSWORD()
18     elif (self.user[5] == 1):      You, 6 hours ago • Nothing changed
19         return self.response_code.USER_ALREADY_ONLINE()
20     else:
21         self.database.update_user(hostname, ipAddress, peerPort)
22         return self.response_code.LOGIN_SUCCESS()
23
```

Server: 192.168.1.8:5000

Host Name: minhloc

IP Address: 192.168.1.8

Peer Port: 5002

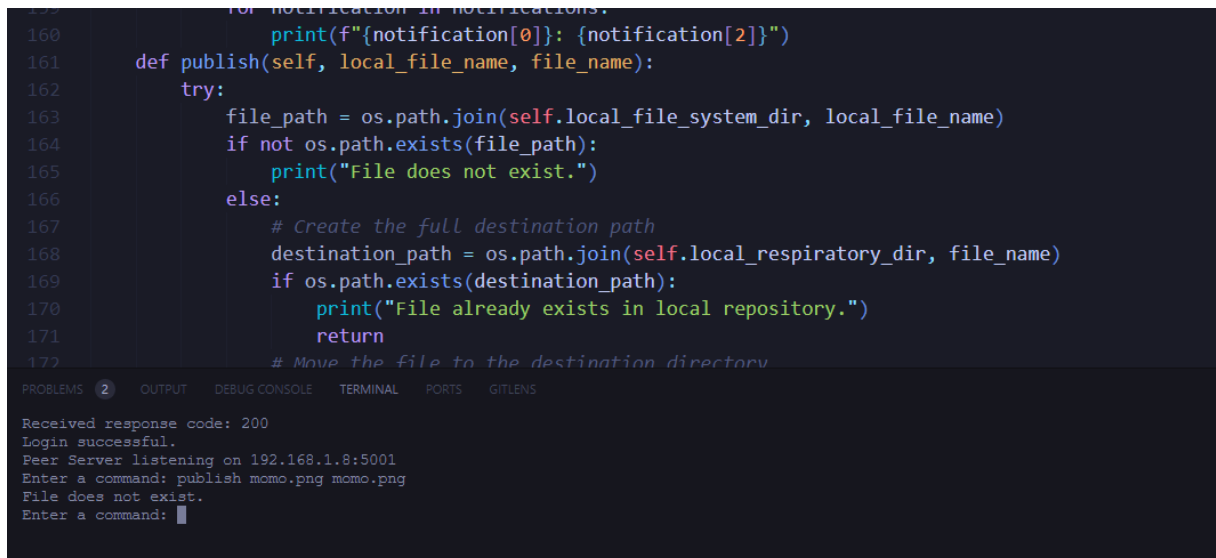
Logout

Enter your password:
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): auto
Invalid username or password. Please try again.
Enter 'login' or 'register' to continue: login
Enter your username: minhloc
Enter your password: 123
Enter your ip address (auto for automatic): auto
Enter your peer port (auto for automatic): auto
Sending message: login minhloc 123 192.168.1.8 5001
Received response code: 403
User already online.
Enter 'login' or 'register' to continue: []

Hình 44: User already online

8.8. Publish: File not found

Hệ thống sẽ kiểm tra xem file có tồn tại hay không. Nếu không tồn tại, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.



```
159         for notification in notifications:
160             print(f"{notification[0]}: {notification[2]}")
161     def publish(self, local_file_name, file_name):
162         try:
163             file_path = os.path.join(self.local_file_system_dir, local_file_name)
164             if not os.path.exists(file_path):
165                 print("File does not exist.")
166             else:
167                 # Create the full destination path
168                 destination_path = os.path.join(self.local_respiratory_dir, file_name)
169                 if os.path.exists(destination_path):
170                     print("File already exists in local repository.")
171                 return
172             # Move the file to the destination directory
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Received response code: 200
Login successful.
Peer Server listening on 192.168.1.8:5001
Enter a command: publish momo.png momo.png
File does not exist.
Enter a command: █

Hình 45: File not found

8.9. Publish: File already published

Hệ thống sẽ kiểm tra xem file đã được Publish hay chưa. Nếu đã được Publish, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.

```
165         print( File does not exist. )
166     else:
167         # Create the full destination path
168         destination_path = os.path.join(self.local_respiratory_dir, file_name)
169         if os.path.exists(destination_path):
170             print("File already exists in local repository.")
171             return
172         # Move the file to the destination directory
173         shutil.move(file_path, destination_path)
174         # Update local_respiratory
175         file_size = os.path.getsize(destination_path)
176         file_date = datetime.now().strftime("%H:%M:%S-%d/%m/%Y")
177         file_description = input("Enter file description: ").replace(" ", "_")
178         new_file = File(file_name, file_size, file_date, file_description)
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

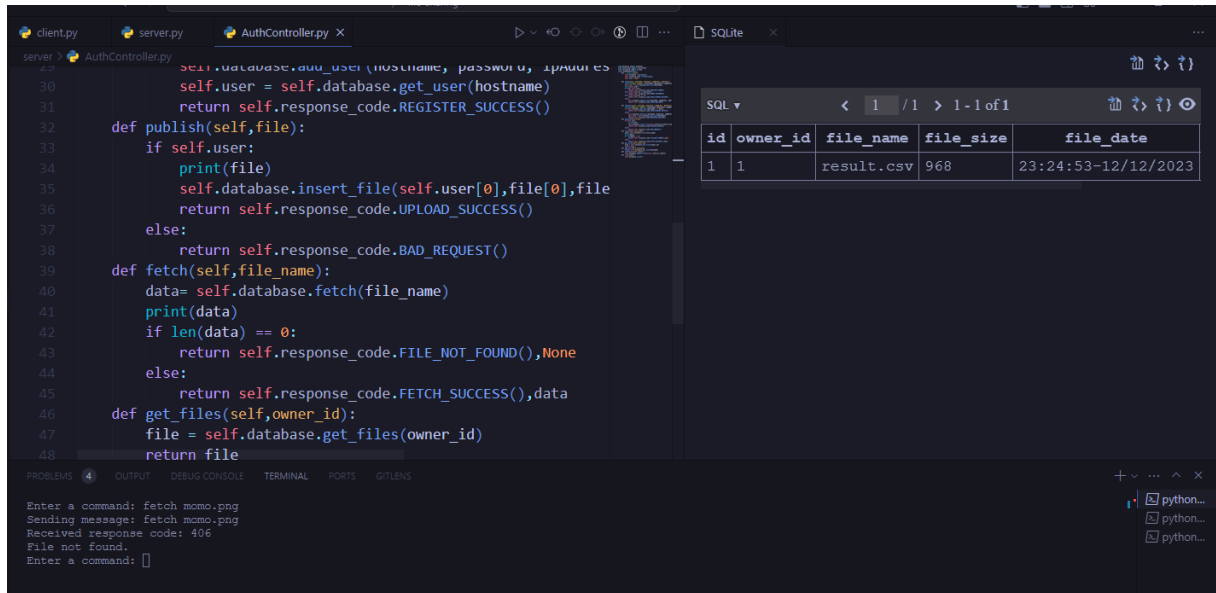
Received response code: 200
Login successful.
Peer Server listening on 192.168.1.8:5001
Enter a command: publish momo.png momo.png
File does not exist.
Enter a command: publis result.csv result.csv
Invalid command. Type 'help' for more information.
Enter a command: publish result.csv result.csv
Enter file description: xstk
Sending message: publish result.csv 968 23:24:53-12/12/2023 xstk
Received response code: 202
File uploaded successfully.
Enter a command: list
List of files in local repository:
File name: result.csv
File size: 968B
File date: 23:24:53-12/12/2023
File description: xstk

Enter a command: publish result.csvvv result.csv
File does not exist.
Enter a command: publish result.csv result.csv
File does not exist.
Enter a command: publish result.csv result.csv
File already exists in local repository.
Enter a command: []

Hình 46: File already published

8.10. Fetch: File not found

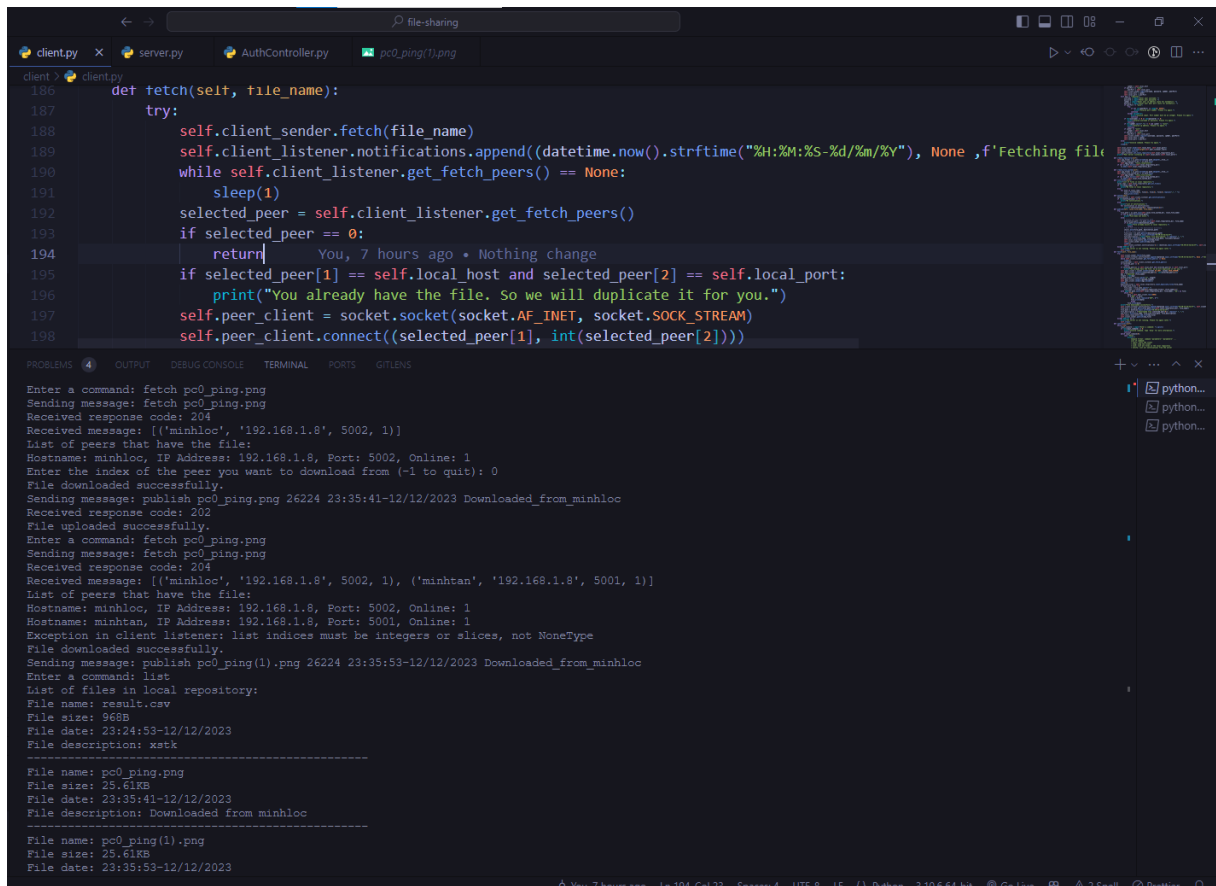
Hệ thống sẽ kiểm tra xem file có tồn tại hay không. Nếu không tồn tại, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại.



Hình 47: File not found

8.11. Fetch: File already fetched

Hệ thống sẽ kiểm tra xem file đã được Fetch hay chưa. Nếu đã được fetch, hệ thống sẽ tự động duplicate file cho người dùng.



```
def fetch(self, file_name):
    try:
        self.client_sender.fetch(file_name)
        self.client_listener.notifications.append((datetime.now().strftime("%H:%M:%S-%d/%m/%Y"), None, f'Fetching file'))
        while self.client_listener.get_fetch_peers() == None:
            sleep(1)
        selected_peer = self.client_listener.get_fetch_peers()
        if selected_peer == 0:
            return You, 7 hours ago • Nothing change
        if selected_peer[1] == self.local_host and selected_peer[2] == self.local_port:
            print("You already have the file. So we will duplicate it for you.")
        self.peer_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.peer_client.connect((selected_peer[1], int(selected_peer[2])))
```

Enter a command: fetch pc0_ping.png
Sending message: fetch pc0_ping.png
Received response code: 204
Received message: [('minhloc', '192.168.1.8', 5002, 1)]
List of peers that have the file:
Hostname: minhloc, IP Address: 192.168.1.8, Port: 5002, Online: 1
Enter the index of the peers you want to download from (~1 to quit): 0
File downloaded successfully.
Sending message: publish pc0_ping.png 26224 23:35:41-12/12/2023 Downloaded_from_minhloc
Received response code: 202
File uploaded successfully.
Enter a command: fetch pc0_ping.png
Sending message: fetch pc0_ping.png
Received response code: 204
Received message: [('minhloc', '192.168.1.8', 5002, 1), ('minhtan', '192.168.1.8', 5001, 1)]
List of peers that have the file:
Hostname: minhloc, IP Address: 192.168.1.8, Port: 5002, Online: 1
Hostname: minhtan, IP Address: 192.168.1.8, Port: 5001, Online: 1
Exception in client listener: list indices must be integers or slices, not NoneType
File downloaded successfully.
Sending message: publish pc0_ping(1).png 26224 23:35:53-12/12/2023 Downloaded_from_minhloc
Enter a command: list
List of files in local repository:
File name: result.csv
File size: 968B
File date: 23:24:53-12/12/2023
File description: xatk

File name: pc0_ping.png
File size: 25.61KB
File date: 23:35:41-12/12/2023
File description: Downloaded from minhloc

File name: pc0_ping(1).png
File size: 25.61KB
File date: 23:35:53-12/12/2023

Hình 48: File already fetched

Tài liệu tham khảo

- [1] Slide môn học Mạng máy tính CO3093.
- [2] ByteByteGo. 8 Popular Network Protocols. Truy cập từ: <https://substackcdn.com>
- [3] Md Rafiul, KabirBhagawat Baanav Yedla RaviBhagawat Baanav Yedla Ravi, Sandip Ray. A Virtual Prototyping Platform for Exploration of Vehicular Electronics. ResearchGate. Truy cập từ: https://www.researchgate.net/publication/370038185_A_Virtual_Prototyping_Platform_for_Exploration_of_Vehicular_Electronics