

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



# MẠNG MÁY TÍNH (CO3094)

## Network Application

GVHD: Nguyễn Phương Duy

Nhóm: L07

SVTH:	Kha Sang	- 2010576
	Nguyễn Huỳnh Tuấn Hưng	- 2011329
	Lê Duy Khang	- 2013425
	Nguyễn Hữu Danh	- 2010174
	Huỳnh Đại Vinh	- 2010785

# Mục lục

<b>1</b>	<b>Bảng phân công nhiệm vụ</b>	<b>3</b>
<b>2</b>	<b>Các chức năng của ứng dụng</b>	<b>4</b>
2.1	Chat với mô hình P2P	4
2.2	Đăng ký - Đăng nhập	4
2.2.1	Đăng ký	4
2.2.2	Đăng nhập	5
2.3	Danh sách bạn bè	5
2.4	Gửi file	5
<b>3</b>	<b>Các protocol của ứng dụng</b>	<b>6</b>
3.1	Chat - Gửi tin nhắn	6
3.2	Server communication - Giao tiếp máy chủ	10
3.3	File transfer - Gửi file	10
<b>4</b>	<b>Mô tả các hàm hiện thực</b>	<b>13</b>
4.1	Các hàm hiện thực Server	13
4.2	Các hàm hiện thực Peer	18
4.2.1	Mô tả về kiến trúc của 1 Peer	18
4.2.2	Các hàm của Peer	19
4.3	Các hàm hỗ trợ đồ họa	20
4.3.1	Thư viện Tkinter	20
4.3.2	Kiến trúc giao diện người dùng (UI) của ứng dụng	20
4.3.3	Các hàm hiện thực UI	20
<b>5</b>	<b>Thiết kế ứng dụng</b>	<b>22</b>
5.1	Kiến trúc sử dụng	22
5.2	Bản vẽ sơ đồ lớp	23
5.3	Mô tả các lớp chính	23
5.3.1	Các lớp dành cho Server	23
5.3.2	Các lớp dành cho Peer	25
5.3.3	Các lớp dành cho UI	28
<b>6</b>	<b>Kiểm thử và đánh giá hiệu năng</b>	<b>31</b>
6.1	Kiểm thử	31
6.1.1	Giao diện	31
6.1.2	Tính năng	31
6.2	Đánh giá hiệu năng	31
6.2.1	Load Testing - Mức độ chịu tải	32
6.2.2	Volume Testing - Giới hạn truyền tin	33
6.2.3	Endurance testing - Mức độ ổn định của hệ thống	34
6.3	Chạy thử trên các máy cùng một subnet	35
6.3.1	Cài đặt để tiến hành chạy thử	35
6.3.2	Thời gian truyền tin	36

<b>7</b>	<b>Hướng dẫn sử dụng</b>	<b>37</b>
7.1	Server . . . . .	37
7.2	Peer . . . . .	37
7.2.1	Register account . . . . .	37
7.2.2	Chat and transfer file . . . . .	41
	<b>References</b>	<b>47</b>

## 1 Bảng phân công nhiệm vụ

Họ tên	Nhiệm vụ
Kha Sang	Hiện thực Server + Login/Register
Nguyễn Huỳnh Tuấn Hưng	Test + Performance Check
Huỳnh Đại Vinh	Hiện thực Peer Chat
Nguyễn Hữu Danh	Gửi file + Manual
Lê Duy Khang	Hỗ trợ liên kết UI

## 2 Các chức năng của ứng dụng

### 2.1 Chat với mô hình P2P

Chức năng chat cho phép các người dùng đã đăng nhập có thể kết nối và gửi tin nhắn dưới dạng văn bản với nhau. Người dùng có thể chọn để gửi tin nhắn với một trong những tài khoản đang online nằm trong danh sách bạn bè. Để có thể bắt đầu một cuộc trò chuyện, người dùng cần phải chọn bắt đầu chat và một request sẽ được gửi đến người dùng còn lại. Người nhận được request có hai lựa chọn:

- Ấn nút "OK": Cuộc trò chuyện được chấp nhận và tạo ra một connection để giao tiếp giữa hai user. Hai người dùng có thể gửi các tin nhắn để giao tiếp với nhau dưới dạng văn bản. Nếu muốn ngắt kết nối, một trong hai người dùng sẽ bấm "Disconnect". Khi đó, connection giữa hai người dùng sẽ được đóng lại.
- Ấn nút "Decline" hoặc không thực hiện thao tác nào trong 10 giây: Sẽ trả về reject cho người gửi yêu cầu và không có bất kỳ connection nào được tạo ra.

Để có thể chat với nhiều người cùng lúc, các cuộc trò chuyện cần phải nằm trong những thread khác nhau và các thread sẽ được tạo ra khi một connection được khởi tạo.

Để người dùng có thể chuyển qua lại giữa các cuộc trò chuyện, nghĩa là chuyển qua lại giữa các thread và các dữ liệu trong từng thread phải được lưu lại và phục hồi khi chuyển lại thread, thì cần phải sử dụng context switch.

Dùng context switch sẽ đáp ứng được tất cả yêu cầu trên khi giúp chuyển qua lại giữa các thread và khi context switch xảy ra, chương trình sẽ lưu lại data và state trước khi context switch và sau đó phục hồi lại data cũng như state được lưu trước đó để thread tiếp tục chạy. Việc này sẽ giúp cho người dùng có thể chuyển qua lại giữa các cuộc trò chuyện đang kết nối và cũng như lưu lại được dữ liệu cuộc trò chuyện trước khi chuyển.

### 2.2 Đăng ký - Đăng nhập

#### 2.2.1 Đăng ký

Chức năng đăng ký cho phép người dùng chưa có tài khoản tạo tài khoản mới. Ứng dụng hiển thị một form gồm 3 field: Username (tên người dùng), password (mật khẩu) và confirm password (xác nhận mật khẩu). Các field này có các ràng buộc như sau:

- Username: Username chỉ được gồm chữ cái hoa và thường, chữ số và dấu gạch dưới (\_). Username có tối thiểu 4 ký tự và tối đa 12 ký tự và phải bắt đầu bằng chữ cái.
- Password: Password không được chứa username, gồm tối thiểu 6 ký tự và phải chứa cả chữ cái và số.
- Confirm password: Phải có giá trị giống như field password.

Sau khi đã nhập đủ thông tin, người dùng có thể ấn nút "Đăng ký". Có hai trường hợp có thể xảy ra:

- Các thông tin đều thỏa mãn các ràng buộc và username hiện tại chưa được sử dụng. Khi đó, hệ thống sẽ hiển thị thông báo đăng ký thành công cho người dùng và yêu cầu người dùng đăng nhập.
- Có một hoặc nhiều thông tin chưa thỏa mãn ràng buộc hoặc username hiện tại đã được sử dụng. Khi đó, hệ thống hiển thị lỗi sai tương ứng cho người dùng và yêu cầu người dùng sửa lại cho đúng (vẫn giữ nguyên giá trị các field của form như trước đó).

Ngoài ra, ở phần đăng ký của ứng dụng có thêm một link để dẫn người dùng đến trang đăng nhập nếu người dùng đã có tài khoản và muốn đăng nhập tài khoản đó.

### 2.2.2 Đăng nhập

Chức năng đăng nhập cho phép người dùng đã có tài khoản đăng nhập vào và sử dụng các tính năng của hệ thống. Ứng dụng hiển thị một form gồm 2 field: Username (tên người dùng) và password (mật khẩu). Sau khi nhập đủ hai thông tin trên, người dùng có thể ấn nút "Đăng nhập" để đăng nhập vào hệ thống. Khi đó, có hai trường hợp có thể xảy ra:

- Username đã được đăng ký và password tương ứng với username đó chính xác: Hệ thống cho phép người dùng vào hệ thống và dẫn người dùng đến trang chủ. Đồng thời, trạng thái của người dùng mới đăng nhập được hiển thị thành "online" và bạn bè của người dùng có thể nhìn thấy được trạng thái này.
- Username không tồn tại hoặc có tồn tại những password tương ứng không chính xác: Hệ thống sẽ báo lỗi "Thông tin đăng nhập chưa chính xác" cho người dùng và yêu cầu người dùng nhập lại các thông tin (vẫn giữ nguyên giá trị của form như trước đó).

Ngoài ra, ở phần đăng nhập của ứng dụng có thêm một link dẫn người dùng đến trang đăng ký nếu người dùng muốn tạo tài khoản mới.

### 2.3 Danh sách bạn bè

Đối với mỗi người dùng, server có lưu danh sách những người dùng khác là bạn bè của người dùng đó. Từ danh sách này, người dùng có thể chọn nhấn vào một hoặc nhiều bạn bè và gửi yêu cầu bắt đầu cuộc trò chuyện.

Khi người dùng gửi yêu cầu chat, có hai trường hợp có thể xảy ra:

- Được chấp nhận: Khi phía người dùng được gửi yêu cầu chat nhận được thông báo yêu cầu được bắt đầu cuộc trò chuyện, người dùng đó có thể nhấn nút "Đồng ý" để chấp nhận yêu cầu, cuộc trò chuyện được bắt đầu và hai người có thể gửi tin nhắn cho nhau.
- Bị từ chối: Tương tự như trường hợp trên nhưng thay vì nhấn nút "Đồng ý", người dùng nhận thông báo có thể nhấn "Từ chối" để từ chối yêu cầu. Bên cạnh nút "Từ chối" cũng có một timer đếm ngược từ 10 giây. Trong khoảng thời gian này, nếu không có phản ứng gì từ người nhận thông báo, yêu cầu sẽ tự động bị hủy. Thông báo yêu cầu bị từ chối được gửi cho người gửi yêu cầu.

### 2.4 Gửi file

Trong phần chat giữa hai người dùng, ngoài ô nhập tin nhắn và nút "Gửi" còn có một input cho việc gửi file (hiện tại chỉ cho phép gửi file text .txt). Khi nhấn vào input này, cửa sổ File Explorer sẽ hiện lên và yêu cầu người dùng chọn một file cần gửi đi.

Sau khi file cần gửi đi được chọn và được xác nhận là thỏa mãn các yêu cầu về loại file và kích thước, tên và kích thước của file sẽ được hiển thị trên màn hình ứng dụng. Tiếp theo, người dùng có thể chọn một trong hai thao tác sau:

- Gửi file: Người dùng ấn vào nút "Gửi" tương tự như gửi đi một tin nhắn văn bản. File được chọn sẽ được gửi đến người nhận. Thông tin của file gồm tên và kích thước sẽ được hiển thị tương tự một tin nhắn văn bản. Khi file được kiểm tra là đã gửi thành công và người nhận đã nhận được, file đó sẽ được tự động tải về máy người nhận.
- Xóa file: Người dùng có thể ấn nút "Hủy" bên cạnh file đã được chọn và hiển thị trên màn hình để hủy gửi file. Khi đó, thông tin file sẽ biến mất, người dùng trở lại giao diện như khi gửi tin nhắn văn bản thông thường.

## 3 Các protocol của ứng dụng

### 3.1 Chat - Gửi tin nhắn

Từ danh sách bạn của người dùng, người dùng có thể chọn nhắn tin với những tài khoản online.

Để bắt đầu trò chuyện với người dùng khác, một request sẽ được gửi đến người dùng còn lại. Người nhận sẽ bấm 'OK' để chấp nhận tin nhắn và tạo ra một connection. Nếu người dùng chọn "Từ chối" hoặc không trả lời trong vòng 10s thì sẽ trả về 'Reject'. Nếu người dùng đang chat thì sẽ vào trạng thái bận và sẽ từ chối request chat với những người dùng khác.

#### P2P Chat Protocol

- Yêu cầu khởi tạo chat session: User sẽ gửi request đến peer, gói tin sẽ bao gồm các thông tin sau

```
{
  type: "Request",
  content: "S <userIP> <port>",    \\ Start Chat Session
}
```

- Đóng chat session: Một request sẽ được gửi đến peer còn lại về việc ngưng chat. Nếu phía peer bên kia còn đang gửi tin nhắn hoặc gửi file thì dữ liệu sẽ bị hủy và nhận được thông báo kết thúc cuộc trò chuyện

```
{
  type: "Request",
  content: 'E' \\ End Chat Session
}
```

- Từ chối yêu cầu chat:

```
{
  type: "Response",
  code: 13    \\ Decline Chat
}
```

- Xác nhận yêu cầu chat. Nếu người dùng xác nhận yêu cầu chat từ người dùng khác thì một thread sẽ được tạo ra để listen message được gửi đến và một thread để có thể để gửi tin nhắn

```
{
  type: "Response",
  code: 1    \\ Accepted
}
```

- Gửi 1 message

```
{
  type: 'M'
  time: <send-message-time>
  message: <msg>
}
```

- Yêu cầu chuyển file

```

{
    type: "Request"
    content : "FT <file-name>" \\ File Transfer
}

```

- Nếu người dùng nhận được một yêu cầu chuyển file thì sẽ được thông báo có file chuyển đến và người dùng có thể lựa chọn nhận file hay không.

```

{
    type: "Response",
    code: 0      \\ Success
}

{
    type: "Response",
    code: 14     \\ Decline File Transferring
}

```

- Bắt đầu chuyển file: Sau khi nhận được xác nhận chuyển file từ peer còn lại thì một thread sẽ được tạo ra để bắt đầu tiến trình gửi file đến bên phía peer còn lại. Đồng thời sẽ gửi đi 1 request đến bên kia là bắt đầu quá trình chuyển file. Bên phía peer bên kia lúc này cũng sẽ tạo ra 1 thread để có thể nhận các gói tin và tập hợp lại file.

```

{
    type : "Request"
    content: "SF"      \\ Start File Transferring
    fname:  <file-name>
    content-length: <file-size>
}

```

- Nội dung file cần chuyển . Lưu ý, một file lớn có thể được chia thành nhiều gói nhỏ để gửi đi. Sau khi nhận được file peer bên kia có thể check lại độ toàn vẹn của gói tin bằng cách so sánh độ dài của data với size.

```

{
    fname: <file-name>
    segment: <file-segment-id>,
    data: <segment-data>
    size: <segment-data-size>
}

```

- Kết thúc chuyển file

```

{
    type: "Request",
    content : "DF" \\ Done File Transferring
}

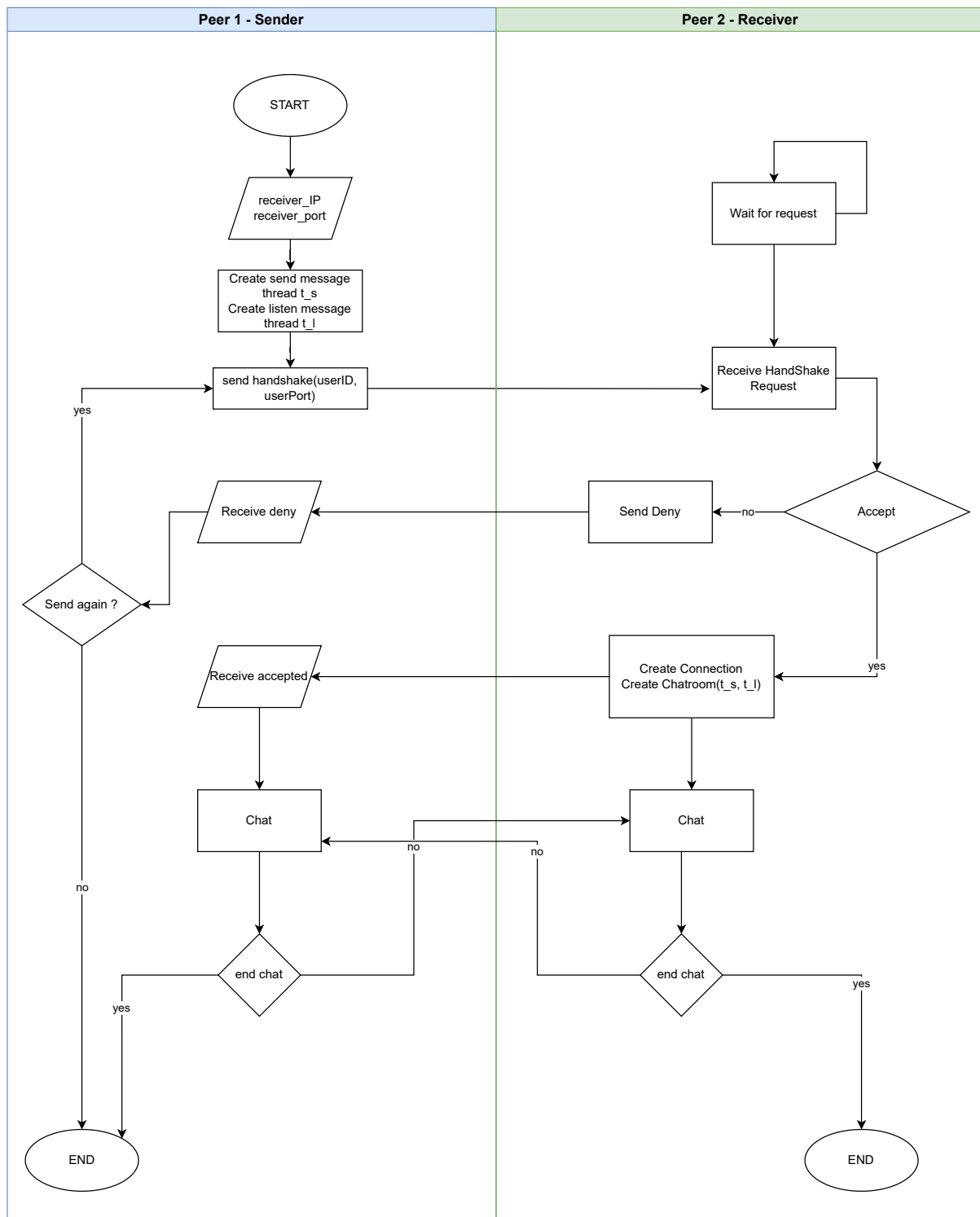
```

- Sau khi bên peer còn lại nhận được request kết thúc quá trình chuyển file thì sẽ tiến hành check độ toàn vẹn của file và gửi lại response



```
{
  type: "Response"
  code: 0      \\ File Transferring Success
}

{
  type: "Response"
  code: 70     \\ Communication error on send
  content:"Error: <Error-detail>"
}
```



Hình 1: Minh họa cho giao thức chat

### 3.2 Server communication - Giao tiếp máy chủ

Server sẽ giữ lại cặp thông tin gồm port, username của các người dùng. Khi người dùng gửi một yêu cầu đến server thì server sẽ nhận username của người nhận, trả lại port cho phía người bắt đầu. Đồng thời ứng với từng loại yêu cầu thì server sẽ phản hồi thêm theo những cách khác nhau như sau

- **Bắt đầu cuộc trò chuyện riêng tư với một người khác:** Sẽ gửi thông báo cho người kia rằng có muốn chấp nhận cuộc trò chuyện hay không. Nếu người kia đồng ý thì sẽ nhận được port từ người gửi và từ đây hai client sẽ tự giao tiếp với nhau. Nếu người kia không đồng ý thì server sẽ gửi thông báo từ chối trò chuyện về phía người gửi
- **Xem danh sách bạn bè:** Người dùng có thể xem danh sách bạn bè do server gửi thông tin về, đồng thời khi người dùng nhấn tin cho tất cả bạn bè cùng lúc qua tính năng Chat chung thì server sẽ đảm nhận request gửi tin nhắn của người dùng và trả response cho bạn bè là tin nhắn của người dùng đó
- **Tính năng file transfer:** Nhóm sử dụng **Kiến trúc Napster** để xử lý truyền tải file. Mạng lưới lúc này sẽ bao gồm các peer - người dùng và một central server - xử lý phát hiện người dùng và tìm kiếm nội dung. Tất cả giao tiếp các peer hay peer-server đều sử dụng TCP.  
Từ đây, nhóm sẽ đề cập đến tương tác peer-to-server trong tính năng gửi file này.  
**Peer-to-server:** Khác với kiến trúc P2P truyền thống (các peer tự dò tìm lẫn nhau), kiến trúc này sử dụng một hay một nhóm các server là nơi chứa danh sách của các peer và địa chỉ thư mục. Ngoài ra, server còn chứa các thông tin như metadata tình trạng các peer đang kết nối. Ở đây để đơn giản hóa, nhóm chỉ sử dụng duy nhất một server làm trung tâm.

### 3.3 File transfer - Gửi file

Giao thức nhóm sử dụng cho truyền gửi file về cơ bản tương tự như Chat, với những điểm giống nhau như sau:

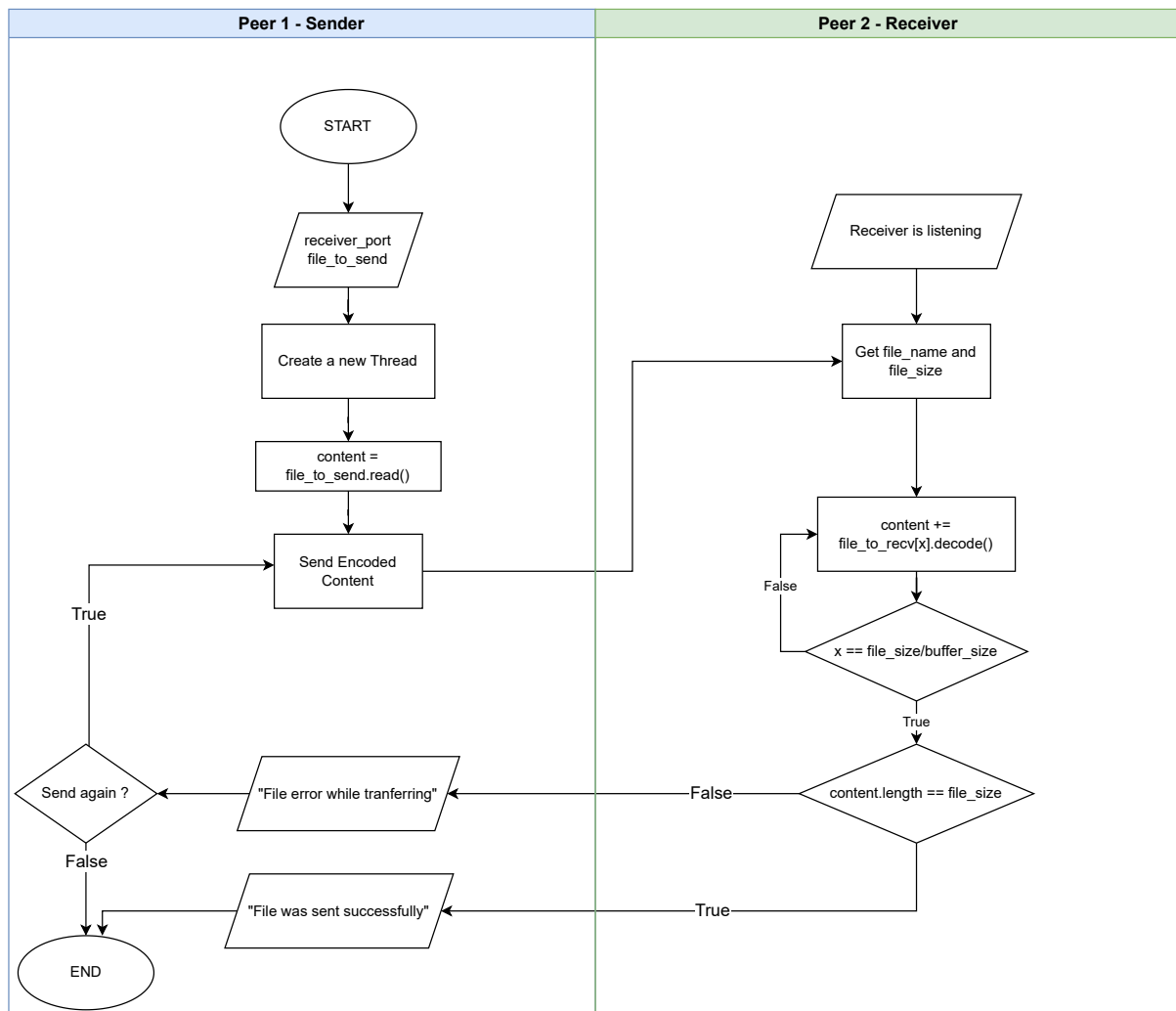
- **Chat và File Transfer đều sử dụng chung giao thức kết nối TCP để kết nối các peer với nhau.** Nếu một trong hai hình thức gửi (tin nhắn hoặc file) đã được khởi tạo từ trước và chưa đóng, peer sẽ tiếp tục gửi thông tin qua port của của peer còn lại mà không cần phải gửi request lên server để phân giải port.
- **Chat và File Transfer đều có cùng cơ chế gửi thông điệp.** Tin sẽ được gửi theo từng gói (với số lượng byte gửi buffer\_size được định nghĩa từ trước) từ người gửi theo một cơ chế mã hóa đến người nhận (phải giải mã trước khi sử dụng).

Tuy nhiên, vì một số đặc thù về bản chất của file và tin nhắn nên nhóm phải phân tách thành hai hình thức liên lạc khác nhau, cụ thể như sau:

- **File là một container chứa văn bản theo quy tắc, còn tin nhắn bản thân đã là văn bản.** Do đó định dạng của file có thể khá đa dạng (ví dụ như .pdf, .docx, .csv) và cần bộ quy tắc chuyên biệt cho từng loại file trước khi gửi để đảm bảo dữ liệu truyền qua được bảo toàn nội dung. Trong khi đó, message chỉ là một chuỗi bit có thể dễ dàng mã hóa.
- **File là một tập chứa văn bản, do đó kích thước của chúng thường lớn hơn tin nhắn.** Nếu như kích thước tin nhắn thường dao động trong khoảng vài byte, dung lượng file thường rất lớn (từ vài MB đến tận GB). Vì vậy file thường được chia nhỏ thành các mini-batch khi gửi. Ngoài ra, do việc gửi file sẽ có thể mất nhiều thời gian, nhóm đề xuất khởi tạo và duy trì 1 thread (luồng) riêng để xử lý việc gửi file, và các peer vẫn hoàn toàn có thể chat với nhau mà không phải chờ đợi.

Do đó, giao thức gửi file sẽ phải bổ sung một số chức năng từ giao thức gửi tin nhắn:

- Sau khi xác định được đối tượng cần gửi và tín hiệu gửi là file, phía **Người gửi (Sender)** cần phải xử lý thêm các bước sau
  1. Tạo một luồng mới để truyền dữ liệu (nếu đã có thì có thể bỏ qua).
  2. Trong luồng data đó, sử dụng thư viện hoặc các hàm có sẵn thích hợp để đọc được file với định dạng X. Nội dung đọc được sau đó sẽ được lưu trữ vào một biến.
  3. Tiến hành mã hóa (có nhiều cơ chế mã hóa, tuy nhiên phải đảm bảo có cả tên file và kích thước file trong nội dung đó) và gửi file đến người nhận. Lưu ý là người nhận phải online thì mới có thể gửi file qua.
- Mặt khác, phía **Người nhận (Receiver)** xử lý dữ liệu nhận được như sau:
  1. Đọc và giải mã tên file cùng kích thước để quản lý dữ liệu.
  2. Như nhóm đã trình bày ở trên, file không thể đến đồng thời mà sẽ theo các gói (batch/packet). Ở đây, sử dụng kích thước để kiểm soát số lượng gói và tổng kích thước, kèm với một số cơ chế (sẽ tìm hiểu sau) để kiểm soát chất lượng file nhận.
  3. Sau khi kiểm tra, phía người nhận phản hồi tình trạng file nhận. Nếu file nhận được xác nhận là toàn vẹn, người nhận sẽ trả về response "Gửi file thành công". Ngược lại, người nhận sẽ tiếp tục gửi response gửi lại file "Gửi file thất bại. Vui lòng gửi lại file."
  4. Ở trường hợp thất bại, người dùng có thể gửi lại file hoặc hủy gửi. Các bước được lặp lại tuần tự đến khi người dùng hủy gửi hoặc người nhận đã nhận toàn vẹn tập tin, sau đó data thread sẽ đóng lại.



Hình 2: Mô tả giao thức gửi file, với điều kiện kết nối các peer đã có

## 4 Mô tả các hàm hiện thực







Trong phần này, nhóm tiến hành mô tả các hàm chính đã thực hiện trong mã nguồn, bao gồm các đặc điểm về Input, Output, Chức năng và Giải thuật.

### 4.1 Các hàm hiện thực Server

Như đã mô tả, các hàm ở phía Server chủ yếu sẽ xử lý việc lưu trữ thông tin và kiểm tra tình trạng người dùng định kì.

Để hỗ trợ quản lý tài khoản, nhóm tiến hành tạo cơ sở dữ liệu lưu trữ thông tin người dùng, bao gồm các trường dữ liệu sau:

- ID: Mã người dùng
- Username: Tên tài khoản
- Password: Mật khẩu
- IP: địa chỉ IP của lần cuối cùng online hoặc của hiện tại
- Port: cổng mạng của lần cuối cùng online hoặc của hiện tại
- Status: Tình trạng người dùng (0 = Offline, 1 = Online)
- Ngoại trừ ID, Username, Password, các cột còn lại có giá trị động và sẽ thay đổi sau mỗi lần đăng nhập.

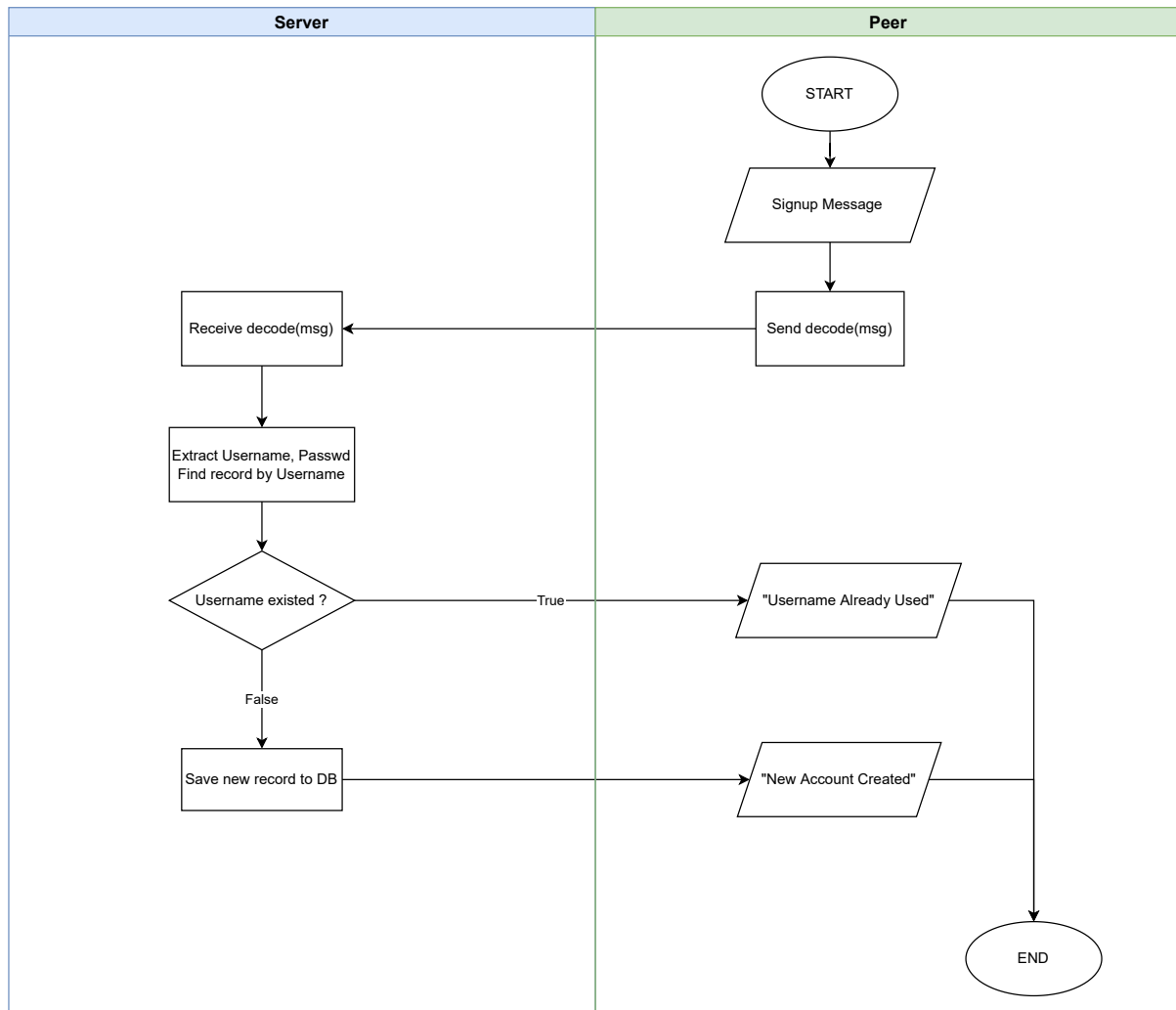
 id	INTEGER	"id" INTEGER UNIQUE
 username	TEXT	"username" TEXT NOT NULL UNIQUE
 password	TEXT	"password" TEXT NOT NULL
 ip	TEXT	"ip" TEXT
 port	TEXT	"port" TEXT
 status	INTEGER	"status" INTEGER

Hình 3: Mô tả Schema bảng User

1. **Quản lý tài khoản:** Ở phần này, nhóm chỉ tập trung vào các xử lý sự kiện ở phía Server, do đó các sự kiện do các service khác thực hiện sẽ được giản lược

- Đăng kí
  - *Input:* Thông tin đăng kí từ Peer gửi đến, bao gồm Username và Password
  - *Output:* Một thông điệp gửi về Peer báo cáo kết quả hành động
  - *Precondition:* Thông tin user nhập vào đã được kiểm lỗi cú pháp phía Client
  - *Chức năng:* Đăng kí một tài khoản mới vào Network và lưu trữ trong database
  - *Giải thuật*
    - (a) Phân giải thông điệp nhận được thành các tham số (username, password).

- (b) Tiến hành kiểm tra sự tồn tại của tên tài khoản, nếu có thì báo lỗi về người dùng, không thì tiếp tục.
- (c) Tiến hành thêm dữ liệu mới vào database. Lưu ý rằng ID sẽ được cấp phát tự động, status = 0 (offline), còn trường IP và port được để giá trị là NULL.
- (d) Thông báo kết quả đến người dùng và chuyển hướng đến quá trình login.

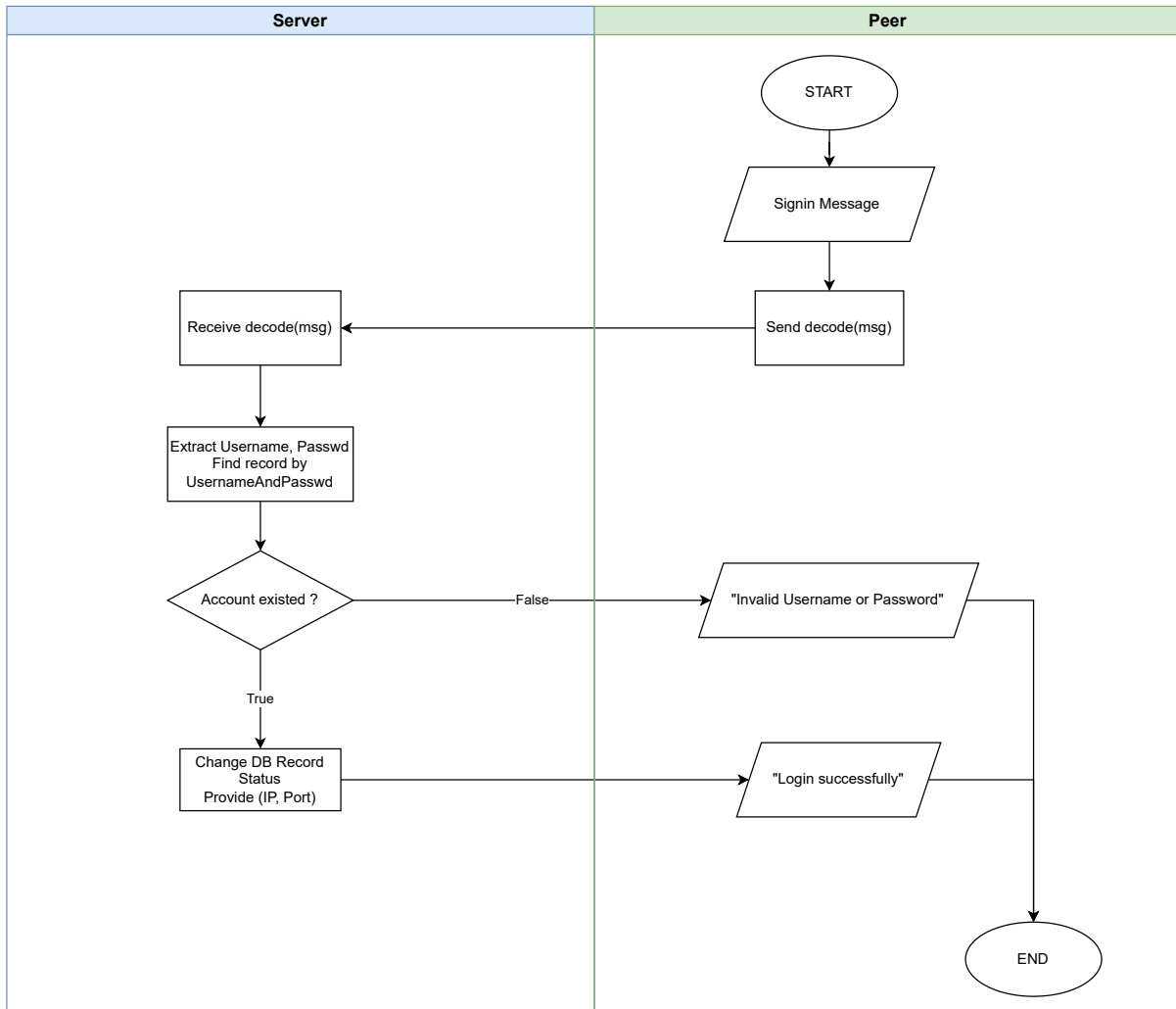


Hình 4: Flowchart Đăng kí

- Đăng nhập
  - *Input*: Thông tin đăng kí từ Peer gửi đến, bao gồm Username và Password
  - *Output*: Một thông điệp gửi về Peer báo cáo kết quả hành động
  - *Precondition*: Thông tin user nhập vào đã được kiểm lỗi cú pháp phía Client
  - *Chức năng*: Đăng nhập một tài khoản vào Network và cập nhật trạng thái trên database

– *Giải thuật*

- Phân giải thông điệp nhận được thành các tham số (username, password).
- Tiến hành kiểm tra sự tồn tại của tên tài khoản và mật khẩu trùng khớp, nếu không thì báo lỗi về người dùng, có thì tiếp tục.
- Tiến hành thay đổi dữ liệu trong database với username tương ứng. Các trường giá trị IP, Port được gán tự động (người dùng không cần nhập vào) và Status = 1 (online).
- Thông báo kết quả đến người dùng và chuyển hướng đến trang danh sách người dùng online.



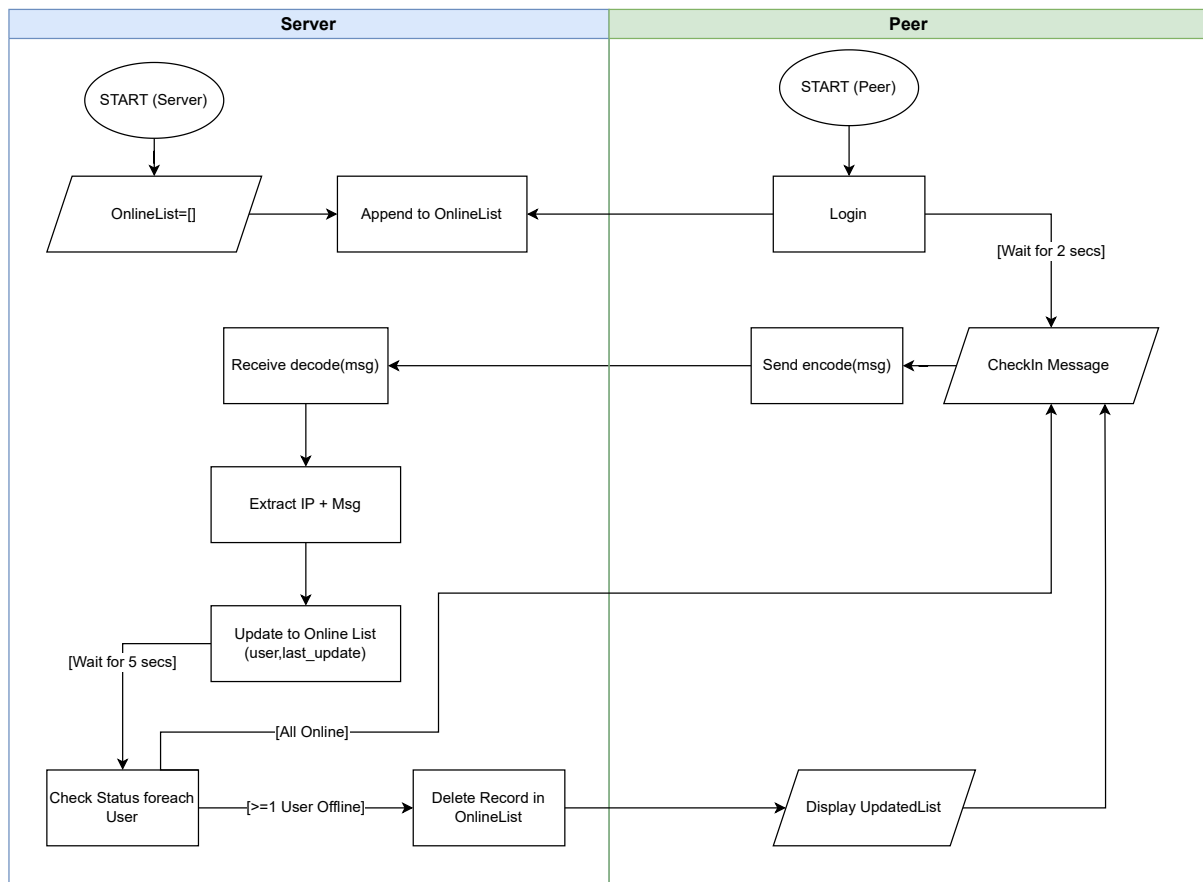
Hình 5: Flowchart Đăng nhập

## 2. Quản lý trạng thái:

- Input*: Thông điệp "điểm danh" từ mỗi Peer



- *Output*: Danh sách Online sau mỗi lần thực hiện "điểm danh"
- *Precondition*: User đã đăng nhập thành công vào Network
- *Chức năng*: Kiểm tra tình trạng Online của mỗi user và cập nhật trạng thái database
- *Giải thuật*
  - (a) Tạo một biến lưu trữ danh sách người dùng online. Biến này được khởi tạo ngay sau khi thiết lập Server. Mỗi khi có một người dùng đăng nhập vào hệ thống, danh sách người dùng tăng lên 1 đơn vị với định dạng mỗi record là (username, last\_checked\_time). Giá trị của tham số thứ 2 được khởi gán là thời điểm đăng nhập.
  - (b) Tiến hành điểm danh: Chờ phản hồi từ từng peer (trong danh sách Online) để nhận thông điệp (username, "Hello"). Server sẽ kiểm tra tình trạng Online sau một khoảng thời gian nhất định. Từ đây sẽ có 2 trường hợp xảy ra:
    - i. Nhận được phản hồi: Tiến hành cập nhật tham số last\_checked\_time của user tương ứng trong danh sách Online.
    - ii. Không nhận được phản hồi: Loại người dùng khỏi danh sách Online và tiến hành cập nhật database (status = 0). Nếu trường hợp này xảy ra, gửi danh sách vừa cập nhật đến các Peer còn Online.
  - (c) Bước trên được lặp lại tuần tự cho từng người dùng Online cho đến khi Server ngừng hoạt động.

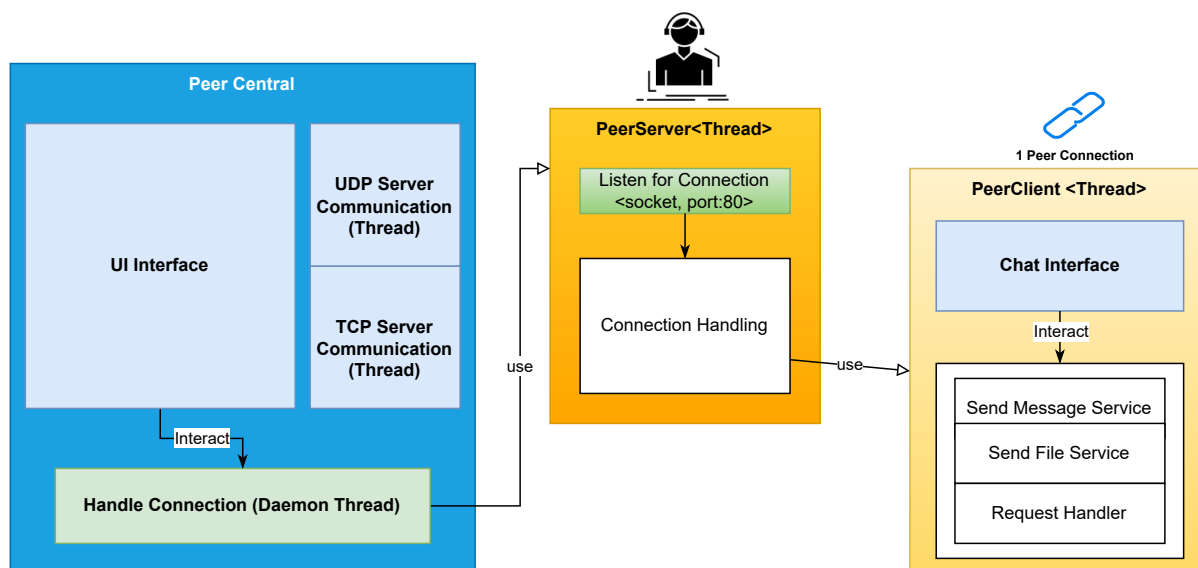


Hình 6: Flowchart Quản lí Trạng thái

## 4.2 Các hàm hiện thực Peer

Mỗi peer sẽ được cung cấp các phương thức để giao tiếp với 1 peer khác Các phương thức sẽ được định nghĩa bằng 1 lớp Encode (protocol.py)

### 4.2.1 Mô tả về kiến trúc của 1 Peer



Hình 7: Kiến trúc của 1 Peer

- PeerCentral: Người dùng sẽ thao tác với UI Interface để thực hiện các chức năng như đăng nhập, đăng kí, xem danh sách người dùng online, yêu cầu chat với 1 người dùng nào đó
  - UDP Server và TCP Server là thành phần dùng để giao tiếp với Server: Lấy danh sách người dùng online, đăng kí, đăng nhập, xác nhận trạng thái của Peer
  - Handle Connection là 1 daemon thread được tạo ra từ Lớp PeerServer và đóng vai trò: lắng nghe kết nối, nhận các yêu cầu từ người dùng và xử lí.
- PeerServer:
  - Là 1 thread và sẽ luôn listen connection ở Port mặc định là 80
  - Nếu có bất cứ Connection nào tới thì sẽ xử lí và tạo ra 1 PeerClient để xử lí riêng Connection này
  - Có vai trò quản lí tất cả các kết nối
- PeerClient
  - Là 1 thread Có vai trò handle một connection của 1 Peer Khác
  - Giao diện chat: gửi tin nhắn, gửi file, xử lí request

#### 4.2.2 Các hàm của Peer

- Yêu cầu chat với 1 Peer khác
  - *Input*: IP của Peer đối tượng, Port mặc định để kết nối là 80
  - *Output*: Nếu kết nối thành công, 1 cửa sổ chat sẽ xuất hiện
  - *Precondition*: Peer đã đăng nhập vào hệ thống và Peer đối tượng online
  - *Chức năng*: Chat với những Peer khác
  - *Giải thuật*
    1. Peer sẽ tạo ra một socket mới để kết nối với Peer còn lại, IP sẽ nhận từ đầu vào và mặc định sẽ luôn kết nối vào port 80
    2. Bên peer đối tượng sẽ luôn có 1 thread listen ở port 80 để nhận connection (HandleConnection). Khi có 1 yêu cầu kết nối thì ngay lập tức chấp nhận và tạo ra 1 Thread class PeerClient để xử lý connection này.
    3. nếu bên đối tượng chấp nhận thì bên Peer gửi request sẽ hiện lên 1 cửa sổ chat , nếu không sẽ thông báo bị từ chối và đóng connection.
- Đóng kết nối với 1 Peer đang chat
  - *Input*: Yêu cầu đóng đoạn chat của người dùng
  - *Output*: Đóng đoạn chat
  - *Precondition*: Người dùng đang thực hiện chat với 1 người dùng khác
  - *Chức năng*: Ngắt kết nối
  - *Giải thuật*
    1. Người dùng chọn tắt đoạn chat
    2. Một tin nhắn thông báo ngừng chat được gửi đến bên kia
    3. Đóng kết nối
    4. Bên Peer chat sau khi nhận được gói tin thông báo cũng sẽ đóng kết nối
- Gửi File đến 1 Peer đang chat
  - *Input*: Đường dẫn đến file cần gửi
  - *Output*: File được gửi đi
  - *Precondition*: Người dùng đang thực hiện chat với 1 người dùng khác
  - *Chức năng*: Gửi file
  - *Giải thuật*
    1. Người dùng chọn file gửi và gửi file
    2. Thông báo file được gửi sẽ hiện lên trong đoạn chat
    3. Bên người nhận sẽ pop up lên cửa sổ chọn vị trí để lưu file
    4. Sau khi chọn được vị trí thì quá trình truyền file sẽ bắt đầu
    5. Nếu file có kích thước thì sẽ được chia nhỏ ra thành các gói để gửi đi
    6. Bên phía người nhận sau khi nhận được file sẽ thực hiện ensemble lại
    7. Nếu việc truyền file thành công thì người nhận sẽ gửi đi 1 tin nhắn thông báo việc gửi file thành công

## 4.3 Các hàm hỗ trợ đồ họa

### 4.3.1 Thư viện Tkinter

Tkinter là một thư viện ngôn ngữ Python hỗ trợ lập trình giao diện đồ họa người dùng (GUI). Tkinter thuộc về Tk - bộ công cụ cung cấp các phương thức hỗ trợ lập trình GUI bằng nhiều ngôn ngữ lập trình khác nhau.

Trong ứng dụng chat này, nhóm sử dụng một số class được Tkinter hỗ trợ để hiện thực một số thành phần:

- **Frame:** Là đơn vị layout cơ bản của một ứng dụng Tkinter, được dùng để chứa các thành phần khác của GUI trên một cửa sổ mới.
- **Button:** Là đơn vị thể hiện một nút ấn (button) trên GUI. Một button thường được liên kết với một hàm và hàm này được gọi khi người dùng nhấn nút.
- **Label:** Là đơn vị thể hiện một đoạn chữ hoặc hình ảnh lên GUI.
- **Scrollbar:** Là đơn vị hiện thực thanh scrollbar cho phép người dùng cuộn để xem nội dung kích thước lớn không thể được hiển thị cùng lúc trên màn hình.
- **Entry:** Là đơn vị hiển thị một textbox input, cho phép người dùng nhập thông tin dưới dạng text. Thông tin này sau đó có thể được dùng để hiện thực các login liên quan hoặc gửi đến các đơn vị GUI khác.

Ngoài ra, nhóm còn sử dụng một số phương thức và tính năng khác của Tkinter.

### 4.3.2 Kiến trúc giao diện người dùng (UI) của ứng dụng

- **FirstPage:** Là cửa sổ hiển thị mặc định khi ứng dụng được mở. Trang này sẽ hiển thị logo của ứng dụng và nút điều hướng để chuyển sang trang đăng nhập và trang đăng ký.
- **RegistryFrame:** Là frame hiển thị trang đăng nhập và trang đăng ký. Khi nút "Login" hoặc "Register" trên FirstPage được nhấn, RegistryFrame sẽ được render trên cửa sổ FirstPage tương ứng với nút nào được nhấn.
- **ListPage:** Là cửa sổ hiển thị danh sách người dùng có kết bạn với người dùng đã đăng nhập. Cửa sổ sẽ hiển thị các thông tin bao gồm username, trạng thái online của mỗi người dùng và nút nhấn "Message" để gửi yêu cầu nhắn tin đến một người dùng trong danh sách.
- **PeerClient:** Hiển thị cửa sổ chat giữa người dùng đã đăng nhập với một người dùng nào đó sau khi người đối phương nhận được và chấp nhận yêu cầu nhắn tin của người dùng. Trang này sẽ hiển thị nội dung và thời điểm các tin nhắn và file đã gửi và nhận, cũng như textbox dùng để nhập tin nhắn, nút "Send" để gửi tin nhắn và nút "File" để gửi file.

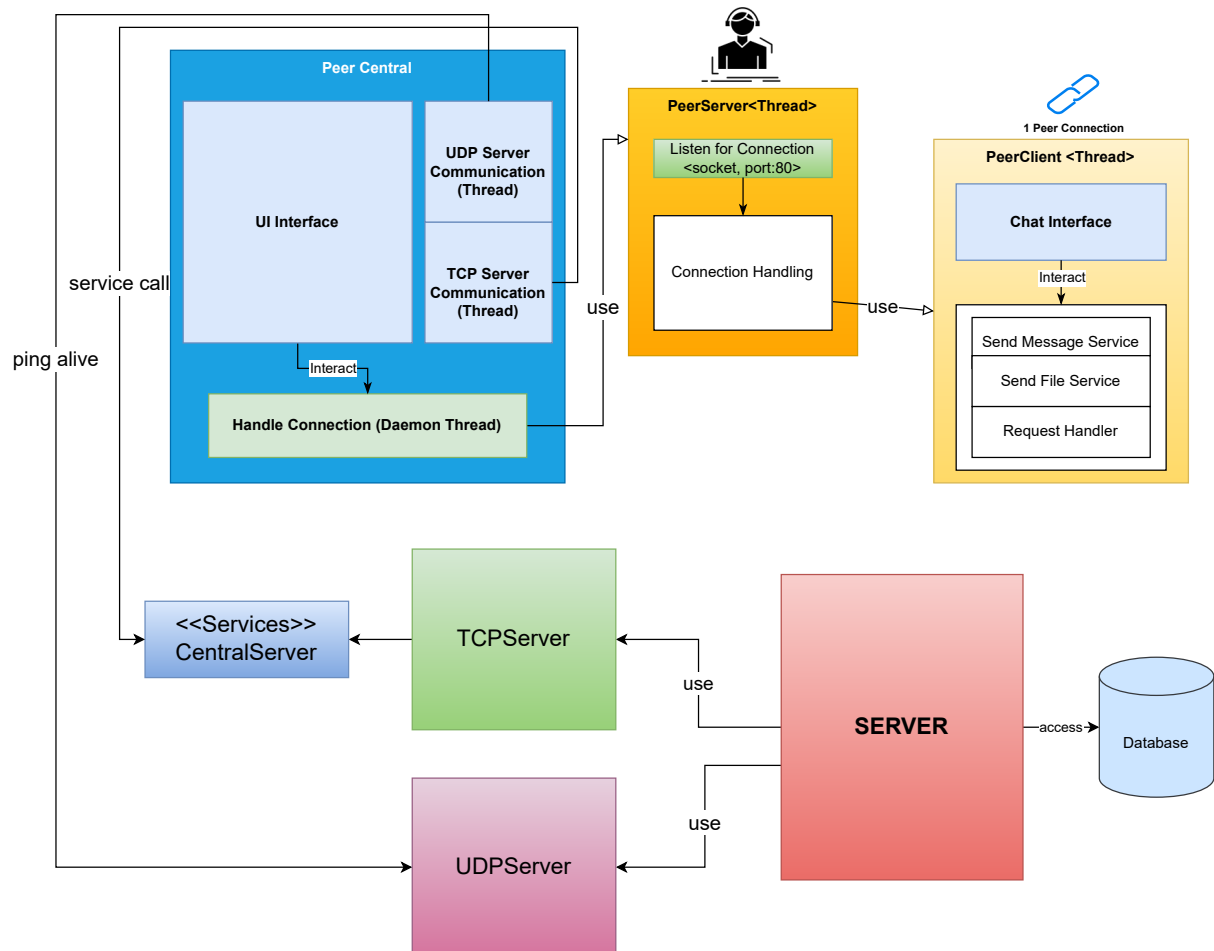
### 4.3.3 Các hàm hiện thực UI

- **Đăng nhập:**
  - **Input:** Username (tên người dùng) và password (mật khẩu) do người dùng nhập.
  - **Output:** Xuất hiện màn hình thông báo lỗi nếu có lỗi trong việc nhập liệu (nhập thiếu username hoặc password) và chuyển hướng sang phần logic tiếp theo nếu thông tin là hợp lệ.
  - **Precondition:** Form login đã được hiển thị trên màn hình ứng dụng.

- Chức năng: Kiểm tra thông tin đăng nhập do người dùng nhập có hợp lệ hay không.
  - Giải thuật: Placeholder mặc định của hai textbox input là "Username" và "Password". Nếu có ít nhất một trong hai trường input có giá trị là trống hoặc placeholder tương ứng, cửa sổ thông báo lỗi sẽ được hiển thị, yêu cầu người dùng nhập lại thông tin. Nếu thông tin người dùng nhập là đầy đủ và hợp lệ, ứng dụng chuyển sang phần logic tiếp theo, bao gồm kiểm tra username và password trong database.
- Đăng ký:
    - Input: Full name (tên đầy đủ), username (tên người dùng), password (mật khẩu), confirm password (mật khẩu nhập lại) do người dùng nhập.
    - Output: Xuất hiện màn hình thông báo lỗi nếu có lỗi trong việc nhập liệu (nhập thiếu thông tin, mật khẩu không đủ mạnh hay hai mật khẩu không trùng khớp) và chuyển hướng sang phần logic tiếp theo nếu thông tin là hợp lệ.
    - Precondition: Form register đã được hiển thị trên màn hình ứng dụng.
    - Chức năng: Kiểm tra thông tin đăng ký do người dùng nhập có hợp lệ hay không.
    - Giải thuật: Placeholder mặc định của các textbox input là "Full name", "Username" và "Password". Nếu có ít nhất một trong các trường input có giá trị là trống hoặc placeholder tương ứng, cửa sổ thông báo lỗi sẽ được hiển thị, yêu cầu người dùng nhập lại thông tin. Nếu thông tin người dùng nhập là đầy đủ và hợp lệ, ứng dụng chuyển sang phần logic tiếp theo, bao gồm kiểm tra xem username đã tồn tại trong database hay chưa.

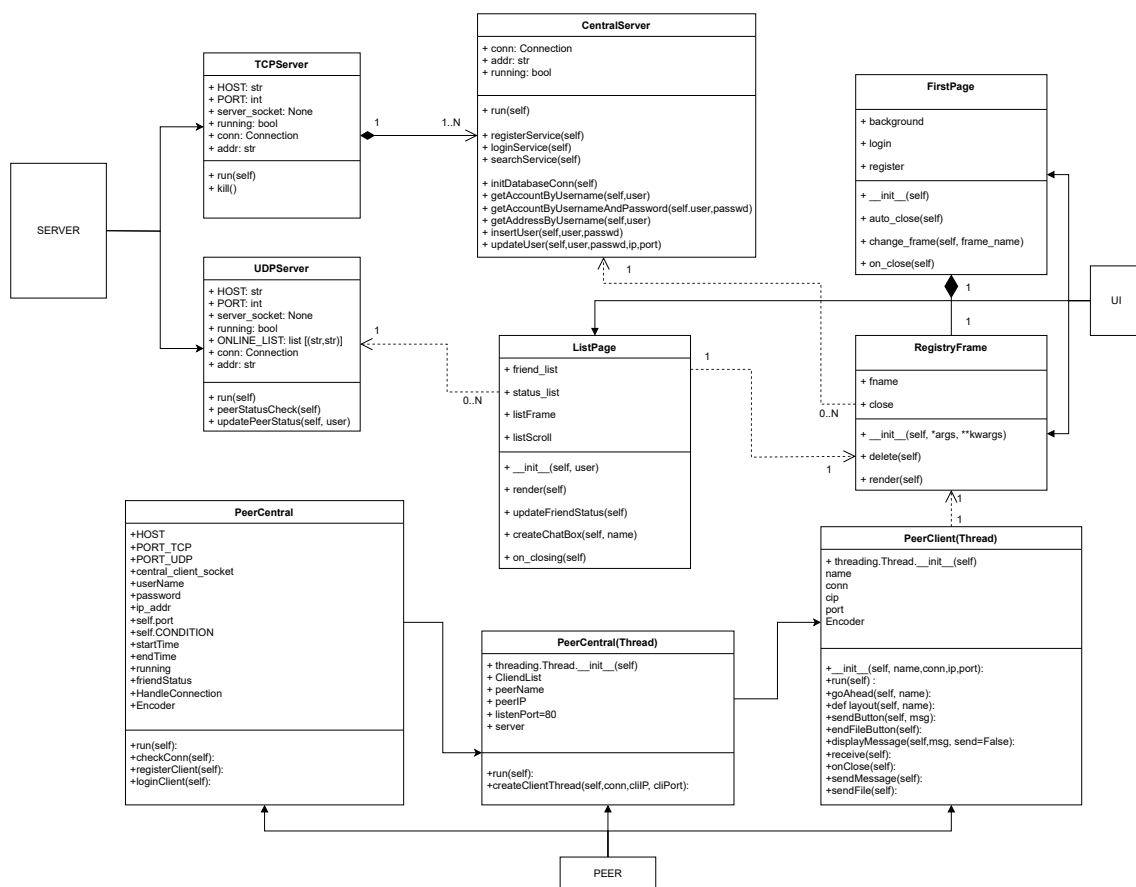
## 5 Thiết kế ứng dụng

### 5.1 Kiến trúc sử dụng



Hình 8: Thiết kế kiến trúc tổng quát

## 5.2 Bản vẽ sơ đồ lớp



Hình 9: Bản vẽ sơ đồ lớp

## 5.3 Mô tả các lớp chính

### 5.3.1 Các lớp dành cho Server

Tất cả mã nguồn về các lớp phía Server được trình bày trong file *server.py*. Để hiện thực ChatApp của nhóm, nhóm chia Server thành 3 lớp chính:

#### 1. *TCPServer*

- Mô tả: Một luồng kết nối từ phía Server, dùng để tạo luồng kết nối chính đến với các Peer trong thao tác Chat + Gửi File và sử dụng giao thức kết nối tại Transport Layer là TCP.
- Các thuộc tính:
  - HOST: Địa chỉ IP của TCP Server
  - PORT: Port của TCP Server
  - server\_socket: nơi kết nối Server lên Network với địa chỉ (HOST, PORT)
  - running: Tình trạng kết nối của luồng



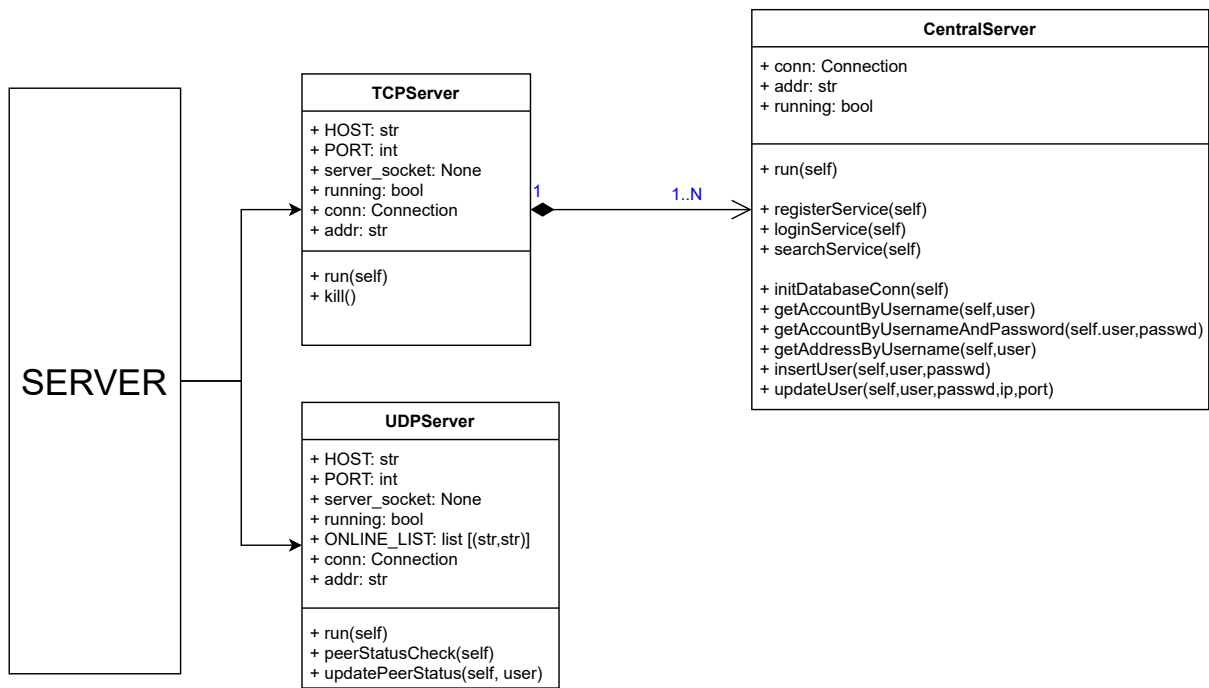
- conn: lưu trữ kết nối từ Peer đến TCPServer
- addr: địa chỉ kết nối của Peer
- Các phương thức:
  - run(): phương thức chính dùng để kích hoạt vận hành TCPServer, bao gồm các tính năng sau đây:
    - \* Đẩy kết nối với (HOST, PORT) cho trước lên network (sử dụng thư viện socket).
    - \* Lắng nghe kết nối của các Peer gửi đến kèm với địa chỉ của chúng, sau đó thêm vào danh sách kết nối.
    - \* Tạo một thread mới kiểu CentralServer và chuyển thông tin kết nối từ Peer sang đây.
  - kill(): kết thúc luồng thực thi chính.

## 2. UDPServer

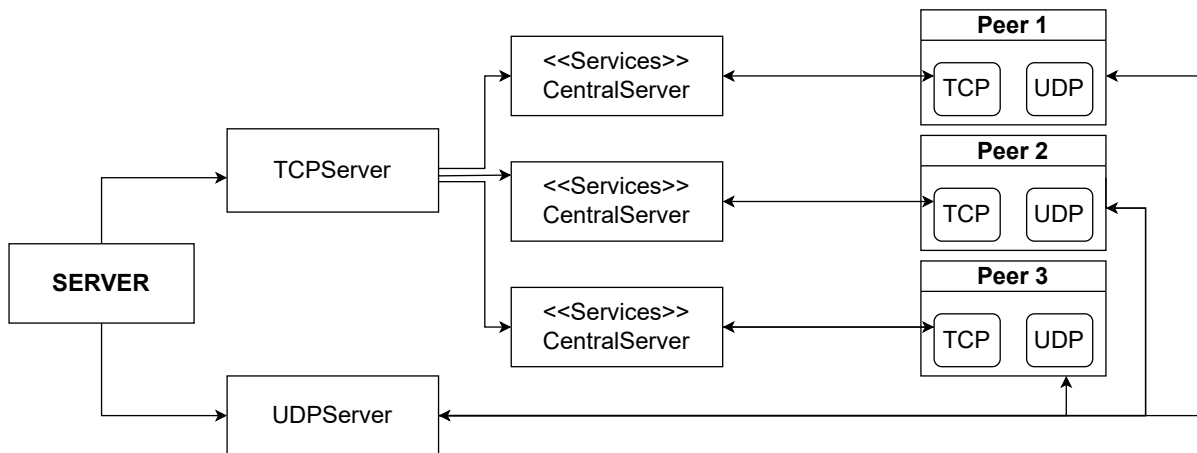
- Mô tả: Một luồng kết nối từ phía Server, dùng để tạo luồng kết nối chính đến với các Peer trong thao tác Checkin (điểm danh) và sử dụng giao thức kết nối tại Transport Layer là UDP.
- Các thuộc tính:
  - ONLINE\_LIST: nơi chứa danh sách các Peer đang Online, mỗi record bao gồm tên tài khoản và thời gian gửi kết nối gần nhất.
  - Các thuộc tính còn lại có mô tả tương tự Lớp TCPServer, tuy nhiên Port phía TCPServer phải đảm bảo phải khác với UDPServer khi khởi tạo.
- Các phương thức:
  - run(): phương thức chính dùng để kích hoạt vận hành TCPServer, bao gồm các tính năng
    - \* Đẩy kết nối với (HOST, PORT) cho trước lên network (sử dụng thư viện socket).
    - \* Tạo một thread để listen tin nhắn Checkin của từng peer, với target là phương thức peerStatusCheck().
    - \* Lắng nghe tin nhắn từ các peer và trả về danh sách online mới nhất.
  - peerStatusCheck(): phương thức kiểm tra trạng thái của người dùng sau một quãng thời gian, từ đó cập nhật danh sách online (sửa thời gian cập nhật nếu còn online, xóa khỏi danh sách nếu offline)
  - updatePeerStatus(): phương thức can thiệp để chỉnh sửa trạng thái trên database (từ Online sang Offline).

## 3. CentralServer

- Mô tả: Lớp tương tác trực tiếp với Peer và cung cấp các dịch vụ từ Server đến với Peer. Chỉ có một TCPServer nhưng lại có N CentralServer khác nhau, ứng với N Peer đang yêu cầu dịch vụ.
- Các thuộc tính: Tương tự như TCP Server, tuy nhiên sẽ không có địa chỉ IP và PORT vì các kết nối này dựa trên địa chỉ của TCP Server cung cấp.
- Các phương thức:
  - run(): phương thức chính dùng để kích hoạt vận hành CentralServer và cung cấp các dịch vụ (service) cho người dùng
  - Các hàm liên quan đến dịch vụ: bao gồm đăng kí, đăng nhập và tìm kiếm. Các thao tác này tiến hành thao tác với Database và trả kết quả về Peer đang kết nối
  - Các hàm liên quan đến thao tác Database: chủ yếu là chứa các câu lệnh dựa vào các tham số truyền vào và phục vụ cho các dịch vụ CentralServer cung cấp.



Hình 10: Thiết kế lớp phía Server



Hình 11: Kiến trúc phía Server, với sự giản lược phía Peer

### 5.3.2 Các lớp dành cho Peer

Tất cả mã nguồn về các lớp phía Peer được trình bày trong file *peer.py*. Để hiện thực ChatApp của nhóm, nhóm chia Peer thành 3 lớp chính:

## 1. *PeerCentral*

- Mô tả: Đây là lớp tương tác trực tiếp với Server, dùng để quản lý các sự kiện bên phía người dùng như đăng nhập, đăng ký và gửi yêu cầu nhắn tin đến các Peer khác
- Các thuộc tính:
  - HOST: Địa chỉ IP của bên phía Server
  - HOST\_TCP: Port mà phía TCP Server sử dụng, ở đây cụ thể nhóm sử dụng port 3000
  - HOST\_UDP: Port mà phía UDP Server sử dụng, ở đây cụ thể nhóm sử dụng port 3004
  - central\_client\_server: Socket dùng để tạo kết nối với TCP Server qua địa chỉ (HOST, HOST\_TCP)
  - username: Tên đăng nhập của người dùng
  - password: Mật khẩu của người dùng
  - ip\_addr: Địa chỉ IP bên phía người dùng
  - port: Giá trị port bên phía người dùng
  - CONDITION: Biến boolean để phía PeerCentral kiểm tra xem người dùng đã đăng nhập vào hệ thống chưa
  - startTime: Thời điểm người dùng đăng nhập thành công vào hệ thống
  - endTime: Thời điểm người dùng thoát khỏi hệ thống
  - running: Biến boolean được tạo để đảm bảo phần đăng ký/ đăng nhập của hệ thống có chạy hay không
  - friendStatus: Danh sách bạn bè bên phía người dùng, được cập nhật liên tục mỗi 3s để kiểm tra xem có bạn bè nào đang online không
  - handleConnection: Biến khởi tạo PeerServer để tiếp nhận việc kết nối khi bắt đầu chat với người dùng khác
  - Encoder: Đối tượng thuộc lớp Encoder bên file *protocol.py*, trong đó lưu giữ các thuộc tính là ip và port của người dùng, dùng khi gửi yêu cầu chat đến người dùng khác
- Các phương thức:
  - run(): Phương thức chính dùng để kích hoạt vận hành PeerCentral, trong đó khởi tạo phần đăng nhập, đăng ký và một thread quản lý việc người dùng có online hay không và danh sách bạn bè đang online
  - checkConn(): Phương thức dùng để kiểm tra người dùng có đang online hay không và danh sách bạn bè đang online của người dùng, bằng cách cứ mỗi 3s từ lúc người dùng đăng nhập thành công vào hệ thống thì sẽ gửi tin nhắn đến UDP Server để kiểm tra tình trạng và cập nhật danh sách bạn bè online mới nhất (sẽ có in ra danh sách bạn bè đang online mỗi khi có cập nhật mới)
  - registerClient(): Phương thức dùng để khởi tạo sự kiện tạo tài khoản bên phía người dùng
  - loginClient(): Phương thức dùng để khởi tạo sự kiện đăng nhập bên phía người dùng, Khi đăng nhập thành công thì phương thức này sẽ tiếp nhận sự kiện tìm người dùng để khởi tạo cuộc trò chuyện mới

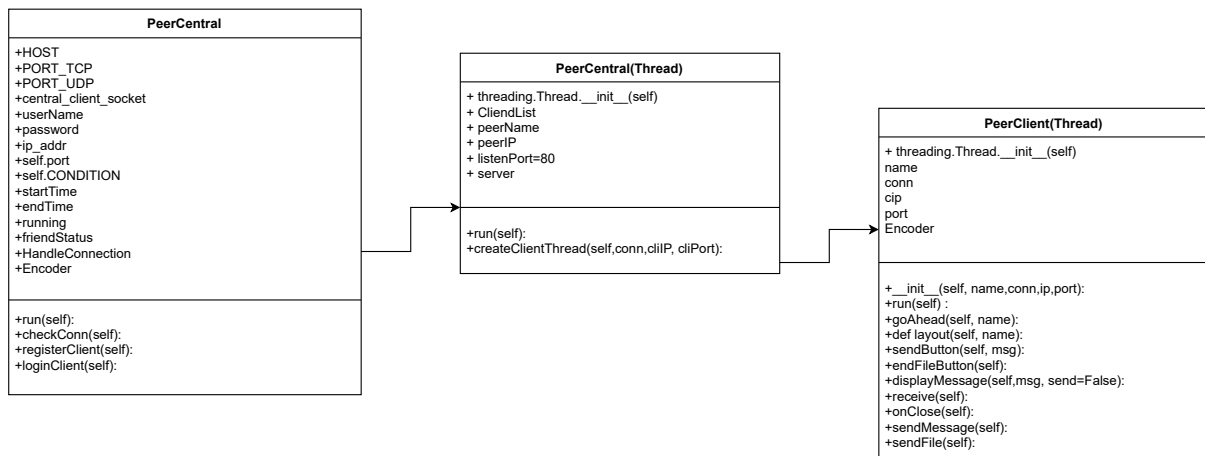
## 2. *PeerServer*

- Mô tả: Đây là lớp bên phía Peer dùng để tương tác với Server sau khi người dùng đã đăng nhập vào hệ thống, và lớp này sẽ quản lý các kết nối của người dùng với những người dùng khác
- Các thuộc tính:

- CliendList: Mảng lưu giữ danh sách các kết nối mà người dùng đang nắm giữ
- peerName: Biến lưu trữ username của người dùng
- peerIP: Biến lưu trữ địa chỉ IP của người dùng
- listenPort: Giá trị port của lớp PeerServer khi kết nối với Server
- server: Kết nối giữa lớp PeerServer với Server
- Các phương thức:

### 3. *PeerClient*

- Mô tả: Đây là lớp để khởi tạo UI khi bắt đầu cuộc trò chuyện với một người dùng khác, trong đó bao gồm các sự kiện gửi tin nhắn và chuyển file. Lưu ý mỗi cuộc trò chuyện với những người khác nhau sẽ là các cửa sổ UI riêng biệt
- Các thuộc tính:
  - name: Tên hiển thị của người dùng bên UI
  - conn: Biến tiếp nhận kết nối của hai người dùng với nhau, dùng khi muốn gửi yêu cầu đến người dùng kia
  - cip: Địa chỉ IP của người dùng
  - port: Giá trị port của người dùng
  - Encoder: Đối tượng thuộc lớp Encoder bên file *protocol.py*, trong đó lưu giữ các thuộc tính là ip và port của người dùng, dùng khi gửi các yêu cầu người dùng kia
- Các phương thức:
  - `__init__(self, name, conn, ip, port)`: Khởi tạo các thuộc tính của lớp PeerClient
  - `run()`: Phương thức dùng để khởi tạo cửa sổ chat
  - `goAhead()`: Phương thức để khởi tạo một thread dùng để nhận tin nhắn từ phía người dùng kia
  - `layout()`: Phương thức dùng để thiết kế các thành phần nằm bên trong cửa sổ chat, bao gồm các thành phần như tên người dùng bên kia, khung chat, hộp để người dùng gõ tin nhắn, các nút gửi tin nhắn và gửi file
  - `sendButton()`: Phương thức để đưa đến hoạt động tiếp nhận sự kiện người dùng gửi tin nhắn đến người dùng bên kia
  - `sendFileButton()`: Phương thức để đưa đến hoạt động tiếp nhận sự kiện người dùng gửi file đến người dùng bên kia
  - `displayMessage()`: Phương thức để hiển thị tin nhắn người dùng vừa gửi lên khung chat, đồng thời hiển thị tin nhắn mà người dùng kia gửi đến
  - `receive()`: Phương thức để tiếp nhận các sự kiện bên phía người dùng kia gửi đến, bao gồm tiếp nhận yêu cầu khởi tạo cuộc trò chuyện, thông báo đến người dùng khi người kia offline thì cuộc trò chuyện này sẽ bị đóng sau 2s, tiếp nhận tin nhắn được gửi từ phía người dùng kia, và cả file mà người dùng kia gửi đến
  - `onClose()`: Phương thức tiếp nhận sự kiện người dùng đóng cửa sổ chat, khi đó người dùng còn lại sẽ được thông báo về việc này
  - `sendMessage()`: Phương thức tiếp nhận sự kiện người dùng gửi tin nhắn đến người dùng kia, trong đó bao gồm việc hiển thị tin nhắn vừa gửi lên khung chat và gửi tin nhắn được mã hóa đến bên kia
  - `sendFile()`: Phương thức tiếp nhận sự kiện người dùng gửi file, bắt đầu từ việc để người dùng chọn file trong máy tính của mình, tiếp đến là gửi yêu cầu chuyển file và chuyển dữ liệu của file đó đến người dùng bên kia, sau đó sẽ hiển thị tin nhắn bên người dùng là đã gửi file thành công



Hình 12: Class Diagram Cho Peer

### 5.3.3 Các lớp dành cho UI

#### 1. FirstPage: Kế thừa class "Tk" của thư viện Tkinter

- Mô tả: Hiển thị cửa sổ mặc định khi ứng dụng được khởi chạy.
- Các thuộc tính:
  - background: Hiển thị logo của ứng dụng.
  - login: Một instance của RegistryFrame với tên frame là "login" cho trang đăng nhập.
  - register: Một instance của RegistryFrame với tên frame là "register" cho trang đăng ký.
- Các phương thức:
  - \_\_init\_\_(self): Constructor của class, xác định các thiết lập cơ bản cho cửa sổ ứng dụng như kích thước, tựa đề, màu nền, ... Ngoài ra, hàm này còn hiện thực các nút login, register và chạy phương thức "mainloop", làm cho cửa sổ này hiển thị trên màn hình.
  - auto\_close(self): Tự tắt cửa sổ login sau 1 giây nếu đăng nhập thành công.
  - change\_frame(self, frame\_name): Chuyển từ trang login sang register và ngược lại khi người dùng nhấn nút tương ứng (khi đó frame\_name sẽ được truyền vào giá trị tương ứng "login" hay "register").
  - on\_closing(self): Hiển thị cửa sổ mới xác nhận nếu người dùng muốn thoát ứng dụng và tiến hành đóng cửa sổ ứng dụng nếu người dùng xác nhận.

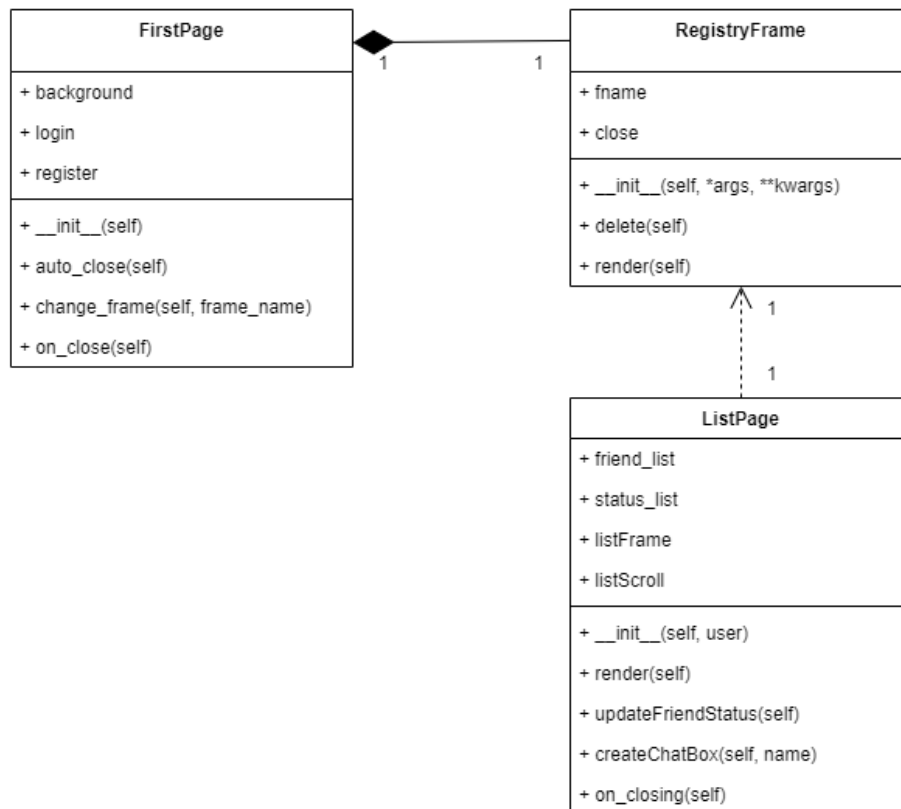
#### 2. RegistryFrame: Kế thừa class "Frame" của thư viện Tkinter

- Mô tả: Hiển thị trang đăng nhập / đăng ký trên cửa sổ ứng dụng khi người dùng ấn nút tương ứng.
- Các thuộc tính:
  - fname: Tên của frame, có hai giá trị là "login" và "register", giúp hiển thị trang đăng nhập hoặc đăng ký cho phù hợp.
  - close: Trạng thái cho phép tự đóng cửa sổ chính của ứng dụng.
- Các phương thức:

- `__init__(self, *args, **kwargs)`: Constructor của class, thiết lập màu nền của cửa sổ, tên frame `fname` và trạng thái close của cửa sổ (xác định bằng tham số được truyền vào ở thuộc tính "login" và "register" của `FirstPage`). Các thuộc tính vừa đề cập được lấy từ `**kwargs`.
- `delete(self)`: Xóa khỏi màn hình tất cả các đơn vị GUI thuộc về class này.
- `render(self)`: Hiển thị các label, trường input và nút nhấn của các trang login và register tùy thuộc vào giá trị của `fname`. Bên trong method này còn có các hàm điều khiển giá trị của các trường input và các hàm điều hướng đến phần logic tiếp theo sau khi người dùng nhập thông tin và submit.

### 3. `ListPage`: Kế thừa class "Tk" của thư viện Tkinter

- Mô tả: Hiển thị cửa sổ mới chứa danh sách các người dùng được kết nối với người dùng đã đăng nhập, trạng thái online của các người dùng đó cũng như nút "Message" để yêu cầu nhắn tin với một người dùng đang online bất kỳ.
- Các thuộc tính:
  - `friend_list`: List chứa các người dùng đã kết nối với người dùng đã đăng nhập.
  - `status_list`: List chứa các trạng thái online của các người dùng đã kết nối với người dùng đã đăng nhập.
  - `listFrame`: Một frame trong cửa sổ chính `ListPage`, dùng để hiển thị các thông tin của trang này.
  - `listScroll`: Thanh scrollbar, cho phép người dùng cuộn cửa sổ này và xem toàn bộ các thông tin khi các thông tin này quá nhiều, không thể hiển thị hết được lên cửa sổ.
- Các phương thức:
  - `__init__(self, user)`: Chức năng tương tự như constructor của `FirstPage`. Tuy nhiên, constructor của `ListPage` còn khởi tạo danh sách người dùng đã kết nối cùng trạng thái của họ, khởi tạo frame mới, scrollbar và render các thông tin cần hiển thị trên cửa sổ này.
  - `render(self)`: Render các thông tin và các đơn vị GUI cần thiết: Scrollbar của trang, tên và trạng thái online của mỗi người dùng và nút "Message" tương ứng với mỗi người dùng. Lưu ý rằng nút "Message" sẽ có trạng thái disabled và không sử dụng được nếu người dùng tương ứng với nó đang offline.
  - `updateFriendStatus(self)`: Lấy dữ liệu status online mới nhất của các user và return.
  - `createChatBox(self, name)`: Tạo một thread và chat box mới khi chọn được người dùng để chat và được đối phương đồng ý.
  - `on_closing(self)`: Hiển thị cửa sổ mới xác nhận nếu người dùng muốn thoát ứng dụng và tiến hành đóng cửa sổ ứng dụng nếu người dùng xác nhận.



Hình 13: Class diagram cho UI

## 6 Kiểm thử và đánh giá hiệu năng

### 6.1 Kiểm thử

#### 6.1.1 Giao diện

- Nút Login và Register cần phải hoạt động bình thường: Khi ấn nút Login cần hiện ra giao diện để đăng nhập, khi ấn nút Register cần hiện ra giao diện đăng ký tài khoản.
- Register form: Khi điền đầy đủ các trường thỏa điều kiện sẽ đăng ký thành công. Thiếu một trong các trường hoặc có ít nhất một trường không thỏa điều kiện sẽ đăng ký thất bại.
- Login form: Khi điền đúng và đủ các trường sẽ đăng nhập thành công và điều hướng đến danh sách bạn bè. Nếu điền sai thông tin tài khoản hoặc thiếu 1 trường thì sẽ thông báo đăng nhập thất bại.
- Giao diện danh sách bạn bè: Phải cập nhật liên tục mỗi 1 giây, nút Message có thể ẩn khi người dùng đó online.
- Giao diện chat: Cần hiện đúng tên người dùng, nội dung tin nhắn phải hiển thị đúng vị trí và màu chữ (bên phải màu xanh đối với người gửi và bên trái màu trắng với người nhận), Khung để nhập nội dung và nút gửi tin nhắn, gửi file nằm phía dưới giao diện chat.

#### 6.1.2 Tính năng

- Gửi tin nhắn: Tin nhắn nhận được phải đúng với tin nhắn đã gửi.
  - Nội dung tin nhắn có thể chứa các ký tự utf-8.
  - Nội dung tin nhắn không thể xuống hàng được.
  - Thời gian gửi và nhận tin nhắn gần như tức thời ( $<1ms$ ).
  - Thông tin ngày giờ gửi và nhận tin nhắn theo thời gian thực tế (theo định dạng dd/mm/yyyy, hh:mm:ss).
- Gửi file: File nhận được phải cùng tên, nội dung và kích thước với file gửi. Người gửi có thể tùy ý chọn file trong máy và người nhận có thể tự do chọn folder để lưu.
  - Có thể gửi file toàn vẹn thuộc các định dạng: txt, csv, xlsx và các định dạng ảnh như: png, jpg, jfif.
  - Có thể gửi các file có tính chất mã hóa cao hơn: video, pdf tuy nhiên độ toàn vẹn dữ liệu và tính năng không được đảm bảo hoàn toàn.

### 6.2 Đánh giá hiệu năng

Nhóm tiến hành đánh giá hiệu năng dựa trên các tiêu chí: độ chịu tải, giới hạn khối lượng và tính bền vững.

Tất cả thí nghiệm đều được thực hiện trên máy của 1 thành viên trong nhóm, do đó kết quả sẽ có thể chênh lệch tùy vào cấu hình từng máy.

Cấu hình máy được sử dụng trong thí nghiệm:




- CPU: Intel Core i7 1065G7 - 1.3GHz (8 Cores)
- RAM: 8GB
- OS: Windows 11 Home Edition







### 6.2.1 Load Testing - Mức độ chịu tải

Hệ thống được giám sát để tính toán thời gian phản hồi và giữ hệ thống ổn định khi khối lượng công việc tăng lên.

- Server: duy trì ở mức 12-13 threads trong suốt quá trình thực thi, và không đổi dù số lượng người dùng tăng lên. Mặc dù bộ nhớ sử dụng tương đối thấp và hầu như không đổi, số lượng luồng chiếm dụng tương đối cao và chiếm tỉ trọng lớn trong CPU Usage (20-30)

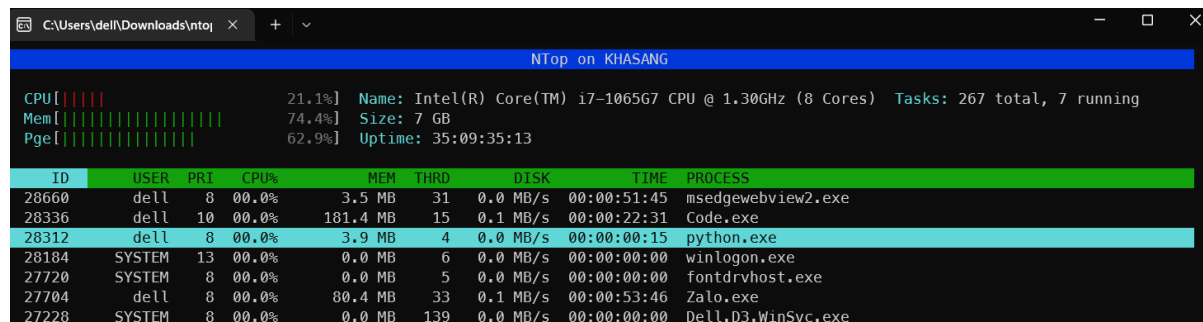
 Python	0%	6.3 MB	0 MB/s	0 Mbps
 Python	29.2%	7.0 MB	0 MB/s	0 Mbps
 Python	0%	0.4 MB	0 MB/s	0 Mbps

 python.exe	18996	Running	dell	00	512 K	x64	Python
 python.exe	19228	Running	dell	12	7,836 K	x64	Python
 python.exe	12884	Running	dell	00	512 K	x64	Python
 python.exe	12944	Running	dell	01	22,288 K	x64	Python

Hình 14: Tài nguyên sử dụng cho server: 12-13 threads

- Peer: mỗi Peer sử dụng 4 thread ở quá trình đăng nhập/ đăng kí (do phải duy trì tương tác với server cho việc xác thực) và duy trì ở mức 1 thread sau khi đăng nhập thành công (tức là khi giao diện danh sách online xuất hiện).



ID	USER	PRI	CPU%	MEM	THRD	DISK	TIME	PROCESS
28660	dell	8	00.0%	3.5 MB	31	0.0 MB/s	00:00:51:45	msedgewebview2.exe
28336	dell	10	00.0%	181.4 MB	15	0.1 MB/s	00:00:22:31	Code.exe
28312	dell	8	00.0%	3.9 MB	4	0.0 MB/s	00:00:00:15	python.exe
28184	SYSTEM	13	00.0%	0.0 MB	6	0.0 MB/s	00:00:00:00	winlogon.exe
27720	SYSTEM	8	00.0%	0.0 MB	5	0.0 MB/s	00:00:00:00	fontdrvhost.exe
27704	dell	8	00.0%	80.4 MB	33	0.1 MB/s	00:00:53:46	Zalo.exe
27228	SYSTEM	8	00.0%	0.0 MB	139	0.0 MB/s	00:00:00:00	Dell.D3.WinSvc.exe

Hình 15: Tài nguyên sử dụng của 1 peer đang login vào hệ thống

CPU[|||||]24.8%

Mem[|||||]73.7%

Pge[|||||]62.8%

Name: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz (8 Cores)

Tasks: 266 total, 8 running

Size: 7 GB

Uptime: 35:09:36:11

ID	USER	PRI	CPU%	MEM	THRD	DISK	TIME	PROCESS
28660	dell	8	00.0%	4.2 MB	31	0.0 MB/s	00:00:52:44	msedgewebview2.exe
28336	dell	10	00.0%	154.9 MB	14	0.0 MB/s	00:00:23:30	Code.exe
28312	dell	8	00.0%	3.8 MB	1	0.0 MB/s	00:00:01:14	python.exe
28184	SYSTEM	13	00.0%	0.0 MB	5	0.0 MB/s	00:00:00:00	winlogon.exe
27720	SYSTEM	8	00.0%	0.0 MB	5	0.0 MB/s	00:00:00:00	fontdrvhost.exe
27704	dell	8	00.0%	80.8 MB	33	0.1 MB/s	00:00:54:44	Zalo.exe
27436	dell	8	00.0%	7.8 MB	8	0.0 MB/s	00:00:00:05	dllhost.exe

Hình 16: Tài nguyên sử dụng của 1 peer sau khi login vào hệ thống

Name	Status	CPU	Memory	Disk	Network	GPU	GPU engine	Power usage	Power usage trend
Apps (7)									
Python		1.5%	18.3 MB	0 MB/s	0 Mbps	0%		Low	Very low
b									
Python		1.6%	20.0 MB	0 MB/s	0 Mbps	0%		Low	Very low
a									

Hình 17: Tài nguyên sử dụng 2 peer đang login vào hệ thống

Name	Status	CPU	Memory	Disk	Network	GPU	GPU engine	Power usage	Power usage trend
Apps (8)									
Python (2)		2.3%	26.6 MB	0 MB/s	0 Mbps	0%		Low	Very low
a									
CHATROOM									
Python (2)		2.8%	24.6 MB	0 MB/s	0 Mbps	0%		Low	Very low
b									
CHATROOM									

Hình 18: Tài nguyên sử dụng 2 peer sau khi login vào hệ thống

python.exe	18320	Running	dell	01	35,308 K	x64	Python
python.exe	15668	Running	dell	03	33,264 K	x64	Python
python.exe	13812	Running	dell	00	512 K	x64	Python
python.exe	28304	Running	dell	12	8,428 K	x64	Python

Hình 19: Tài nguyên sử dụng chỉ tiết 2 peer sau khi login vào hệ thống

### 6.2.2 Volume Testing - Giới hạn truyền tin

Kiểm thử khối lượng xác định phần mềm hoạt động hiệu quả như thế nào với khối lượng lớn dữ liệu. Nhóm tiến hành chuyển file với kích thước lần lượt là 1MB, 10MB và 100MB và buffersize là 4096B, sau đó thống kê kết quả dựa trên Wireshark Packet Sniffer.

- File 1MB:
  - Thời điểm người gửi phát tín hiệu: 21.303731s
  - Thời điểm người gửi nhận tín hiệu hoàn tất: 21.310736s
  - Tổng thời gian: 7ms
- File 10MB:
  - Thời điểm người gửi phát tín hiệu: 4.422571s
  - Thời điểm người gửi nhận tín hiệu hoàn tất: 12.228450s
  - Tổng thời gian: 7.9s
- File 100MB:
  - Thời điểm người gửi phát tín hiệu: 228.441693s
  - Thời điểm người gửi nhận tín hiệu hoàn tất: 245.586187s
  - Tổng thời gian: 18s

No.	Time	Source	Destination	Protocol	Length	Info
455	21.303731	192.168.43.1	192.168.43.1	TCP	136	50662 → 81 [PSH, ACK] Seq=1 Ack=1 Win=1279 Len=92
457	21.304161	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=93 Ack=1 Win=1279 Len=4096
459	21.304203	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=4189 Ack=1 Win=1279 Len=4096
461	21.304512	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=8285 Ack=1 Win=1279 Len=4096
463	21.304554	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=12381 Ack=1 Win=1279 Len=4096
465	21.304596	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=16477 Ack=1 Win=1279 Len=4096
467	21.304620	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=20573 Ack=1 Win=1279 Len=4096
469	21.304659	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=24669 Ack=1 Win=1279 Len=4096
471	21.304695	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=28765 Ack=1 Win=1279 Len=4096
473	21.304730	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=32861 Ack=1 Win=1279 Len=4096
475	21.304758	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=36957 Ack=1 Win=1279 Len=4096
477	21.304813	192.168.43.1	192.168.43.1	TCP	4140	50662 → 81 [PSH, ACK] Seq=41053 Ack=1 Win=1279 Len=4096

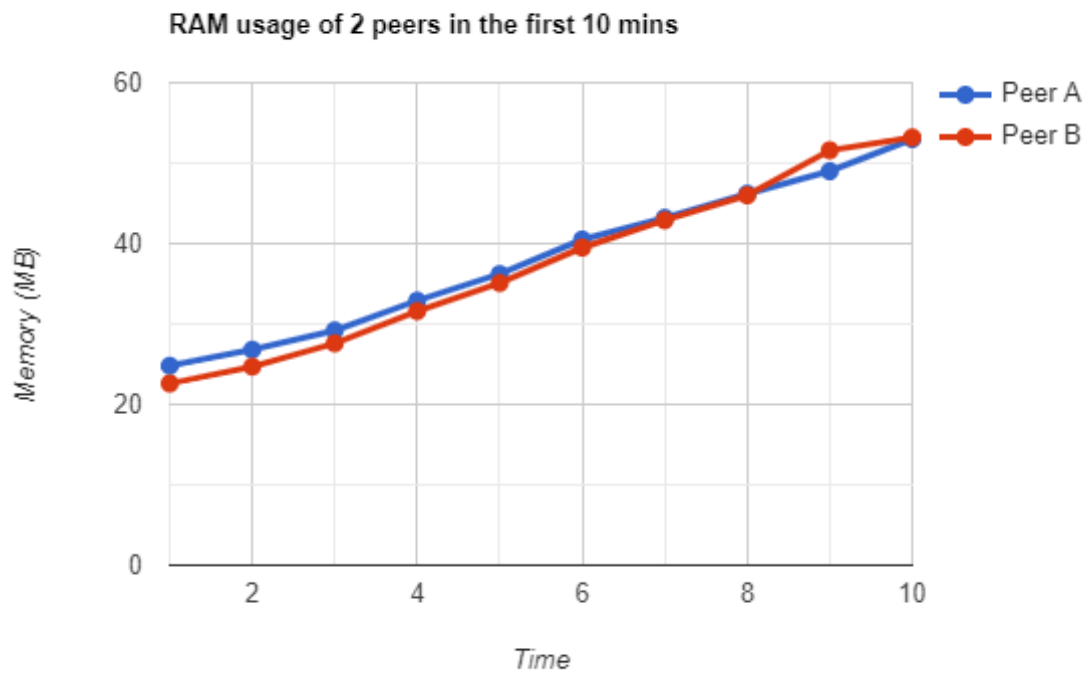
Frame 455: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits) on interface \Device\NPF{...}	0000	02 00 00 00 45 00 00 84	c8 10 40 00 80 06 00 00	....E....@....
> Null/Loopback	0010	c0 a8 2b 01 c0 a8 2b 01	c5 e6 00 51 a2 6f 8b 4c	...+...+...Q o L
> Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.1	0020	08 b5 69 f9 50 18 04 ff	c6 41 00 00 70 22 7a 79	...i.p...A...[ty
> Transmission Control Protocol, Src Port: 50662, Dst Port: 81, Seq: 1, Ack: 1, Len: 92	0030	70 65 22 3a 20 22 46 22	2c 20 22 66 6c 61 67 22	...pe": "F", "flag
> Data (92 bytes)	0040	3a 20 31 2c 20 22 7a 69	6d 65 22 3a 20 22 31 32	...: 1, "time": "12
	0050	2f 31 31 2f 32 30 32 32	2c 20 30 39 3a 3a 36 3a	.../11/2022 , 09:46:
	0060	34 38 22 2c 20 22 66 6e	61 6d 65 22 3a 20 22 31	...48", "fname": "1
	0070	2e 74 78 74 22 2c 20 22	66 73 69 7a 65 22 3a 20	...txt", " fsize":
	0080	31 30 34 38 35 37 36 7d		1048576]

Hình 20: Theo dõi gói gửi bằng Wireshark

### 6.2.3 Endurance testing - Mức độ ổn định của hệ thống

Nhóm đánh giá hiệu suất phần mềm với khối lượng công việc thông thường trong một thời gian dài. Do thời gian có hạn, nhóm đặt giới hạn là 10 phút sau khi người dùng đăng nhập thành công.

- Server: nhìn chung Server chiếm dụng RAM không nhiều, cũng như là số lượng thread sử dụng không đổi. Tuy nhiên CPU usage thường ở mức tầm 100MB.
- Peer: RAM do Peer sử dụng tăng tuyến tính theo thời gian, bởi tính năng cập nhật giao diện (trạng thái online của friend list) sau một chu kỳ nhất định. Chúng ta có thể làm giảm tốc độ tiêu thụ bằng cách tăng chu kỳ cập nhật.
- Sau khoảng thời gian trên, các peer vẫn có thể chat với nhau bình thường, tuy nhiên tính năng gửi file chỉ hoạt động trên dung lượng 10MB (fail test 100MB).



Hình 21: Tiêu tốn tài nguyên trong 10 phút đầu

### 6.3 Chạy thử trên các máy cùng một subnet

#### 6.3.1 Cài đặt để tiến hành chạy thử

- Thiết lập **EXTERNAL\_IP\_SERVER** trong file **peer.py** thành địa chỉ IP của máy dùng làm server.

```

13
14 # Your External IPv4
15 # EXTERNAL_IP_SERVER = '192.168.1.6'
16 EXTERNAL_IP_SERVER = "192.168.227.215"
17

```

Hình 22: Thiết lập EXTERNAL\_IP\_SERVER

- Tiến hành mở server và kết nối các client như bình thường.

### 6.3.2 Thời gian truyền tin

Nhóm thực hiện việc gửi file qua lại giữa hai máy kết nối cùng một WIFI được phát từ điện thoại có 4G. Thời gian cho việc gửi và nhận file được ghi nhận như sau:

- File 1MB: Tổng thời gian truyền là 0.44s
- File 10MB: Tổng thời gian truyền là 0.33s
- File 100MB: Tổng thời gian truyền là 59.24s

Có thể thấy, việc gửi và nhận file tốn nhiều thời gian hơn so với kiểm thử chương trình trên cùng một. Ngoài ra, thời gian truyền file với độ lớn 10MB là nhỏ hơn file có độ lớn 1MB. Đó là do việc truyền file giữa hai máy chịu ảnh hưởng từ tốc độ đường truyền của WIFI.

## 7 Hướng dẫn sử dụng

### 7.1 Server

Chúng ta sẽ chạy server trên 1 máy để tiến hành khởi động hệ thống, sau khi khởi tạo sẽ có kết quả như sau

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\OneDrive\Máy tính\BTL\HK221\Computer Networks\ComputerNetwork_Assignment1>
python utils/server.py
UDP server started on port 3004

TCP Server started on port 3000

█
```

Hình 23: Server được khởi tạo với port 3000 dành cho TCP server, 3004 dành cho UDP server

### 7.2 Peer

#### 7.2.1 Register account

Sau khi tiến hành khởi động server, việc tiếp theo chúng ta sẽ cần một accooount để sử dụng hệ thống này. Khởi tạo giao diện người dùng với câu lệnh *py utils/LinkedUI.py*

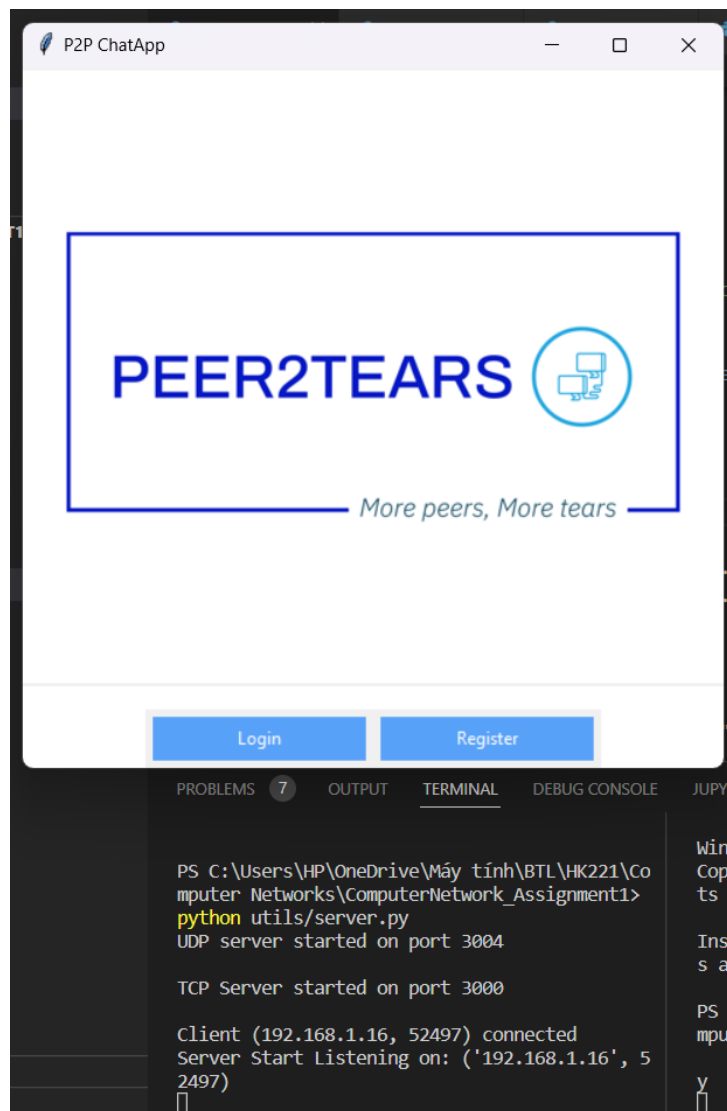
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\OneDrive\Máy tính\BTL\HK221\Computer Networks\ComputerNetwork_Assignment1>
                                                                    utils/LinkedUI.py
y█
```

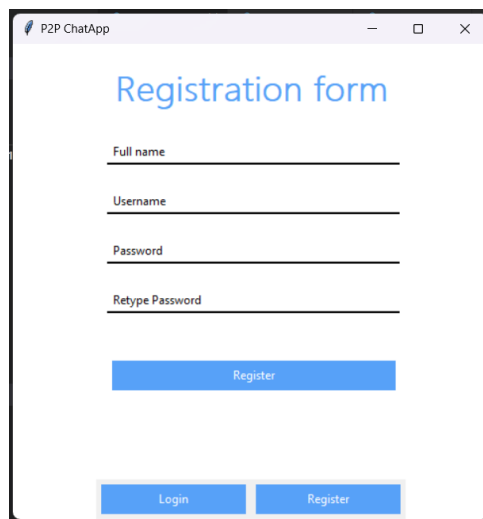
Hình 24: Người dùng khởi tạo giao diện

Sau khi khởi tạo, một giao diện sẽ xuất hiện



Hình 25: Giao diện người dùng xuất hiện

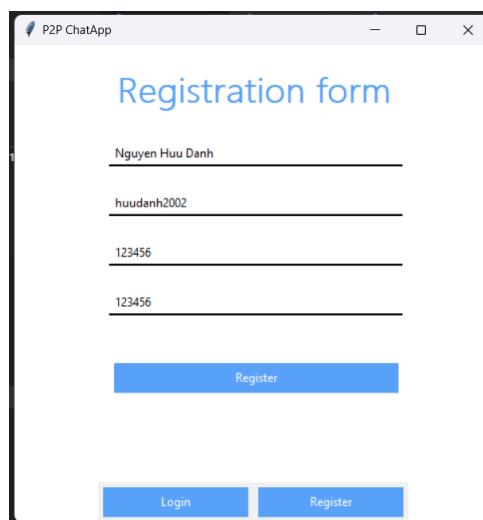
Người dùng chọn Register để tạo tài khoản mới



The screenshot shows a web browser window titled "P2P ChatApp". The main heading is "Registration form" in blue. Below the heading are four input fields: "Full name", "Username", "Password", and "Retype Password". Each field has a horizontal line for text entry. Below these fields is a large blue button labeled "Register". At the bottom of the form, there are two smaller blue buttons: "Login" on the left and "Register" on the right.

Hình 26: Giao diện tạo người dùng

Sau đó điền thông tin vào các ô input trên giao diện như họ tên, username, password

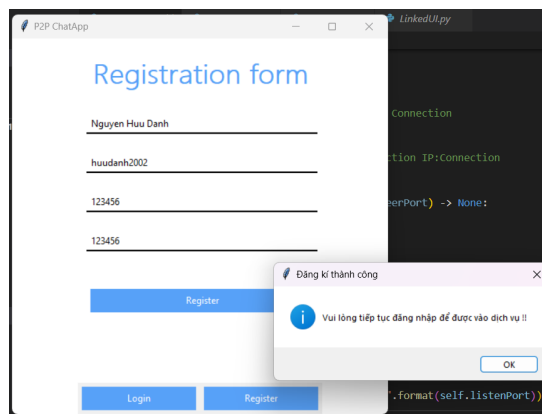


This screenshot shows the same "Registration form" as in Figure 26, but with the input fields filled. The "Full name" field contains "Nguyen Huu Danh", the "Username" field contains "huudanh2002", and both the "Password" and "Retype Password" fields contain "123456". The "Register" button remains blue, while the "Login" and "Register" buttons at the bottom are still present.

Hình 27: Điền các thông tin cần thiết

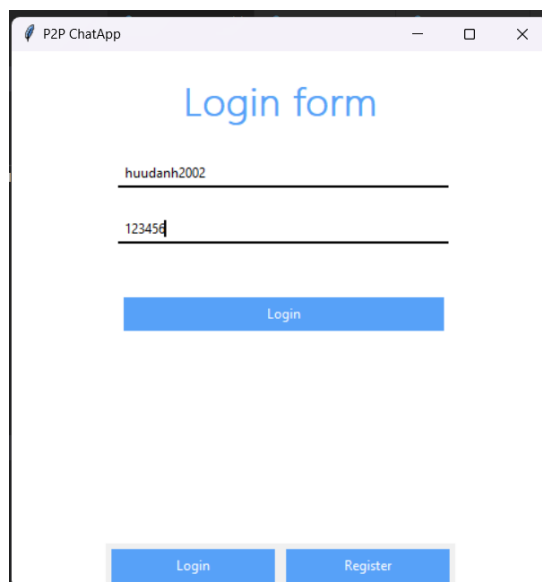


Khi đăng ký xong, sẽ xuất hiện thông báo trở lại trang đăng nhập để sử dụng dịch vụ



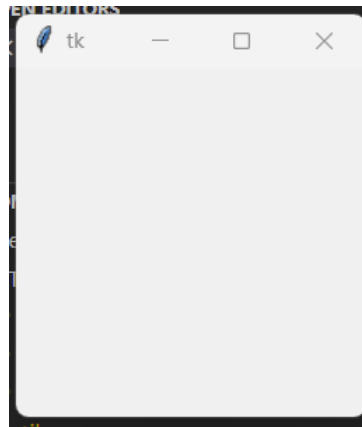
Hình 28: Thông báo nổ ra

Sau đó ta quay trở lại trang đăng nhập (nút Login trên client) để tiến hành đăng nhập



Hình 29: Trở về trang đăng nhập

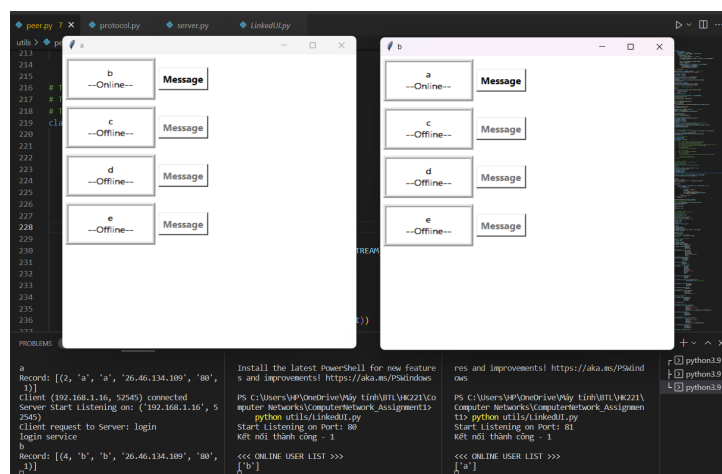
Sau khi đăng nhập sẽ xuất hiện giao diện nơi chứa danh sách các bạn bè. Do account này mới được tạo (chưa kết bạn) nên ta sẽ không thấy gì trên đó cả.



Hình 30: Giao diện danh sách bạn bè (account mới tạo)

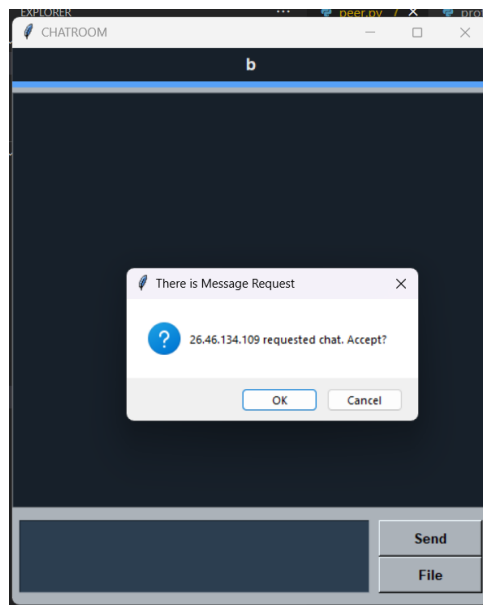
### 7.2.2 Chat and transfer file

Để hướng dẫn sử dụng phần này, nhóm sẽ sử dụng 2 account có sẵn có username là a và b, chúng đều đã được kết bạn với nhau. Ở trường hợp này nhóm sẽ tạo 3 terminal để mở server và 2 client trên cùng 1 máy, một lưu ý là mỗi client phải sử dụng port khác nhau để kết nối được với server



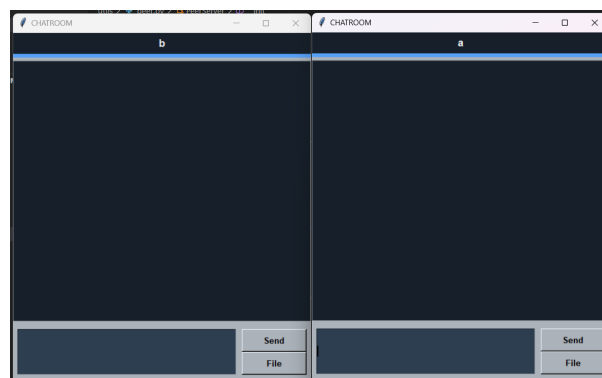
Hình 31: Giao diện chính sau khi đăng nhập cả hai account a và b

Sau đó ta nhấn "Message" vào người mà ta muốn trò chuyện, bên này sẽ hiển thị trước cửa sổ chat còn bên kia sẽ nhận được một thông báo để phản hồi việc tiếp nhận cuộc trò chuyện này



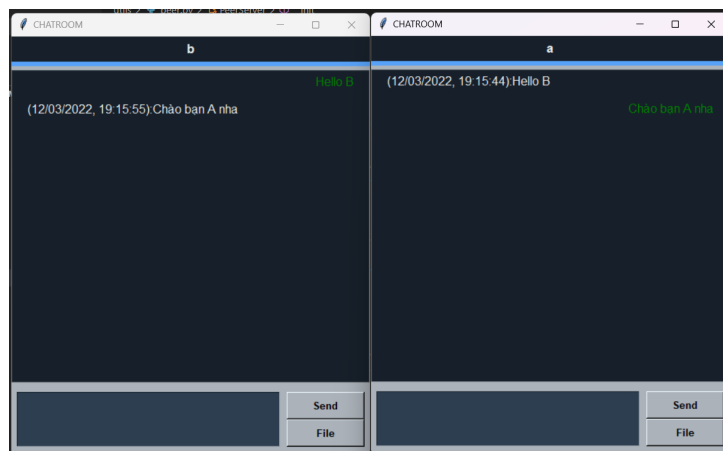
Hình 32: Giao diện chat và thông báo tiếp nhận cuộc trò chuyện

Khi người kia chấp nhận lời mời, bên họ sẽ xuất hiện một cửa sổ chat để giao tiếp với người này



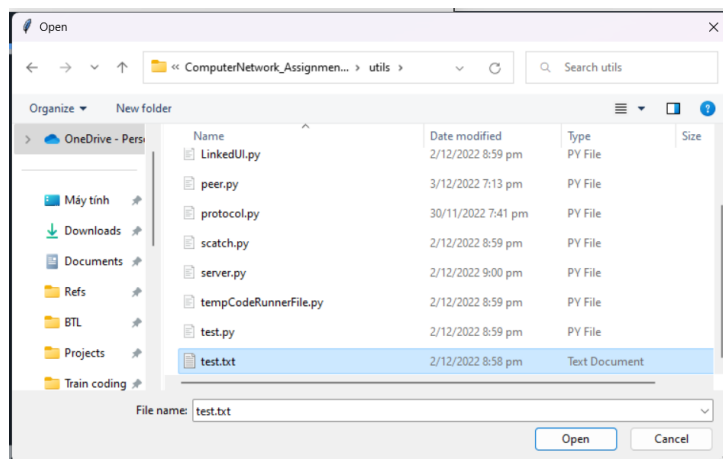
Hình 33: Người dùng kia cũng xuất hiện cửa sổ chat

Hai bên có thể nhắn tin qua lại như sau



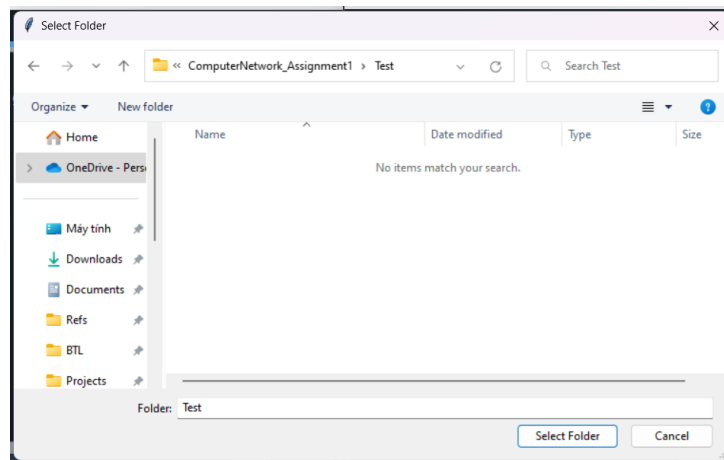
Hình 34: Hai người dùng nhắn tin qua lại với nhau

Để có thể chuyển file cho người khác, ta nhấn vào nút File, khi đó hệ thống sẽ báo người dùng chọn file để gửi



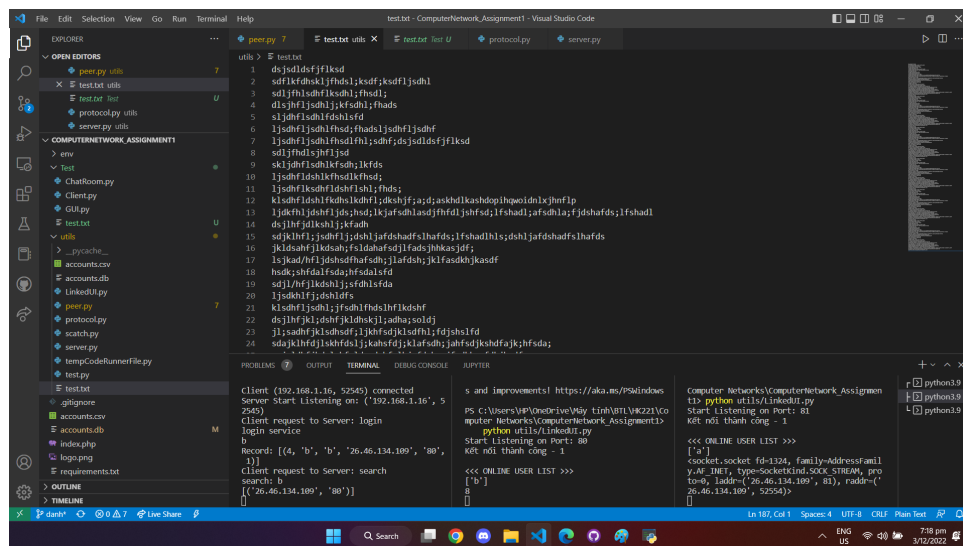
Hình 35: Người dùng chọn file để gửi

Người dùng bên kia sẽ nhận được thông báo chọn folder để nhận file sắp tới



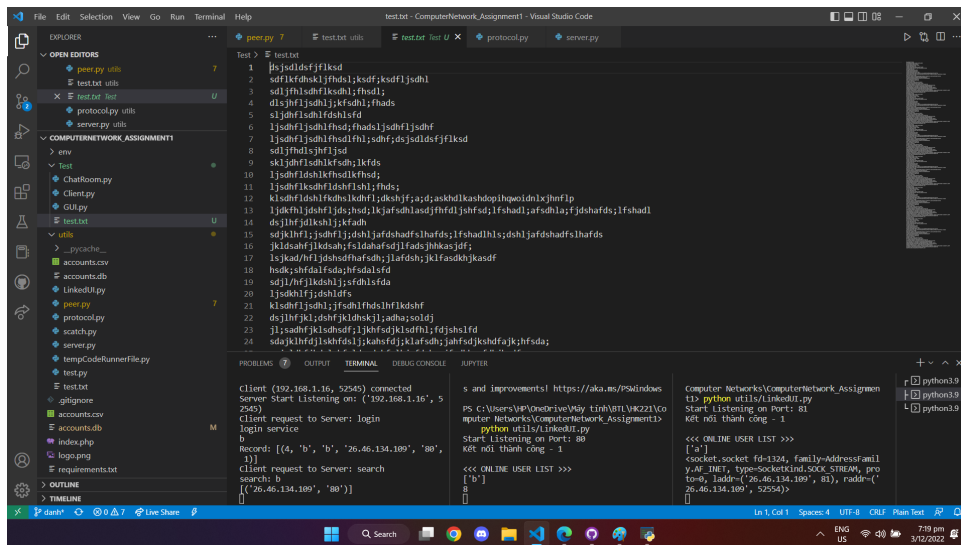
Hình 36: Người dùng chọn file để gửi

Đây là nội dung của file mà người dùng gửi, nhóm sẽ show ở đây để đối chiếu với file người dùng kia nhận được



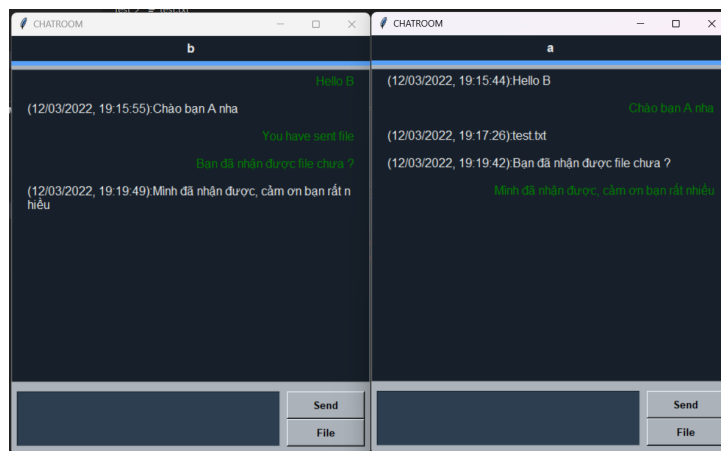
Hình 37: Nội dung file được gửi

Như hình trên khi ta chọn folder Test để nhận file, file *test.txt* đã được gửi về đúng directory và nội dung nhận được hoàn toàn đầy đủ



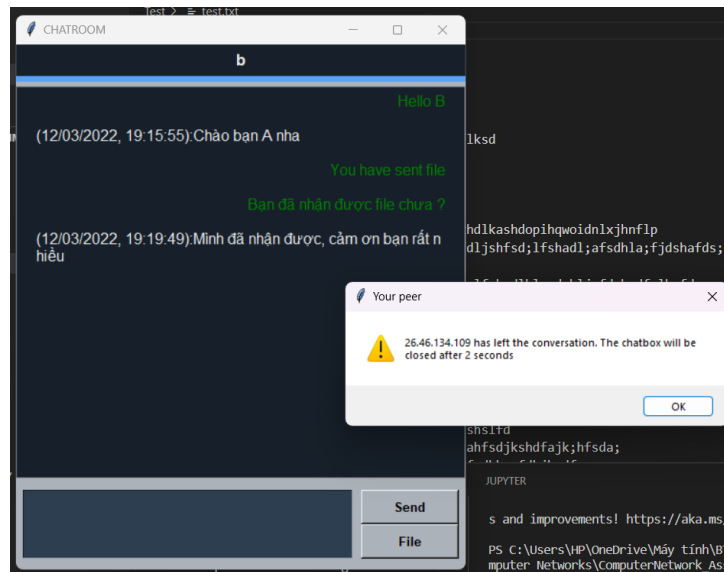
Hình 38: Nội dung file được nhận bên phía người dùng kia

Sau khi gửi file thì hai người dùng có thể tiếp tục chat với nhau như bình thường



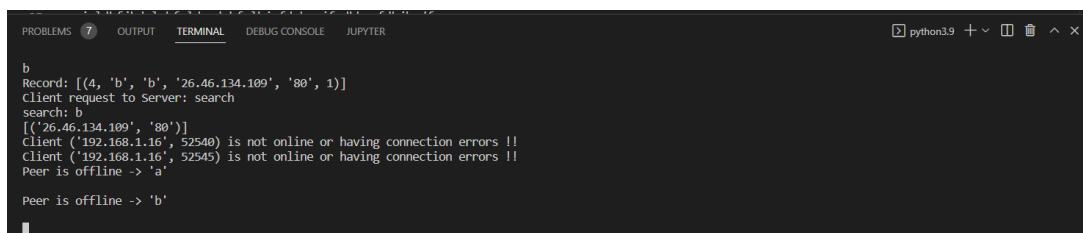
Hình 39: Hai người dùng tiếp tục trò chuyện với nhau

Khi một người dùng đóng cửa sổ chat, người dùng bên kia sẽ nhận được thông báo cuộc trò chuyện sắp đóng. Khi nhấn OK thì 2 giây sau cửa sổ trò chuyện sẽ tự đóng lại



Hình 40: Một người dùng rời khỏi cuộc trò chuyện

Khi người dùng tắt hệ thống thì server sẽ xử lý như sau: Hiển thị thông báo người dùng nào đã offline trên terminal của server



Hình 41: Khi có người dùng tắt hệ thống đi

## Tài liệu

- [1] TCP socket error code. <https://gist.github.com/gabrielfalcao/4216897>
- [2] Siu Man Lui and Sai Ho Kwok, Interoperability of Peer-To-Peer File Sharing Protocols. [https://www.sigecom.org/exchanges/volume\\_3/3.3-Lui.pdf](https://www.sigecom.org/exchanges/volume_3/3.3-Lui.pdf)
- [3] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble, A Measurement Study of Peer-to-Peer File Sharing Systems. <https://people.mpi-sws.org/~gummadi/papers/p2ptechreport.pdf>
- [4] Siu Man Lui and Sai Ho Kwok, Interoperability of Peer-To-Peer File Sharing Protocols. [https://www.sigecom.org/exchanges/volume\\_3/3.3-Lui.pdf](https://www.sigecom.org/exchanges/volume_3/3.3-Lui.pdf)