

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



MẠNG MÁY TÍNH (CO3093)

BÀI TẬP LỚN 1

DEVELOP A NETWORK APPLICATION (FILE-SHARING APPLICATION)

GVHD: Lê Bảo Khánh
Nhóm: L07_7
SV: Đào Duy Long - 2113928
Trương Thuận Hưng - 2113619
Nguyễn Phạm Thiên Phúc - 2114445
Trần Minh Hiếu - 2113363

TP. HỒ CHÍ MINH, THÁNG 10/2023

1 Danh sách thành viên & Tiến độ

Phase 1:

Mssv	Họ và Tên	Công việc	Đánh giá
2113928	Đào Duy Long	Mô tả	100%
2113619	Trương Thuận Hưng	Mục tiêu	100%
2114425	Nguyễn Phạm Thiên Phúc	Giao thức	100%
2113363	Trần Minh Hiếu	Chức năng, Use case	100%

Phase 2:

Mssv	Họ và Tên	Công việc	Đánh giá
2113928	Đào Duy Long	Hiện thực phần backend cho server, models	100%
2113619	Trương Thuận Hưng	Hiện thực phần giao diện trang web Viết báo cáo và trình bày powerpoint	100%
2114445	Nguyễn Phạm Thiên Phúc	Hiện thực phần truyền tải P2P Hiện thực phần backend cho server	100%
2113363	Trần Minh Hiếu	Hiện thực phần giao diện trang web Kết nối API đến Server	100%



Mục lục

1	Danh sách thành viên & Tiến độ	1
2	Giới thiệu	4
2.1	Bối cảnh	4
2.2	Mục tiêu	4
3	Mô tả hệ thống	5
3.1	Phân tích yêu cầu	5
3.1.1	Yêu cầu chức năng	5
3.1.2	Yêu cầu phi chức năng	6
3.2	Giao thức	7
3.2.1	Giao thức Client-Server	7
3.2.2	Giao thức Peer-to-Peer	8
4	Protocol	9
4.1	TCP trong kết nối Peer-to-Peer	9
4.2	HTTP trong kết nối Client-Server	10
5	Diagram	12
5.1	Use Case Diagram	12
5.2	Activity Diagram	13
5.2.1	publish lname fname	13
5.2.2	fetch fname	14
5.2.3	discover hostname	15
5.2.4	ping hostname	16
5.3	Sequence Diagram	17
5.3.1	publish lname fname	17
5.3.2	fetch fname	18
5.3.3	discover hostname	19
5.3.4	ping hostname	20
5.4	Class Diagram	21
5.5	Deployment Diagram	22
6	Technology and Architecture	23
6.1	Technology	23
6.1.1	ReactJS	23
6.1.2	TailwindCSS	25
6.1.3	NodeJS	27
6.1.4	ExpressJS	29
6.2	Architecture	31



7	Kiểm tra và chạy thử ứng dụng	32
7.1	Tính năng đăng ký và đăng nhập	32
7.2	Tính năng của client	36
7.2.1	Tính năng xem file trong repository của client	36
7.2.2	Tính năng xem danh sách các file được chia sẻ	38
7.2.3	Tính năng publish file	40
7.2.4	Tính năng delete file	42
7.2.5	Tính năng fetch file	44
7.2.6	Các tính năng khác của client	46
7.3	Các tính năng của server	48
8	Implementation Code	52
9	Hướng dẫn sử dụng	52
10	Tài liệu tham khảo	53

2 Giới thiệu

2.1 Bối cảnh

Ứng dụng chia sẻ file P2P là một mạng máy tính trong đó các máy tính kết nối trực tiếp với nhau để truyền dữ liệu. Điều này trái ngược với mô hình truyền thống, trong đó dữ liệu được truyền qua một máy chủ trung tâm. Với sự phân bổ tài nguyên hợp lý tránh sự quá tải của Server, mô hình P2P chia sẻ file ra đời giúp tránh phụ thuộc quá nhiều vào Server và tài nguyên của Server dẫn đến tắt nghẽn do lượng truy cập quá nhiều một thời điểm.

- **Chia sẻ file giữa các cá nhân:** Ứng dụng chia sẻ file P2P có thể được sử dụng để chia sẻ file giữa các cá nhân, chẳng hạn như bạn bè, gia đình hoặc đồng nghiệp. Điều này có thể được thực hiện qua mạng cục bộ hoặc qua Internet.
- **Chia sẻ file trong doanh nghiệp:** Ứng dụng chia sẻ file P2P có thể được sử dụng để chia sẻ file trong doanh nghiệp. Điều này có thể được sử dụng để chia sẻ tài liệu, mã nguồn hoặc các loại file khác giữa các nhân viên.
- **Chia sẻ file trong các tổ chức phi lợi nhuận:** Ứng dụng chia sẻ file P2P có thể được sử dụng để chia sẻ file trong các tổ chức phi lợi nhuận. Điều này có thể được sử dụng để chia sẻ tài liệu, hình ảnh hoặc video giữa các thành viên của tổ chức.

2.2 Mục tiêu

Mục tiêu chính của dự án là phát triển một ứng dụng chia sẻ tệp tin (File-sharing application) đơn giản, sử dụng giao thức TCP/IP. Hệ thống bao gồm một máy chủ trung tâm (Server) và các máy khách (Client) kết nối với máy chủ. Các máy khách sẽ thông báo cho máy chủ về các tệp tin có trong local repository của mình và có thể yêu cầu truy xuất các tệp tin từ các máy khách khác.

Đối với Server:

- Server nắm giữ thông tin về các Client tham gia vào hệ thống.
- Server quản lý danh sách các file mà các Client chia sẻ.
- Khi nhận được yêu cầu tải file từ Client, Server phải có khả năng xác định file đó đang thuộc Client nào và phản hồi lại.

Đối với Client:

- Client có khả năng gửi yêu cầu tải file lên Server và nhận được phản hồi từ Server về thông tin Client đang chứa file cần tải.
- Client có thể tương tác với Client khác theo cơ chế Peer-to-Peer để có thể gửi và nhận file giữa các Client.

- Client có thể upload file từ máy vào repository của mình và thông báo cho Server khi hoàn tất.
- Tương tự, Client cũng có thể xóa file khỏi repository và thông báo cho Server khi hoàn tất.
- Multithread: Cho phép nhiều Client cùng tải một file của một Client đích ở cùng một thời điểm.

3 Mô tả hệ thống

3.1 Phân tích yêu cầu

3.1.1 Yêu cầu chức năng

Chức năng cho ứng dụng:

- Đăng ký và tạo tài khoản: Ứng dụng phải cung cấp chức năng cho người dùng đăng ký và tạo tài khoản để tham gia hệ thống.
- Đăng nhập và đăng xuất: Ứng dụng phải hỗ trợ chức năng đăng nhập và đăng xuất cho người dùng.

Chức năng cho Server:

- Xem danh sách hostname các client ("**getHostname**"): Máy chủ (Server) phải có khả năng xem danh sách hostname của tất cả các client đã tham gia vào ứng dụng.
- Kiểm tra trạng thái của một client ("**ping hostname**"): Máy chủ (Server) sẽ được cung cấp chức năng kiểm tra trạng thái của một client.
- Xem thông tin các file đang chia sẻ của một client ("**discover hostname**"): Máy chủ (Server) được cung cấp khả năng xem thông tin về các file đang được chia sẻ bởi các client.

Chức năng cho Client:

- Upload file vào repository ("**publish lname fname**"): Client phải có khả năng upload file từ máy của họ vào repository của client trong ứng dụng và sau khi hoàn thành, thông báo sẽ được gửi đến máy chủ (Server).
- Xem danh sách file trong local repository ("**myRepository**"): Client phải có chức năng xem các danh sách file trong local repository của mình.
- Xem danh sách file và tìm kiếm file ("**communityFile**"): Client phải có chức năng xem danh sách các file đang được chia sẻ trong ứng dụng.
- Tải file từ client khác ("**fetch fname**" hoặc "**fetch fname peerIP**"): Client phải có khả năng tải file từ client khác về repository của họ. Để có thể tải file thì client sẽ gửi yêu cầu tìm thông tin về client nào đang nắm giữ file đó đến máy chủ (Server), máy chủ sẽ gửi lại cho client thông tin của một client đang nắm giữ file cần tải, sau đó sẽ bắt đầu tải file từ client đó.

- Xóa file khỏi repository ("**delete fname**") : Client có khả năng xóa file khỏi repository chia sẻ của họ và gửi thông báo đến máy chủ (Server).
- Xem danh sách file nằm ngoài repository trên máy người dùng ("**myDisk**")

3.1.2 Yêu cầu phi chức năng

a. Cho toàn hệ thống:

Hiệu suất

- Đảm bảo khả năng tải file thành công 90%: Hệ thống phải có khả năng đảm bảo rằng tối thiểu 90% các tải file được thực hiện thành công mà không gặp lỗi.
- Tỷ lệ mất gói tin dưới 10%: Hệ thống phải đảm bảo rằng tỷ lệ mất gói tin trong quá trình truyền dữ liệu không vượt quá 10%.

Tính khả dụng

- Giao diện của hệ thống phải hiển thị đầy đủ các tính năng chính: public, fetch, discover, ping...
- Ứng dụng dễ sử dụng: Người dùng mới phải có khả năng sử dụng cơ bản ứng dụng một cách dễ dàng trong vòng 2 phút sau khi làm quen với ứng dụng.

Tính bảo mật

- Đảm bảo các file được chia sẻ trong hệ thống không bị rò rỉ ra bên ngoài.
- Xác thực và phân quyền: Người dùng cần có tài khoản và mật khẩu mới có thể đăng nhập và sử dụng được hệ thống. Đặc biệt, đối với tài khoản admin có thể sử dụng được các chức năng của server.

Tính mở rộng

- Ứng dụng phải có khả năng mở rộng thêm các tính năng trong tương lai nếu có thêm yêu cầu như: trò chuyện qua tin nhắn, quản lý bạn bè, tin nhắn thoại...

b. Cho chức năng:

Chức năng publish lname fname:

- Thời gian server nhận được thông báo tính từ khi client publish file lên repository hoàn tất sẽ không quá 2s.
- Tên fname của file trong repository của client phải là khác với các tên file đã được chia sẻ có trong hệ thống.

Chức năng fetch fname và fetch fname peerIP:

- Multithread: Hệ thống cần hỗ trợ chạy nhiều luồng để cho phép nhiều client cùng tải file đồng thời đến một client đích.
- Thời gian từ khi client yêu cầu một file đến khi nhận được file từ client khác hoặc nhận thông báo yêu cầu thất bại từ server khi không có file sẽ không quá 2s.

Chức năng discover hostname:

- Khi được gọi, chức năng "discover" cần chấp nhận thông tin cụ thể về máy khách, chẳng hạn như địa chỉ IP hoặc tên máy.
- Tính năng "discover" cần trả về danh sách các tập tin cục bộ có sẵn trên máy khách cụ thể, cung cấp thông tin về tên tập tin, kích thước, ngày tạo và các chi tiết khác nếu có.

Chức năng ping hostname:

- Tính năng này cần có khả năng chấp nhận hostname của client để thực hiện kiểm tra.
- Sau khi thực hiện kiểm tra, tính năng "ping" cần trả về kết quả, thông báo về trạng thái của client (có thể là "hoạt động", "không hoạt động" hoặc "không tồn tại hostname").

3.2 Giao thức

3.2.1 Giao thức Client-Server

Xác định giao thức dùng cho các chức năng liên quan đến mô hình Client-Server:

Chức năng Đăng kí và tạo tài khoản:

- Giao thức sử dụng là: **HTTP**.
- Cơ sở lý thuyết: HTTP được sử dụng cho chức năng đăng ký do tính phổ biến và tiện lợi. Giao thức này cho phép gửi thông tin đăng ký từ máy khách đến máy chủ một cách dễ dàng thông qua yêu cầu HTTP POST. Hơn nữa, HTTP hỗ trợ tích hợp bảo mật thông qua HTTPS, giúp đảm bảo tính bảo mật của thông tin đăng ký.

Chức năng Đăng nhập và đăng nhập:

- Giao thức sử dụng là: **HTTP**.
- Cơ sở lý thuyết: HTTP cũng được sử dụng cho chức năng đăng nhập và đăng xuất vì tính phổ biến và đơn giản của giao thức. Người dùng có thể gửi thông tin đăng nhập (tên đăng nhập hoặc email và mật khẩu) thông qua yêu cầu HTTP POST và đăng xuất thông qua yêu cầu HTTP POST

Chức năng Kiểm tra trạng thái của một client ("ping hostname"):

- Giao thức sử dụng là: **ICMP**.
- Cơ sở lý thuyết: ICMP (Internet Control Message Protocol) là giao thức chuyên dụng để kiểm tra trạng thái của mạng và các thiết bị mạng. Nó thường được sử dụng để kiểm tra kết nối và tính khả dụng của một máy client thông qua gửi gói tin kiểm tra (ping) và nhận phản hồi.

Chức năng Xem thông tin các file đang chia sẻ của một client ("discover hostname"):

- Giao thức sử dụng là: **HTTP**.
- Cơ sở lý thuyết: Giao thức HTTP được sử dụng cho chức năng này vì tính phổ biến và tích hợp dễ dàng. Máy chủ có thể gửi yêu cầu HTTP để lấy thông tin về các file đang chia sẻ từ máy client, và máy client có thể phản hồi thông tin thông qua phản hồi HTTP.

Chức năng Upload file vào repository ("publish lname fname"):

- Giao thức sử dụng là: **HTTP**.
- Cơ sở lý thuyết: Giao thức HTTP là để cho phép máy khách gửi thông tin (tên và kích thước) của file sau khi đã tải lên thành công vào repository. Khi máy khách hoàn tất việc tải lên, nó sử dụng yêu cầu HTTP POST để truyền thông tin về file (tên, kích thước) lên máy chủ, và sau đó máy chủ lưu trữ thông tin này trong database.

Chức năng Xóa file khỏi repository ("delete fname")

- Giao thức sử dụng là: **HTTP**.
- Cơ sở lý thuyết: Giao thức HTTP là để cho phép máy khách gửi thông báo đến máy chủ sau khi thực hiện thành công việc xóa file từ repository. Khi máy khách hoàn tất việc xóa file, nó sử dụng yêu cầu HTTP POST để gửi thông tin về việc xóa file lên máy chủ. Máy chủ sau đó có thể cập nhật thông tin trong database.

3.2.2 Giao thức Peer-to-Peer

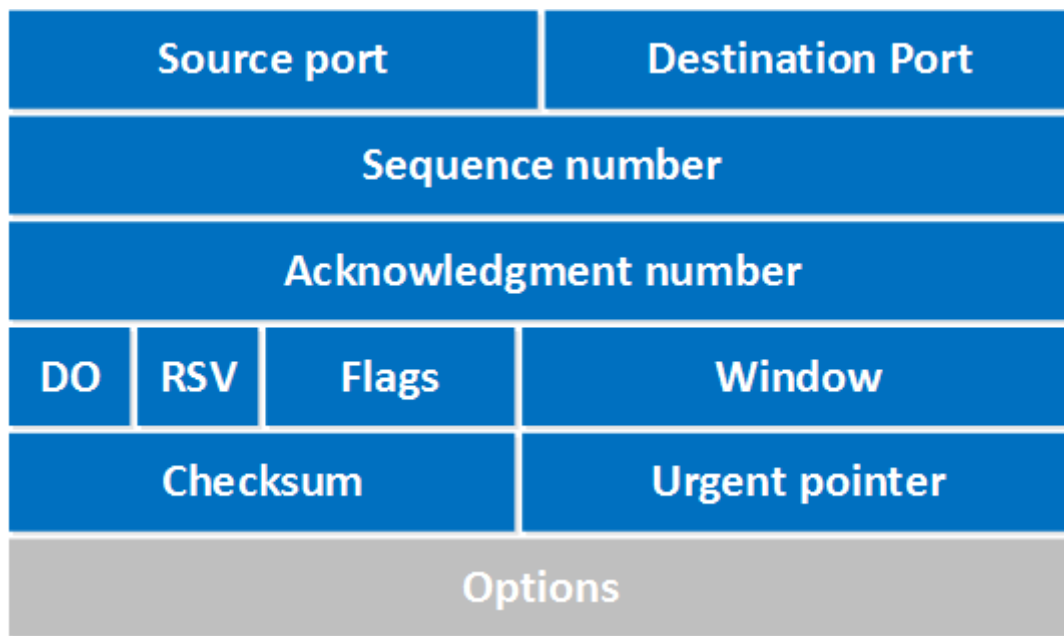
Xác định giao thức dùng cho chức năng liên quan đến mô hình Peer-to-Peer:

Tải file từ client khác ("fetch fname") hoặc ("fetch fname peerIP")

- Giao thức sử dụng là: **TCP**.
- Cơ sở lý thuyết: Sử dụng TCP protocol trong việc truyền dữ liệu giữa các peers. TCP cho phép tạo các kết nối trực tiếp giữa các máy client, thực hiện việc truyền tải dữ liệu dưới dạng các luồng dữ liệu (data streams). Điều này giúp tối ưu hóa trải nghiệm tải file nhanh chóng và hiệu quả.

4 Protocol

4.1 TCP trong kết nối Peer-to-Peer



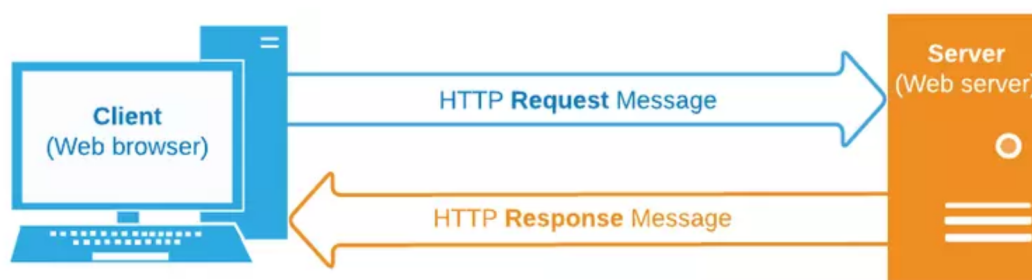
Hình 1: Header TCP trong kết nối Peer-to-Peer

Header của giao thức TCP:

- Port nguồn và Port đích: Mỗi gói dữ liệu TCP chứa hai trường này để chỉ định cổng nguồn và cổng đích. Chúng giúp định rõ quy trình nguồn và đích trên máy tính và xác định ứng dụng nào trên máy tính đích nên nhận gói dữ liệu.
- Số xác định (Sequence Number): Trường này là một số nguyên 32 bit, thường được sử dụng để xác định vị trí của dữ liệu trong luồng. Nó cho phép gói dữ liệu được sắp xếp lại nếu chúng đến không theo thứ tự.
- Số xác nhận (Acknowledgment Number): Trường này chứa số xác định của gói dữ liệu tiếp theo mà máy tính đích mong đợi. Nó giúp đảm bảo rằng dữ liệu đã gửi đã đến đúng cách.
- Chiều dài header: Trường này cho biết độ dài của header TCP. Nó cần thiết để xác định vị trí bắt đầu của dữ liệu trong gói.
 - Cờ (Flags): Cờ bao gồm nhiều bit để điều khiển các khía cạnh của kết nối TCP. Các cờ quan trọng bao gồm:
 - URG (Urgent): Được thiết lập khi có dữ liệu khẩn cấp trong gói.

- ACK (Acknowledgment): Được thiết lập khi gói chứa số xác nhận.
- PSH (Push): Cho biết rằng dữ liệu nên được đẩy xuống lớp ứng dụng ngay lập tức.
- RST (Reset): Được sử dụng để kết thúc một kết nối một cách đột ngột.
- SYN (Synchronize): Sử dụng trong quá trình thiết lập kết nối.
- FIN (Finish): Được sử dụng để kết thúc kết nối.
- Cửa sổ (Window): Trường này cho biết kích thước của cửa sổ trượt, mà máy tính nguồn đang cho phép máy tính đích gửi dữ liệu mà không cần xác nhận.
- Kiểm tra của dữ liệu (Checksum): Trường này chứa giá trị kiểm tra để kiểm tra tính toàn vẹn của dữ liệu TCP và header. Nó đảm bảo rằng gói dữ liệu không bị hỏng trong quá trình truyền tải.
- Ưu tiên (Urgent Pointer): Trường này chỉ ra vị trí của dữ liệu khẩn cấp nếu cờ URG được thiết lập.
- Lựa chọn (Options): Trường này chứa thông tin bổ sung, như có thể chứa thời gian trễ, mảng máy tính (MSS), hoặc các thông tin đặc biệt khác.
- Padding: Đây là một trường để đảm bảo rằng header có kích thước là một số nguyên bội của 32 bit, nếu có tùy chọn mà làm cho header không phải là bội số của 32 bit.

4.2 HTTP trong kết nối Client-Server



Hình 2: HTTP trong kết nối Client-Server

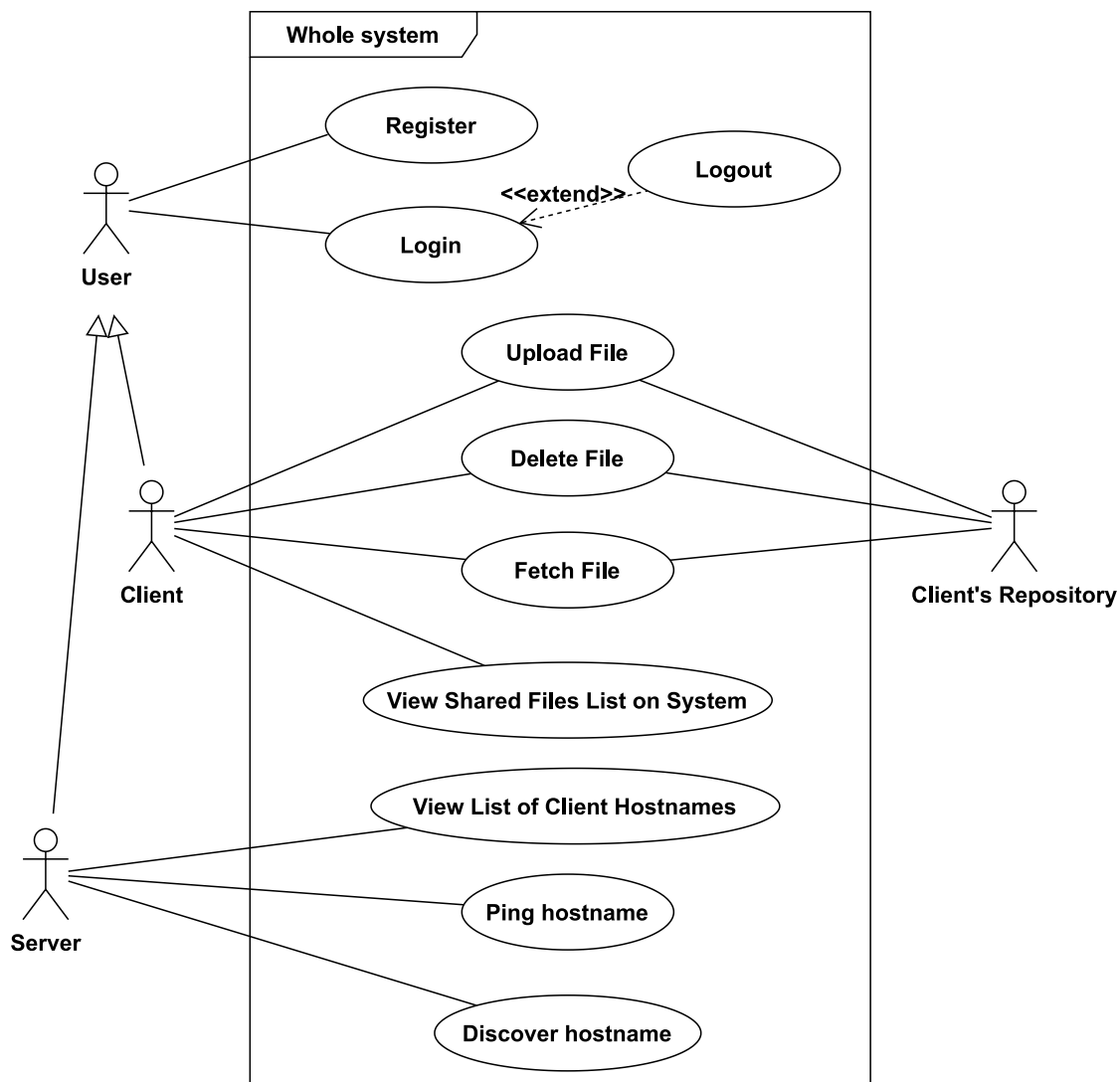
Phân tích HTTP:

- Giao thức dựa trên văn bản: HTTP là một giao thức dựa trên văn bản, có nghĩa là các yêu cầu và phản hồi HTTP được truyền tải dưới dạng văn bản đơn giản. Điều này làm cho việc đọc và hiểu dễ dàng, nhưng nó cũng khiến cho HTTP không thích hợp cho việc truyền tải dữ liệu lớn hoặc nhạy cảm.

- Mô hình giao tiếp yêu cầu/phản hồi: HTTP hoạt động dựa trên mô hình giao tiếp yêu cầu/phản hồi. Một máy khách gửi yêu cầu HTTP đến một máy chủ, sau đó máy chủ trả lời bằng một phản hồi HTTP. Yêu cầu và phản hồi này chứa các thông tin quan trọng như method, URL, trường header, và dữ liệu.
- Method (Phương thức): Phương thức HTTP định nghĩa hành động mà máy chủ nên thực hiện. Các phương thức phổ biến bao gồm:
 - GET: Lấy thông tin từ máy chủ.
 - POST: Gửi dữ liệu lên máy chủ để xử lý.
 - PUT: Cập nhật hoặc tạo mới tài nguyên trên máy chủ.
 - DELETE: Xóa tài nguyên trên máy chủ.
 - HEAD: Giống như GET, nhưng không trả về dữ liệu, chỉ trả về tiêu đề.
 - URL (Uniform Resource Locator): URL định danh tài nguyên trên internet. Nó bao gồm một phần giao thức (ví dụ: "http://"), tên miền (ví dụ: "www.example.com"), và đường dẫn (ví dụ: "/page.html") đến tài nguyên.
- Trường Header (Headers): Các trường header chứa thông tin bổ sung về yêu cầu hoặc phản hồi, bao gồm thông tin về máy khách, máy chủ, kiểu nội dung, ngôn ngữ, phiên bản giao thức, và nhiều thông tin khác.
- Phản hồi (Response): Phản hồi HTTP chứa các thông tin về kết quả của yêu cầu, bao gồm mã trạng thái, tiêu đề, và dữ liệu phản hồi.
- Stateless (Không lưu trạng thái): HTTP là một giao thức không lưu trạng thái, điều này có nghĩa rằng mỗi yêu cầu được xử lý độc lập và không có thông tin về trạng thái trước đó của giao dịch. Điều này có thể gây khó khăn trong việc quản lý phiên và trạng thái của người dùng.
- Phiên bản giao thức: HTTP có nhiều phiên bản, trong đó phiên bản phổ biến nhất là HTTP/1.1. Các phiên bản cũ hơn bao gồm HTTP/1.0 và HTTP/0.9.
- Mã trạng thái (Status Codes): Mã trạng thái HTTP được sử dụng để chỉ rõ kết quả của yêu cầu. Ví dụ, mã 200 nghĩa là "OK," mã 404 nghĩa là "Không tìm thấy," và mã 500 nghĩa là "Lỗi máy chủ nội bộ."

5 Diagram

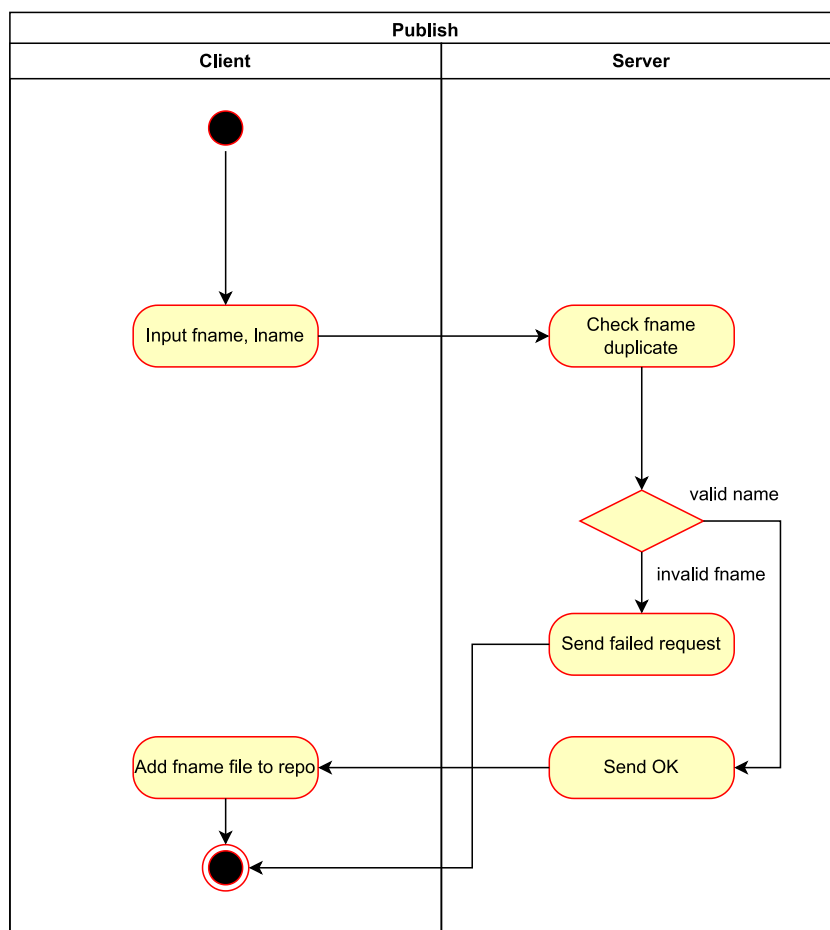
5.1 Use Case Diagram



Hình 3: Usecase diagram for whole system

5.2 Activity Diagram

5.2.1 publish lname fname



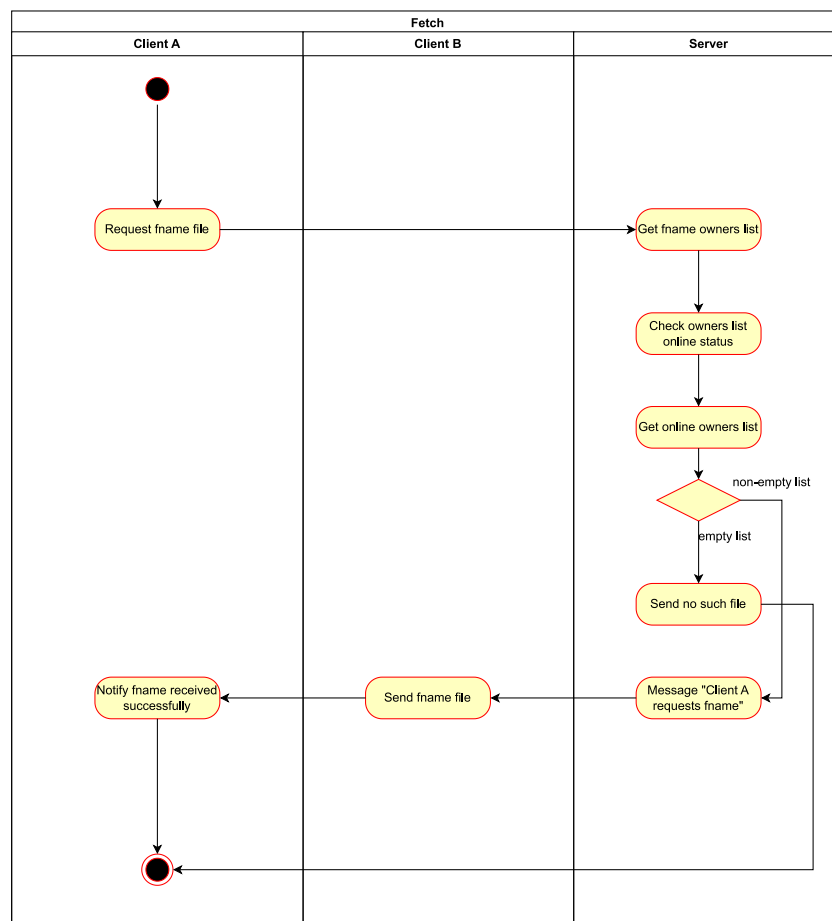
Hình 4: Activity diagram for publish

Mô tả: Đây là chức năng cho phép client tải file ở máy cục bộ (được lưu với tên lname) lên repository của client đó (được lưu với tên fname) và gửi đến server thông tin của file đó.

Đầu tiên, client nhập giá trị cho fname, lname và gửi lên server.

Tiếp đó, server kiểm tra giá trị của fname có hợp lệ hay không. Nếu tên hợp lệ, file được thêm thành công vào repository của client với tên fname và server sẽ nắm giữ thông tin về file đó. Nếu tên không hợp lệ (đã bị trùng) server thông báo yêu cầu thất bại và yêu cầu đổi tên khác.

5.2.2 fetch fname



Hình 5: Activity diagram for fetch

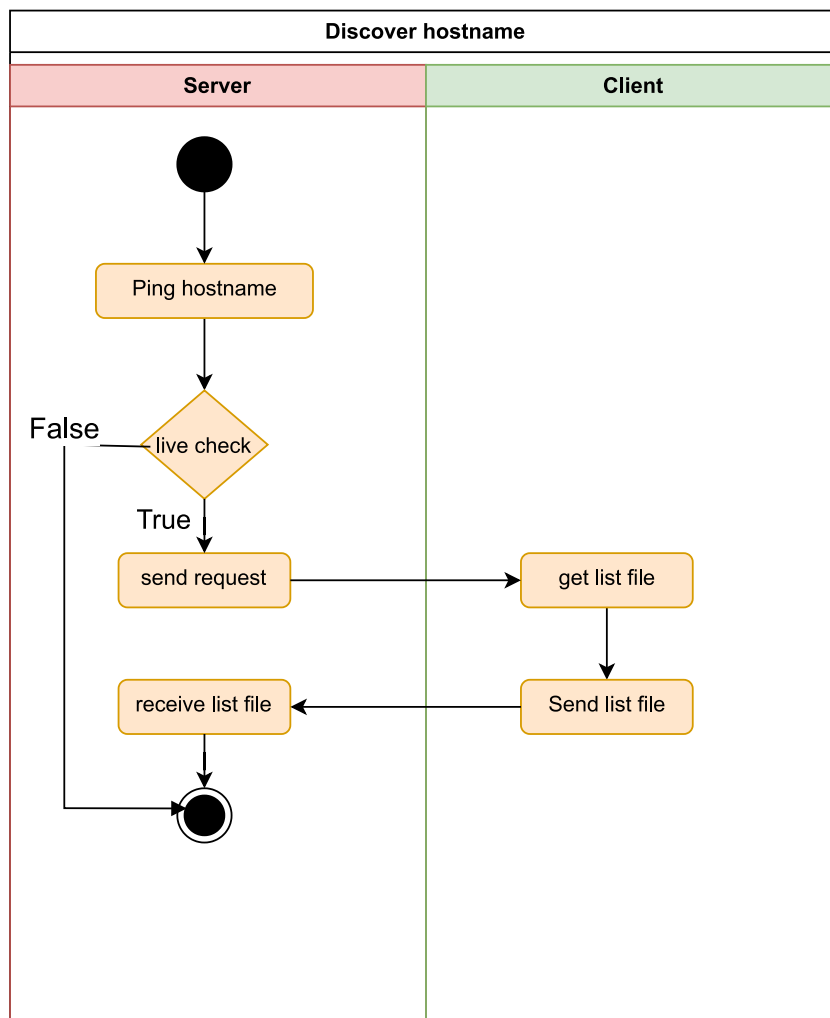
Mô tả: Đây là chức năng cho phép client gửi yêu cầu lên server để tải một file từ client khác và thêm file đó vào repository của mình.

Đầu tiên, client A gửi yêu cầu cho server về tên file (fname) mình muốn tải.

Tiếp đó, server trả về danh sách các client có chứa file đó và kiểm tra trong danh sách đó có client nào đang hoạt động.

Sau khi nhận được danh sách, client A sẽ tạo kết nối tới client B nào đó để truyền yêu cầu fetch một file cụ thể ở client B. Khi client A nhận được file thì thêm file vào repository, gửi thông báo nhận thành công đến server để server update thông tin file hiện tại trong repository của A

5.2.3 discover hostname

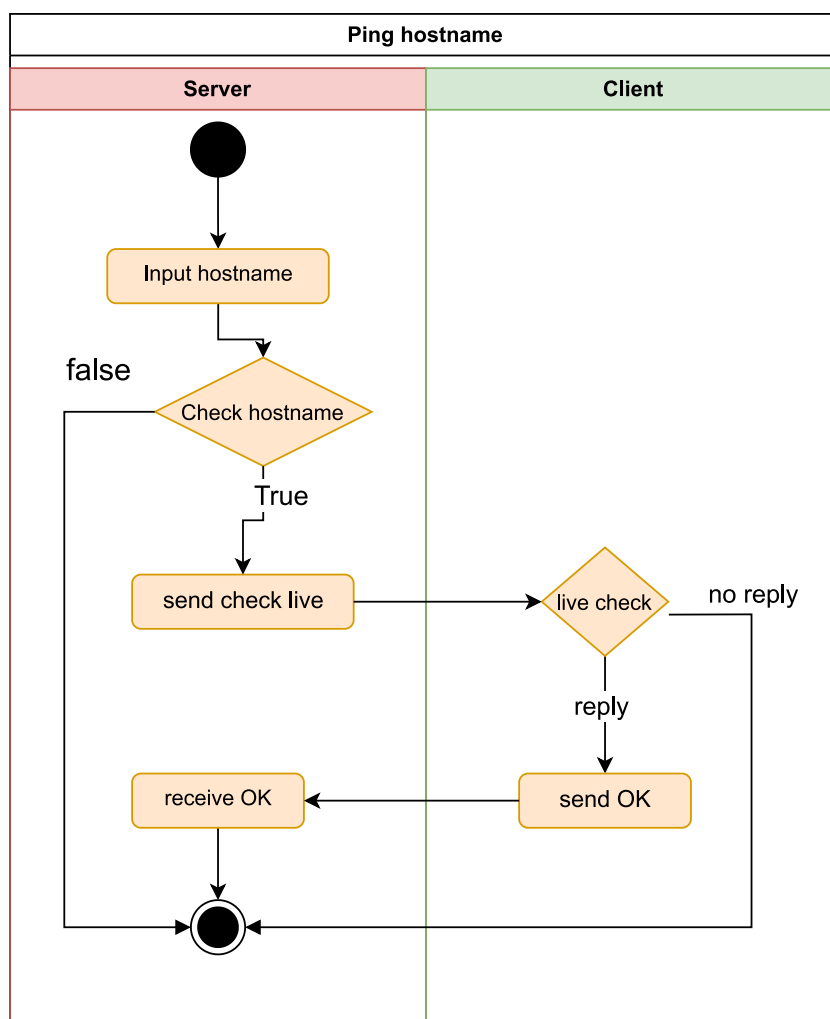


Hình 6: Activity diagram for discover

Mô tả: Đây là chức năng cho phép server xem danh sách file chia sẻ trong repository của một client bất kỳ.

Đầu tiên, server kiểm tra trạng thái hoạt động của client. Nếu trạng thái của client là đang hoạt động thì trả về danh sách các file trong repository của client đó và ngược lại, câu lệnh thực thi thất bại.

5.2.4 ping hostname



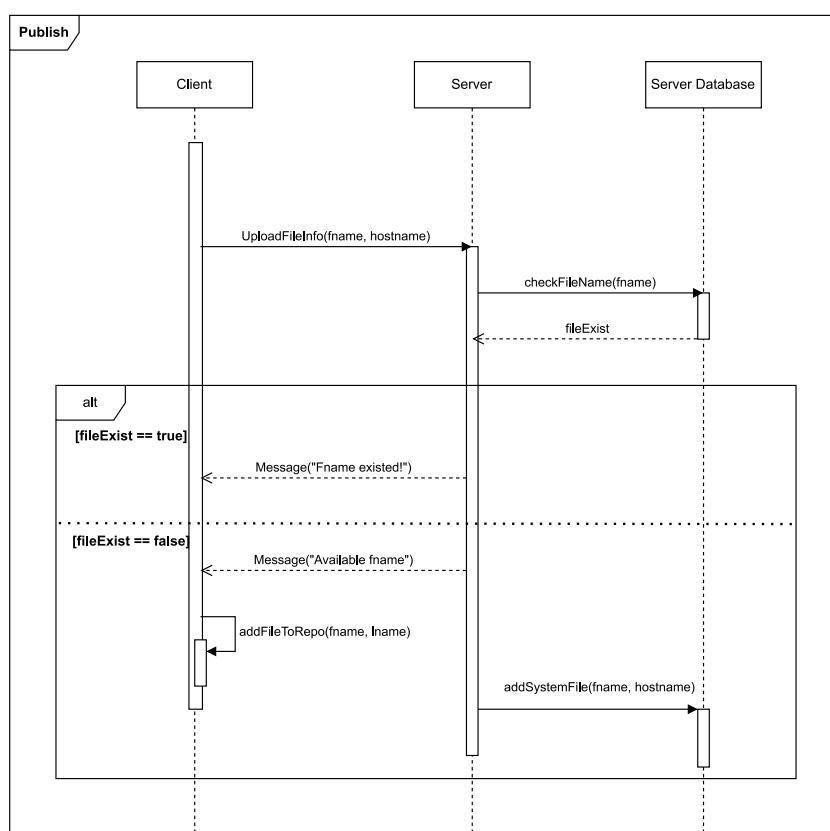
Hình 7: Activity diagram for ping

Mô tả: Đây là chức năng cho phép server kiểm tra trạng thái của một client có đang hoạt động hay không.

Server sẽ gửi thông điệp đến client. Nếu client phản hồi lại thông điệp đó thì nó đang hoạt động và ngược lại, nếu client không phản hồi có nghĩa là nó không hoạt động.

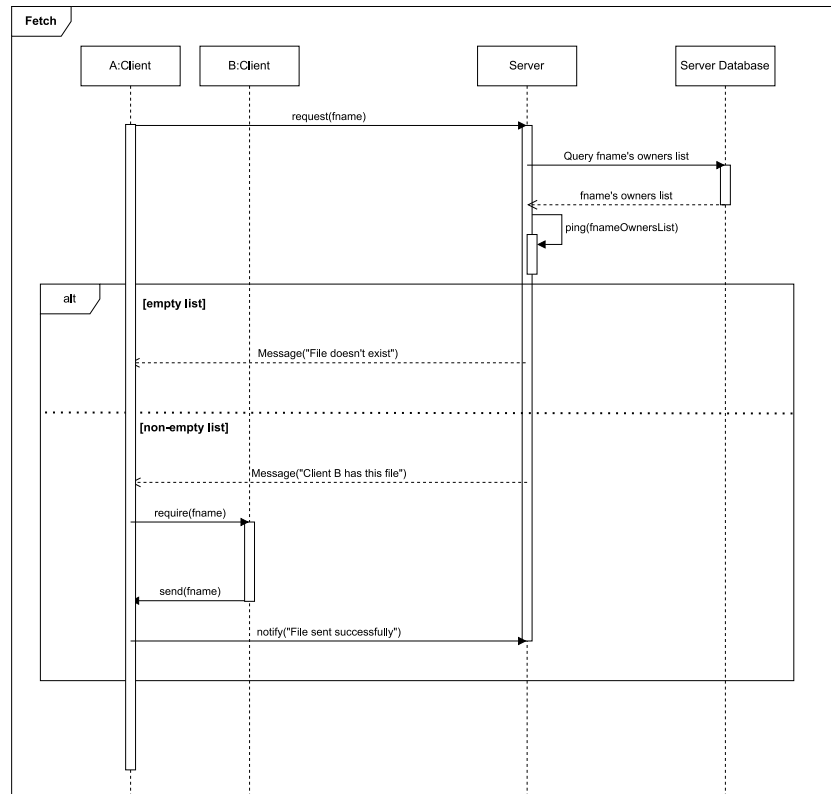
5.3 Sequence Diagram

5.3.1 publish lname fname



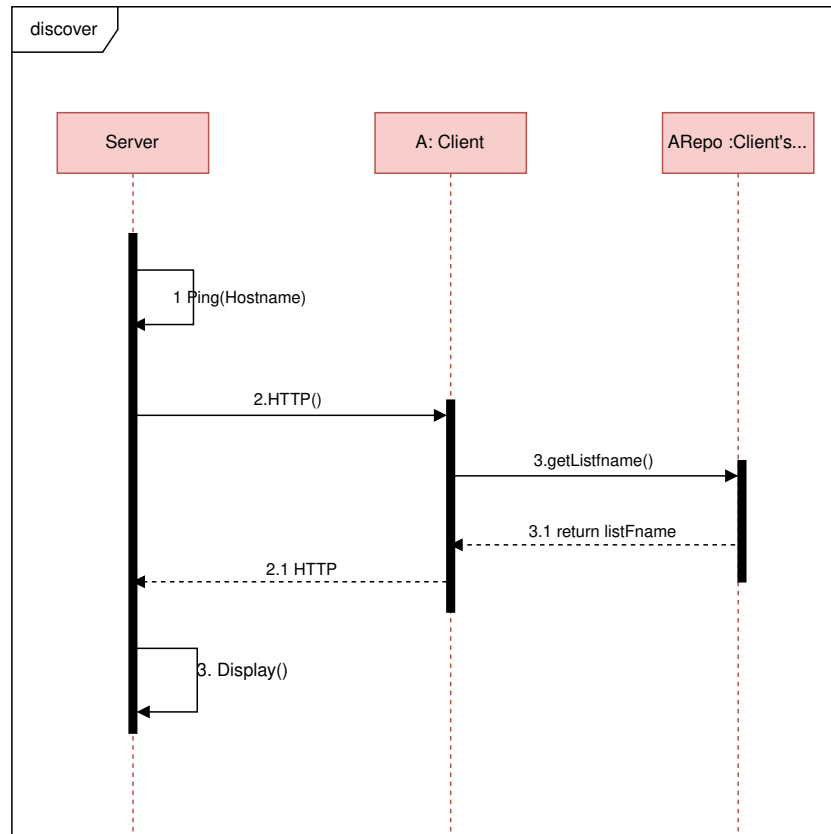
Hình 8: Sequence diagram for publish

5.3.2 fetch fname



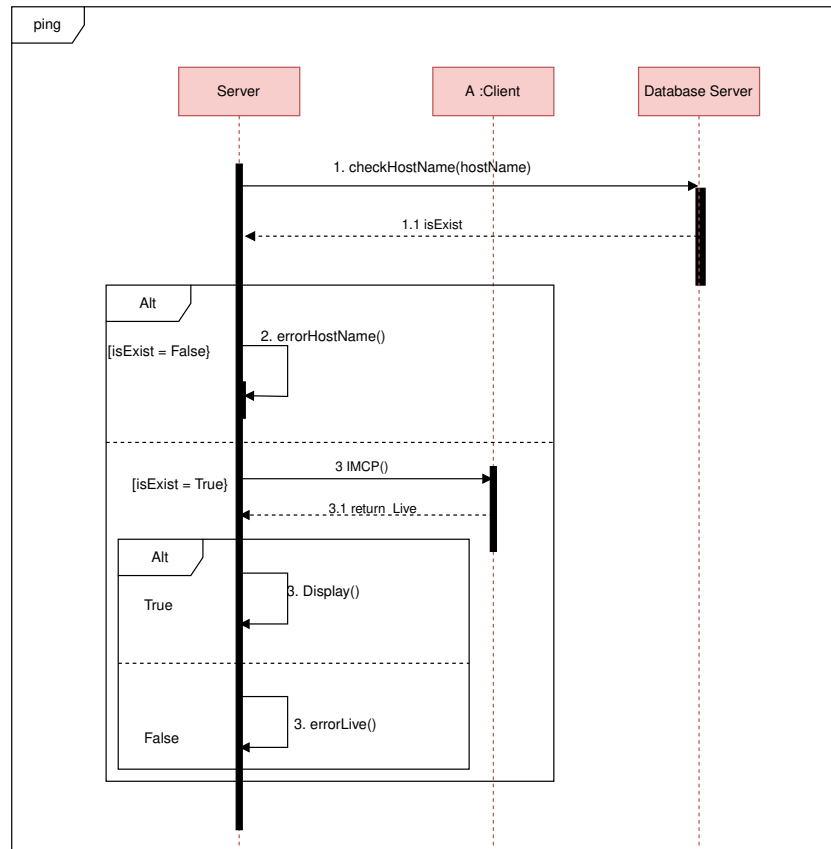
Hình 9: Sequence diagram for fetch

5.3.3 discover hostname



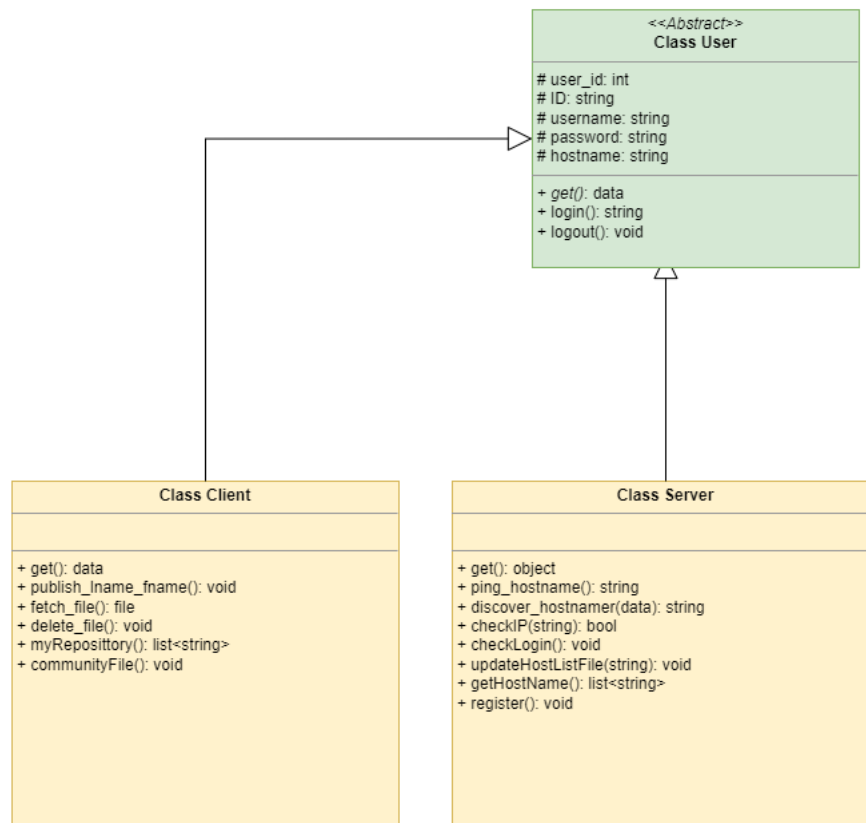
Hình 10: Sequence diagram for discover

5.3.4 ping hostname



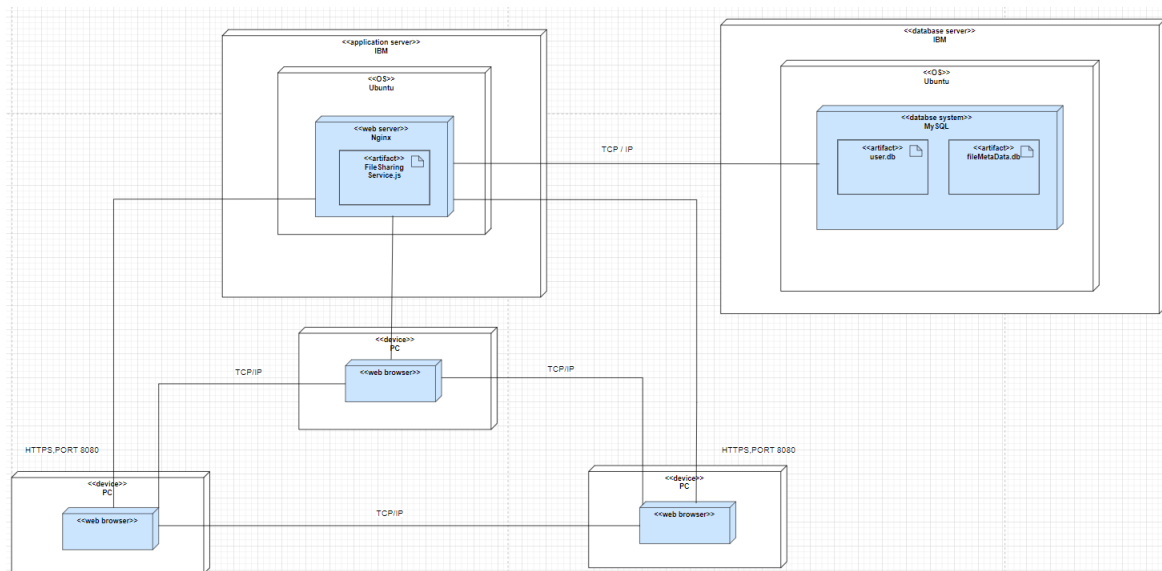
Hình 11: Sequence diagram for ping

5.4 Class Diagram



- Client đảm bảo nhiệm chức năng của quá trình tải file từ 1 client khác từ hostname của server gửi về.
- Server có chức năng xử lý xác thực, kiểm tra hoạt động của các client và gửi hostname của client khác cho client muốn tải file.

5.5 Deployment Diagram

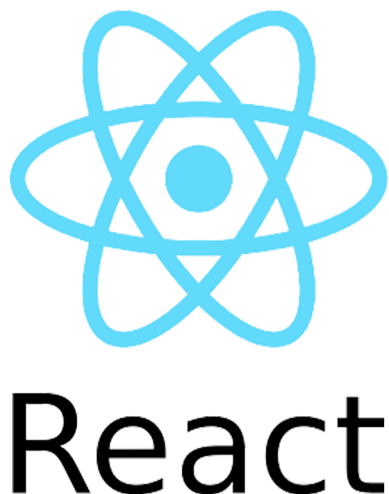


Hình 12: Deployment diagram

6 Technology and Architecture

6.1 Technology

6.1.1 ReactJS



ReactJS được phát triển bởi Facebook và được giới thiệu lần đầu tiên vào năm 2011. Ban đầu, ReactJS được phát triển để xây dựng giao diện người dùng trên trang web Facebook, nhằm cải thiện tốc độ và hiệu suất của ứng dụng web.

Tuy nhiên, ReactJS không được công bố cho cộng đồng phát triển cho đến năm 2013, khi Facebook công bố mã nguồn mở của nó và giới thiệu cho cộng đồng lập trình viên. Từ đó, ReactJS nhanh chóng trở thành một trong những thư viện phổ biến nhất để phát triển các ứng dụng web động.

Sau đó, vào năm 2015, Facebook giới thiệu phiên bản React Native, một framework phát triển ứng dụng di động sử dụng ReactJS. React Native cho phép các nhà phát triển xây dựng các ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn, tương tự như ReactJS trên web.

Hiện nay, ReactJS đã trở thành một trong những thư viện phát triển web phổ biến nhất, được sử dụng rộng rãi bởi các công ty lớn và nhỏ trên toàn thế giới. Facebook cũng tiếp tục đầu tư phát triển và nâng cấp ReactJS để đáp ứng nhu cầu của cộng đồng phát triển.

Những lợi ích tuyệt vời mà ReactJS mang lại cho lập trình viên

ReactJS mang đến nhiều lợi ích cho việc phát triển ứng dụng web, bao gồm:

- **Hiệu suất cao:** ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM cho phép ReactJS cập nhật các thay đổi trên trang web một cách nhanh chóng và hiệu quả hơn so với cách truyền thống, giúp tăng tốc độ và hiệu suất của ứng dụng.
- **Tái sử dụng:** ReactJS cho phép tái sử dụng các thành phần UI, giúp giảm thiểu thời gian và chi phí phát triển. Các thành phần UI có thể được sử dụng lại trong nhiều phần khác nhau của ứng dụng, giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.
- **Dễ dàng quản lý trạng thái:** ReactJS giúp quản lý trạng thái của ứng dụng một cách dễ dàng. Sử dụng State và Props, ReactJS cho phép các nhà phát triển quản lý trạng thái của các thành phần UI một cách chính xác và dễ dàng.
- **Hỗ trợ tốt cho SEO:** ReactJS cho phép các nhà phát triển xây dựng ứng dụng web với khả năng tương thích tốt với SEO. Với sự hỗ trợ của các thư viện như React Helmet, ReactJS cho phép các nhà phát triển tùy chỉnh và quản lý các phần tử meta và title cho từng trang web.
- **Hỗ trợ đa nền tảng:** ReactJS không chỉ được sử dụng để phát triển các ứng dụng web, mà còn được sử dụng để phát triển các ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.

Các tính năng nổi bật của ReactJS

Các tính năng nổi bật của ReactJS

- **Components:** ReactJS cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.
- **Virtual DOM:** ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.
- **JSX:** JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.

- **State và Props:** ReactJS cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
- **Hỗ trợ tốt cho SEO:** ReactJS hỗ trợ tốt cho việc tối ưu hóa SEO. Với các thư viện như React Helmet, các nhà phát triển có thể quản lý các phần tử meta và title cho từng trang web, giúp tăng khả năng tìm kiếm và tăng cường trải nghiệm người dùng.
- **Hỗ trợ đa nền tảng:** ReactJS không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.
- **Redux:** Redux là một thư viện quản lý trạng thái cho các ứng dụng ReactJS. Nó giúp quản lý trạng thái của ứng dụng một cách chính xác và dễ dàng, đồng thời giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.

6.1.2 TailwindCSS



Với nền tảng web hiện đại bây giờ có một yếu tố rất quan trọng ảnh hưởng trực tiếp đến sự thành công của trang web đấy chính là giao diện. Trong quá trình phát triển chúng ta đã quá quen với những bộ phát triển giao diện như là Bootstrap,

Foundation, Bulma. Đây hầu như là những bộ giao diện gần như là hoàn chỉnh nhất rồi. Vậy Tailwind sinh ra để làm gì???

Tailwind css là một utility-first CSS framework nó hỗ trợ phát triển xây dựng nhanh chóng giao diện người dùng, nó cũng có điểm chung giống như Bootstrap và điểm làm nó nổi bật hơn cả đó là chúng ta có thể tùy biến phát triển css theo cách mà chúng ta định nghĩa ra.

Ưu điểm của Tailwind CSS

Ưu điểm nhìn thấy rõ nhất của Tailwind CSS đây là:

- Người sử dụng có thể chẳng phải viết đến 1 dòng css nào mà vẫn có giao diện tùy biến theo mong muốn.
- Style, màu sắc, font chữ hiện đại, phù hợp với phong cách web hiện đại
- Cách đặt tên class dễ hiểu, 1 class đại diện cho 1 thuộc tính.
- Tailwind CSS có gần như đủ gần 85 phần trăm thuộc tính css.
- Sử dụng Flex nên rất dễ chia Layout
- Dễ cài đặt, dễ sử dụng, document của Tailwind rất dễ hiểu. Tailwind CSS phù hợp cho các dự án nhỏ, người dùng tùy biến nhiều, cần làm nhanh giao diện. Trong khi nếu bạn Bootstrap mà không tùy biến gì thì trong web của bạn sẽ đúng đậm chất Bootstrap. còn với Tailwind thì khi mỗi người dùng sẽ ra mỗi giao diện khác nhau mà không hề đụng hàng.

Nhược điểm của Tailwind CSS

- Khi sử dụng tailwind thì bạn đang phải sử dụng số class cực kì nhiều, số class sẽ tương ứng với số thuộc tính mà bạn muốn cài đặt
- Khi dùng font-size hoặc màu sắc vẫn đang còn phải custom lại bằng css riêng.
- Chưa có những bộ mixin khi muốn set nhiều thuộc tính cần thiết.

6.1.3 NodeJS



NodeJS là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009 và phiên bản cuối cùng là v0.10.36. Định nghĩa NodeJs bởi tài liệu chính thức như sau:

Nền tảng Node.js dựa trên Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

NodeJs là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành Window, Linux...

Node.js cũng cung cấp cho chúng ta các module Javascript đa dạng, có thể đơn giản hóa sự phát triển của các ứng dụng web sử dụng Node.js với các phần mở rộng.

Đặc điểm của Node.js

Dưới đây là vài đặc điểm quan trọng biến Node.js trở thành sự lựa chọn hàng đầu trong phát triển phần mềm:

- **Không đồng bộ và Phát sinh sự kiện (Event Driven):** Tất cả các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dữ liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận đưa phản hồi từ các API gọi trước đó.

- **Chạy rất nhanh:** Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.
- **Các tiến trình đơn giản nhưng hiệu năng cao:** Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lập. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.
- **Không đệm:** Ứng dụng Node.js không lưu trữ các dữ liệu buffer.
- **Có giấy phép:** Node.js được phát hành dựa vào MIT License.

Với Node.js, bạn phải làm mọi thứ

Node.js chỉ là một môi trường – điều này có nghĩa bạn tự phải làm mọi thứ. Sẽ chẳng có bất kỳ máy chủ mặc định nào cả!!! Một đoạn script xử lý tất cả các kết nối với Client. Điều này làm giảm đáng kể số lượng tài nguyên được sử dụng trong ứng dụng.

NodeJs được sử dụng ở đâu

Dưới đây là các lĩnh vực mà Node.js được sử dụng như là một sự lựa chọn hoàn hảo:

- Các ứng dụng về I/O
- Các ứng dụng về luồng dữ liệu
- Các ứng dụng về dữ liệu hướng đến thời gian thực
- Các ứng dụng dựa vào JSON APIs
- Các ứng dụng Single Page Application

Nodejs không nên sử dụng ở đâu: Nó không nên sử dụng trong các ứng dụng đòi hỏi về CPU.

6.1.4 ExpressJS



ExpressJS là một framework ứng dụng web có mã nguồn mở và miễn phí được xây dựng trên nền tảng Node.js. ExpressJS được sử dụng để thiết kế và phát triển các ứng dụng web một cách nhanh chóng. Để hiểu ExpressJS, người dùng chỉ cần phải biết JavaScript, do đó nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn đối với các lập trình viên và nhà phát triển đã thành thạo JavaScript trước đó.

Vì ExpressJS là một framework của Node.js nên hầu hết các mã đã được viết sẵn cho các lập trình viên làm việc. Bạn có thể tạo các ứng dụng web cho một trang, nhiều trang hoặc kết hợp lại bằng cách sử dụng ExpressJS. framework này khá nhẹ, giúp tổ chức các ứng dụng web ở phía máy chủ thành một kiến trúc hoàn hảo hơn.

ExpressJS hỗ trợ nâng cao các chức năng của NodeJS. Nếu bạn không sử dụng ExpressJS, bạn phải thực hiện rất nhiều lập trình phức tạp để xây dựng một API hiệu quả. ExpressJS đã giúp cho việc lập trình trong NodeJS trở nên dễ dàng hơn rất nhiều.

ExpressJS được sử dụng để làm gì?

ExpressJS sẽ giúp bạn tổ chức kiến trúc back-end của mình, điều này cho phép viết một codebase back-end bảo trì tương đối dễ dàng.

Bởi vì ExpressJS hoạt động trên back-end, nên bạn có thể coi công nghệ này như một “bộ não đằng sau một trang web”. Ví dụ: ExpressJS có thể xác định cách những trang được định tuyến trên một trang web. Hơn nữa, một nhà phát triển có thể sử dụng ExpressJS để quản lý xác thực trên một trang web.

Dưới đây là một số trường hợp sử dụng của ExpressJS:

- Sử dụng cookie trên một trang web
- Triển khai xác thực
- Thêm thanh tìm kiếm vào một trang web

- Cung cấp các tệp tĩnh như hình ảnh

Các tính năng của ExpressJS:

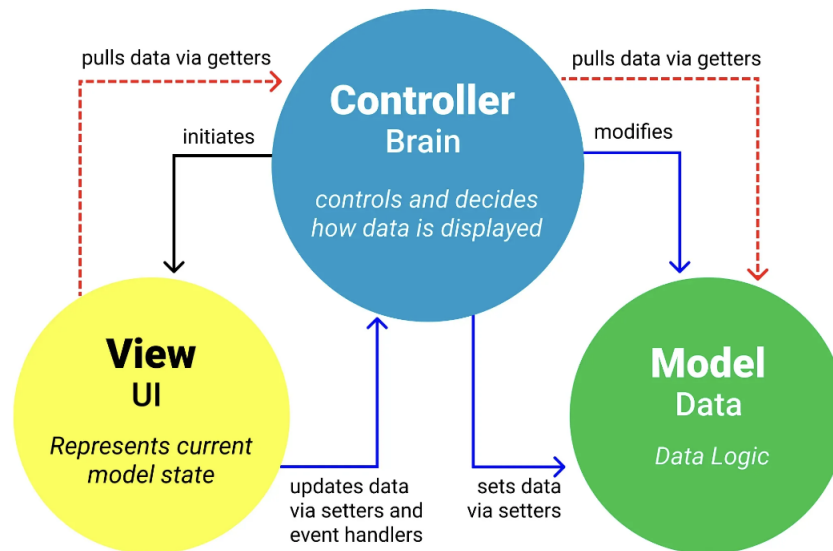
- **Phát triển máy chủ nhanh hơn:** ExpressJS cung cấp cho bạn nhiều tính năng phổ biến của Node.js dưới dạng hàm có thể dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này sẽ giúp rút ngắn thời gian để viết code.
- **Phần mềm trung gian:** Phần mềm trung gian là một phần trong chương trình cho phép truy cập vào cơ sở dữ liệu, xem xét yêu cầu của khách hàng và các phần mềm trung gian khác. Tính năng này chịu trách nhiệm chính cho việc tổ chức chức năng khác nhau của ExpressJS.
- **Định tuyến:** ExpressJS cung cấp một cơ chế định tuyến nâng cao giúp duy trì trạng thái của trang web.
- **Khuôn mẫu:** ExpressJS cung cấp các công cụ tạo khuôn mẫu cho phép các nhà phát triển tạo nội dung động trên các trang web bằng việc xây dựng các mẫu HTML ở phía máy chủ.
- **Gỡ lỗi:** Gỡ lỗi là yếu tố quan trọng để phát triển các ứng dụng web. ExpressJS giúp gỡ lỗi dễ dàng hơn bằng cách cung cấp một cơ chế có khả năng xác định chính xác phần ứng dụng web có lỗi.

Những lợi ích của ExpressJS

- Rất dễ học, chỉ cần bạn biết JavaScript, bạn sẽ không cần phải học một ngôn ngữ mới để học ExpressJS.
- Giúp cho việc phát triển back-end dễ dàng hơn nhiều khi sử dụng ExpressJS.
- Mã JavaScript được diễn giải thông qua Google V8 JavaScript Engine của Node.js. Do đó, mã sẽ được thực hiện một cách nhanh chóng và dễ dàng.
- ExpressJS rất đơn giản để tùy chỉnh và sử dụng theo nhu cầu.
- Cung cấp một module phần mềm trung gian linh hoạt và rất hữu ích để thực hiện các tác vụ bổ sung theo phản hồi và yêu cầu.
- Hỗ trợ phát triển ứng dụng theo mô hình MVC, đây là mô hình phổ biến cho việc lập trình web hiện nay.

6.2 Architecture

MVC Architecture Pattern



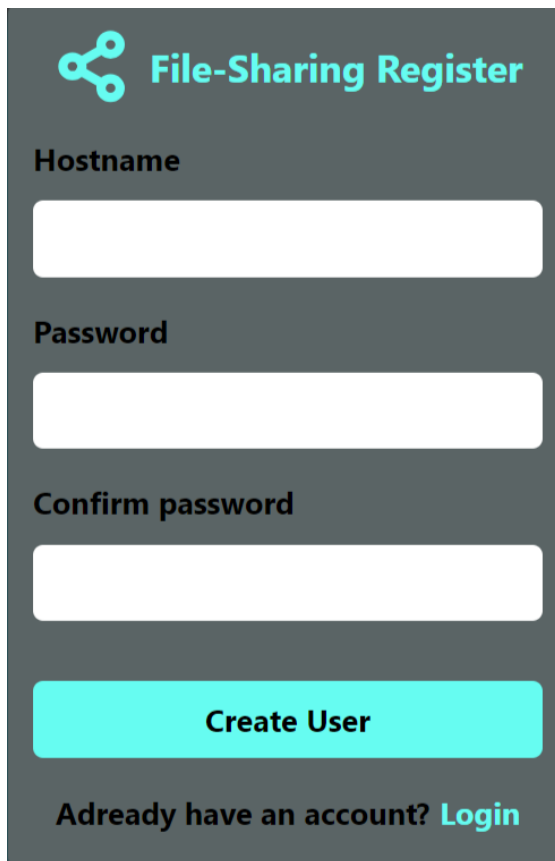
Mô hình MVC gồm 3 loại chính là thành phần bên trong không thể thiếu khi áp dụng mô hình này:

- **Model:** Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Chẳng hạn như biểu tượng hay là một nhân vật trong game.
- **View:** Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.
- **Controller:** Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các update tương ứng.

7 Kiểm tra và chạy thử ứng dụng

7.1 Tính năng đăng ký và đăng nhập

Khi người dùng sử dụng ứng dụng, nếu người dùng chưa có tài khoản, người dùng cần thực hiện việc đăng ký tạo tài khoản mới qua giao diện đăng ký tài khoản.

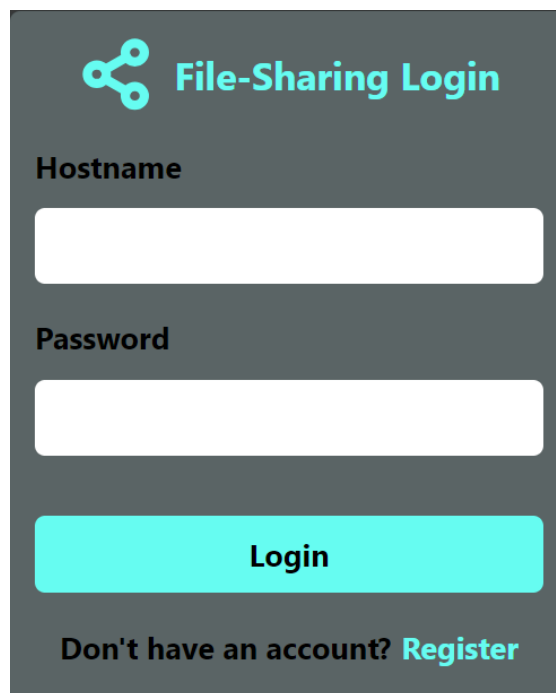


The image shows a registration form titled "File-Sharing Register". It has a dark gray background with white text and input fields. At the top left is a logo consisting of three connected circles. Below the title, there are three input fields labeled "Hostname", "Password", and "Confirm password". At the bottom, there is a red button labeled "Create User" and a link labeled "Login" preceded by the text "Adready have an account?".

Hình 13: Giao diện đăng ký tài khoản



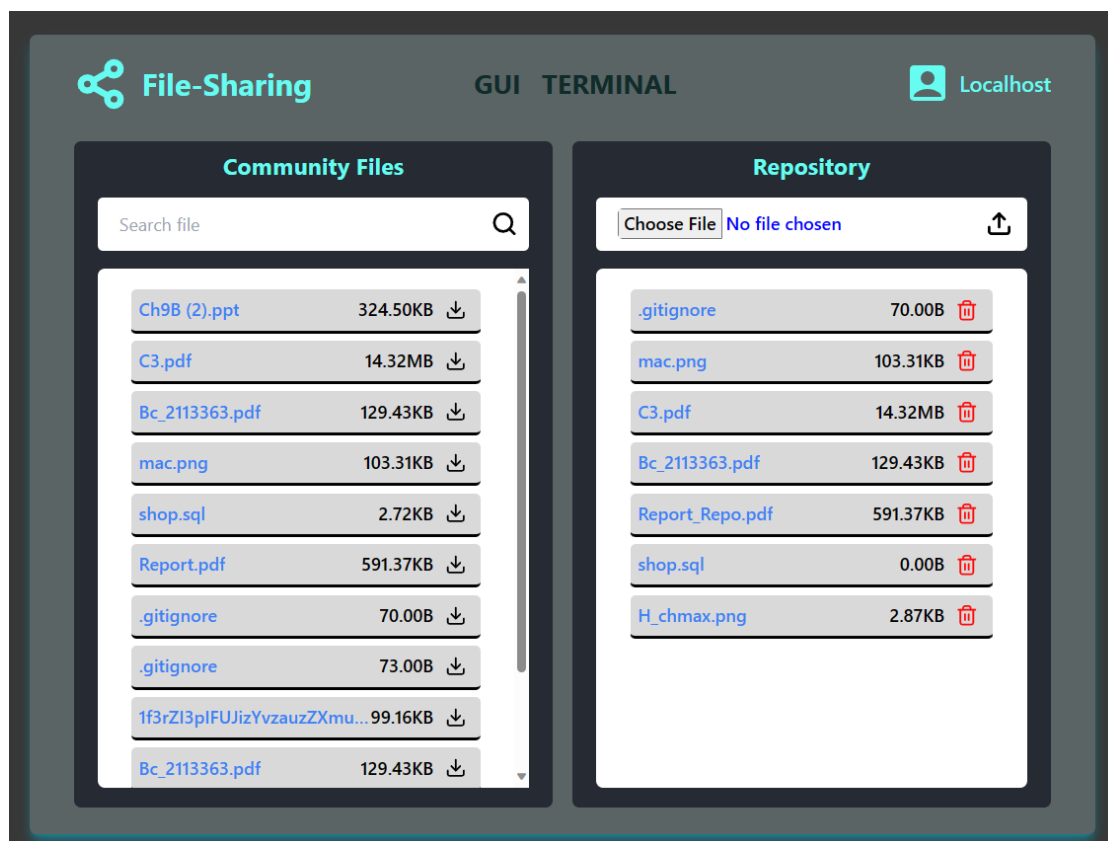
Sau khi tạo tài khoản, người dùng có thể đăng nhập vào ứng dụng thông qua giao diện đăng nhập.



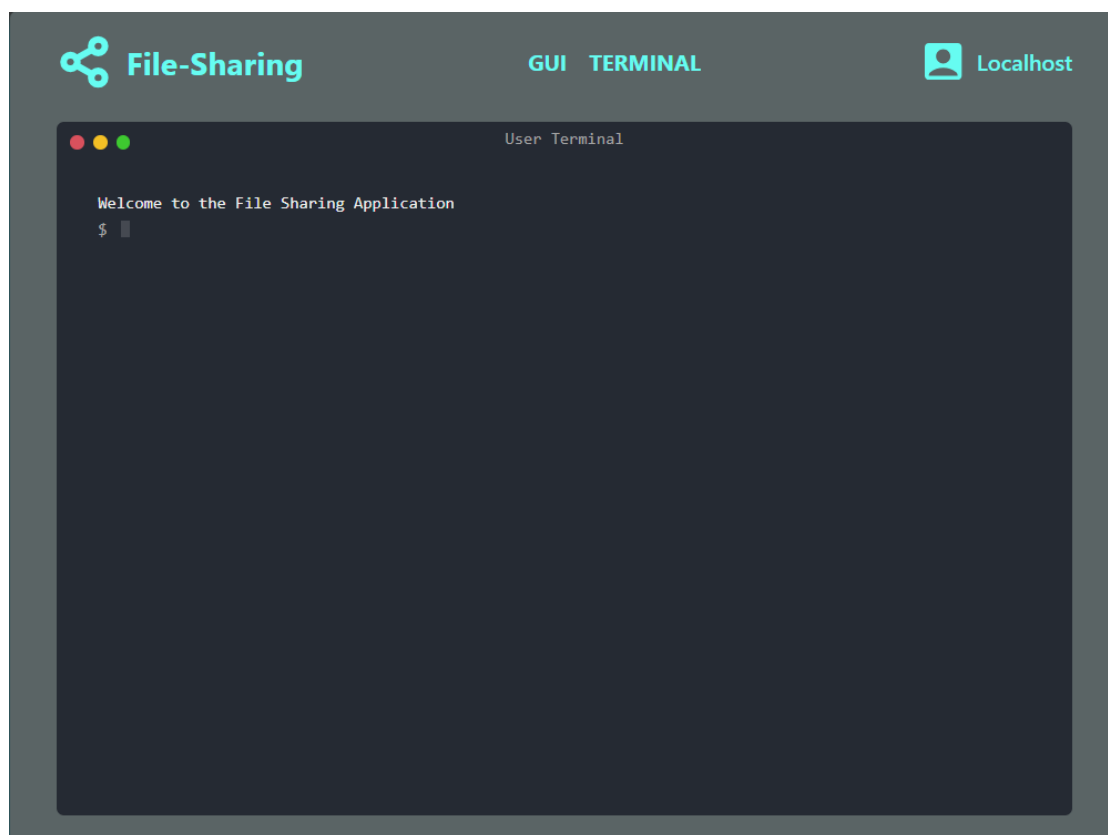
The image shows a login interface for a file-sharing application. It has a dark gray background. At the top left is a teal icon of three connected circles, followed by the text "File-Sharing Login" in teal. Below this are two white input fields: the first is labeled "Hostname" and the second is labeled "Password". Under the password field is a teal button with the text "Login". At the bottom, there is a line of text: "Don't have an account? Register", where "Register" is a teal link.

Hình 14: Giao diện đăng nhập

Sau khi đăng nhập, người dùng có thể truy cập và sử dụng các chức năng của ứng dụng. Có 2 dạng giao diện hiển thị cho người dùng, đó là Giao diện đồ họa (Graphical User Interface) và Giao diện dòng lệnh (Command Line Interface).



Hình 15: Giao diện đồ họa (Graphical User Interface)

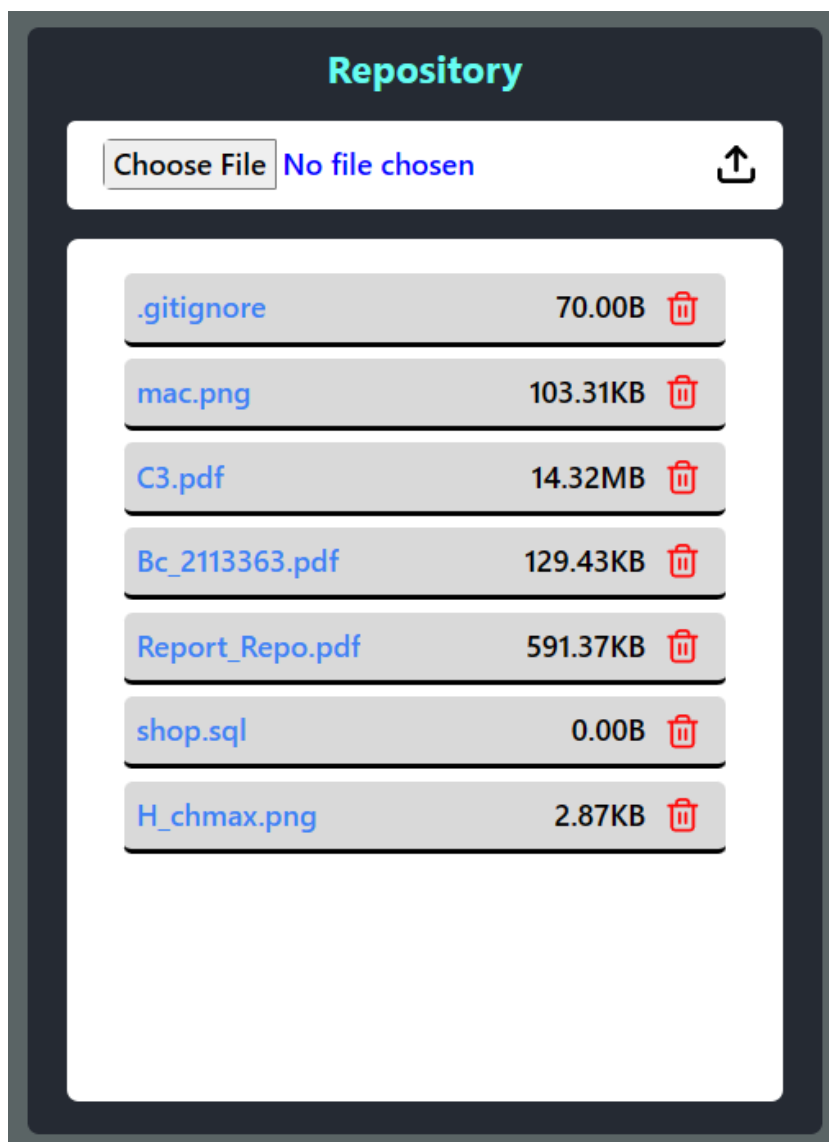


Hình 16: Giao diện dòng lệnh (Command Line Interface)

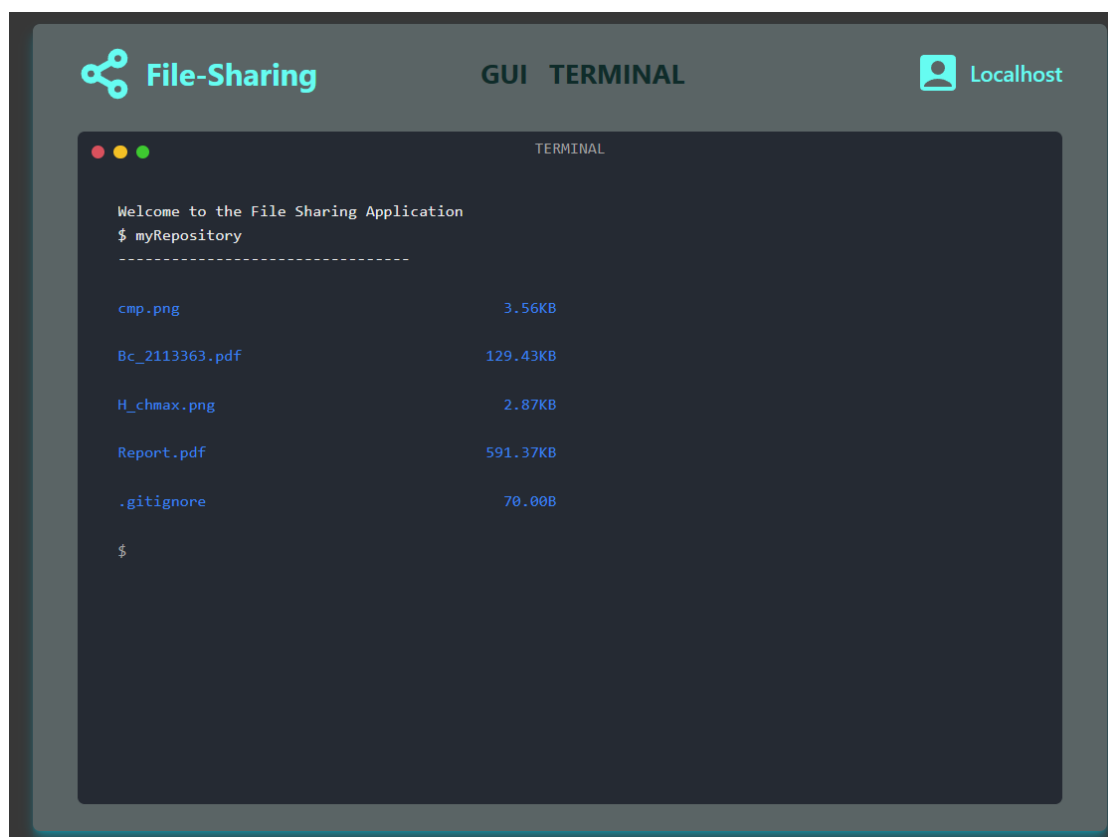
7.2 Tính năng của client

7.2.1 Tính năng xem file trong repository của client

Ứng dụng cho phép người dùng quản lý các tệp tin trong local repository thông qua cửa sổ Repository trong GUI hoặc lệnh myRepository trong CLI.



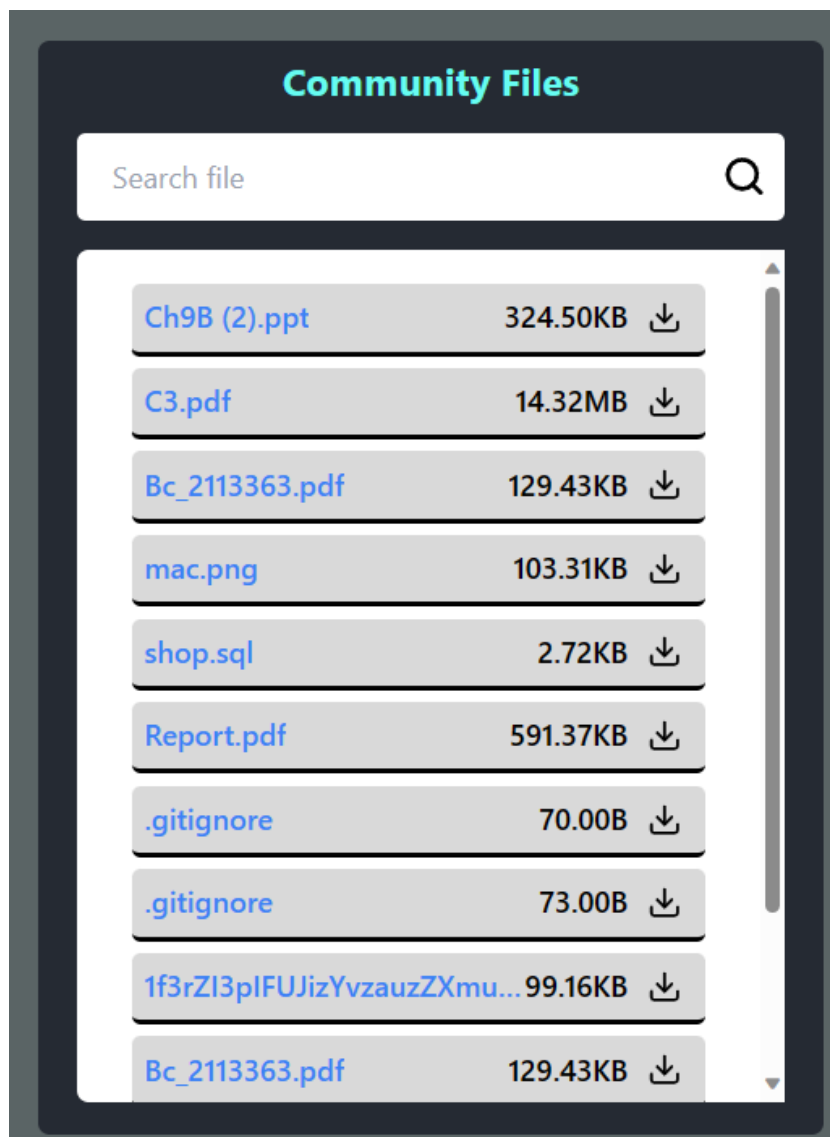
Hình 17: Tính năng quản lý local repository trong GUI



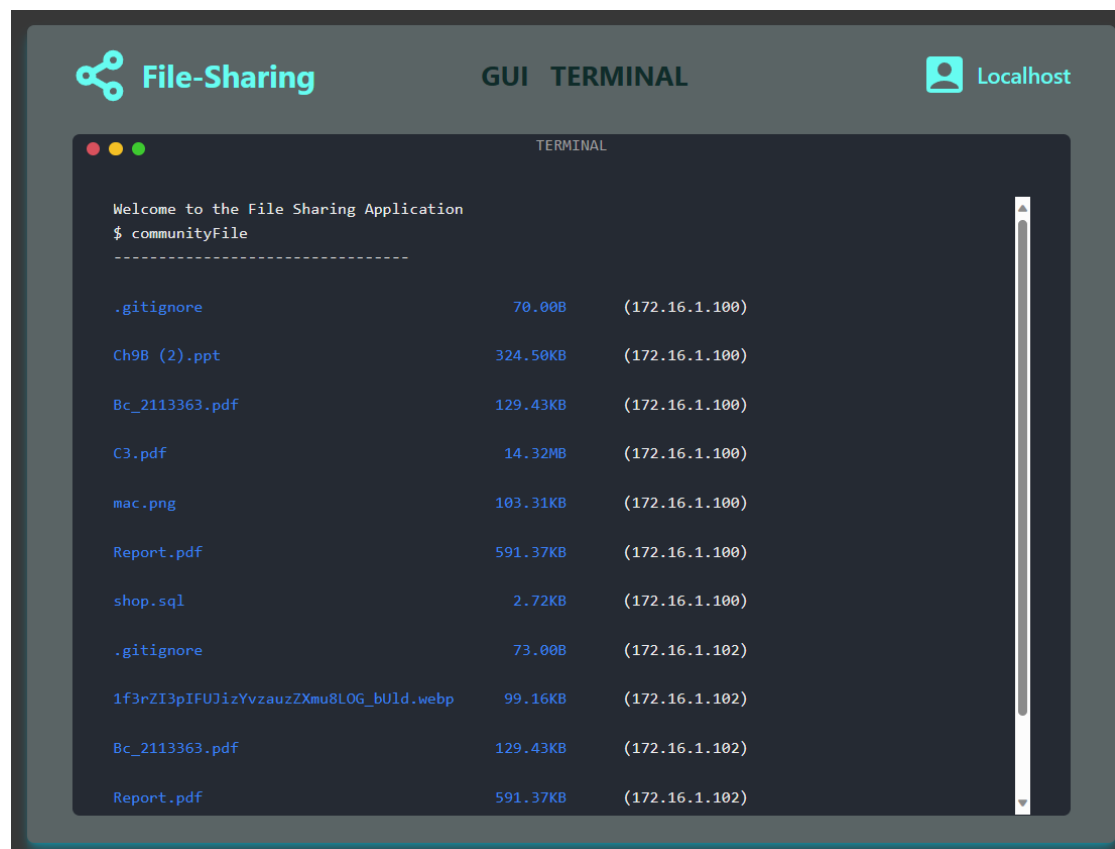
Hình 18: Tính năng quản lý local repository trong CLI

7.2.2 Tính năng xem danh sách các file được chia sẻ

Ứng dụng cho phép người dùng xem các tệp tin được chia sẻ trong hệ thống thông qua cửa sổ Community Files trong GUI và lệnh communityFile trong CLI.



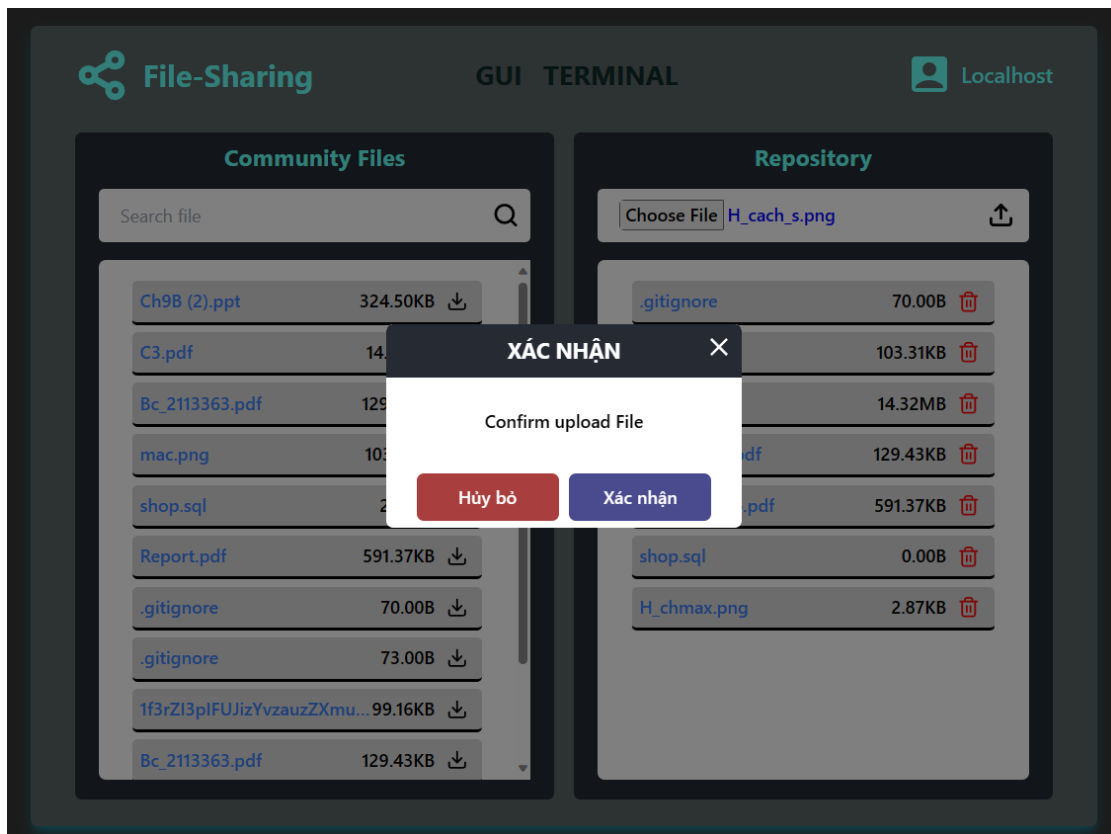
Hình 19: Tính năng xem tệp tin chia sẻ trong GUI



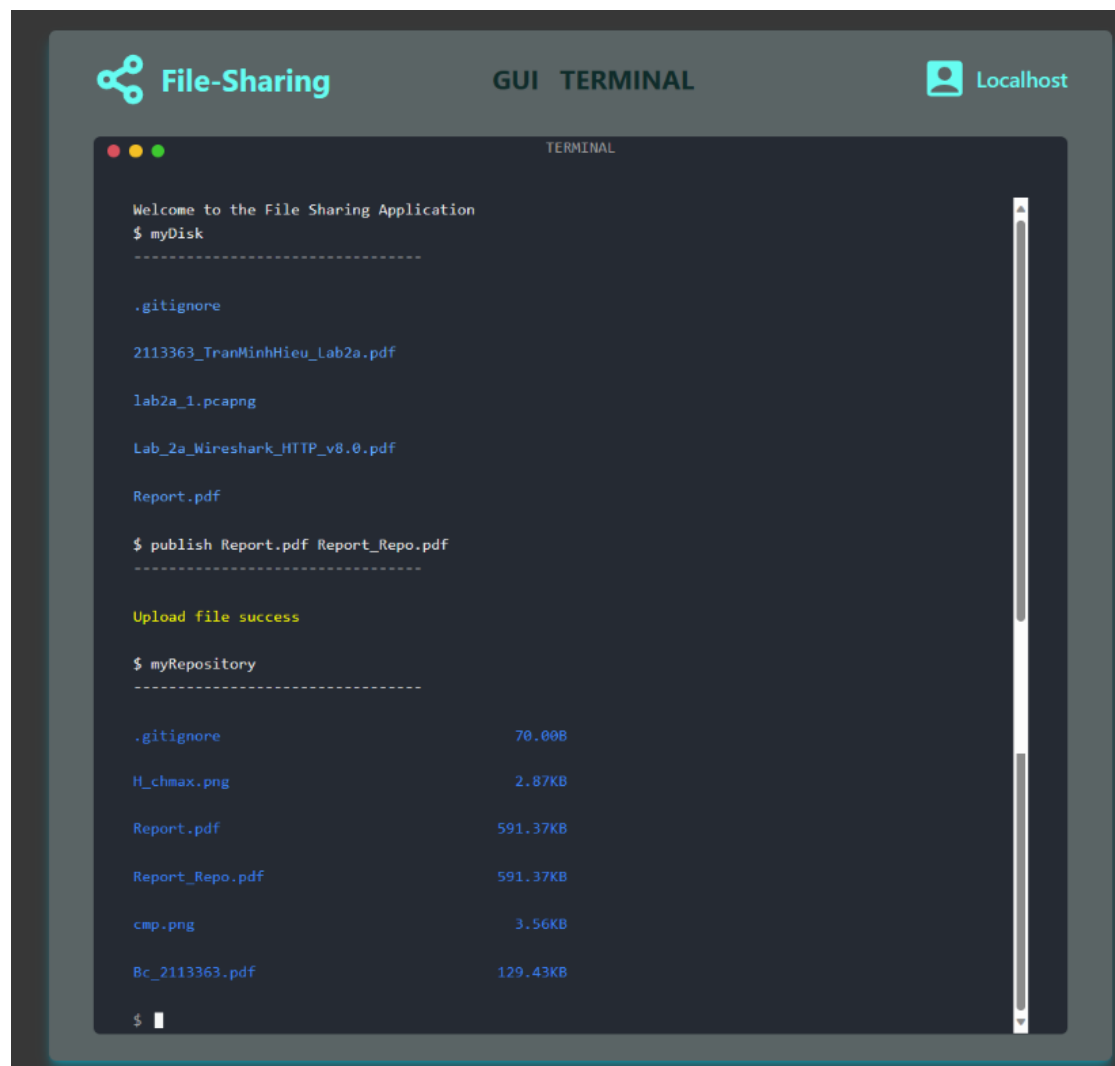
Hình 20: Tính năng xem tệp tin chia sẻ trong CLI

7.2.3 Tính năng publish file

Ứng dụng cho phép người dùng tải file từ máy lên local repository bằng cách nhấn vào nút "Choose file" trên cửa sổ "Repository" hoặc dùng lệnh "publish lname fname" (với lname là tên file trong máy và fname là tên file muốn đặt trong hệ thống file chia sẻ).



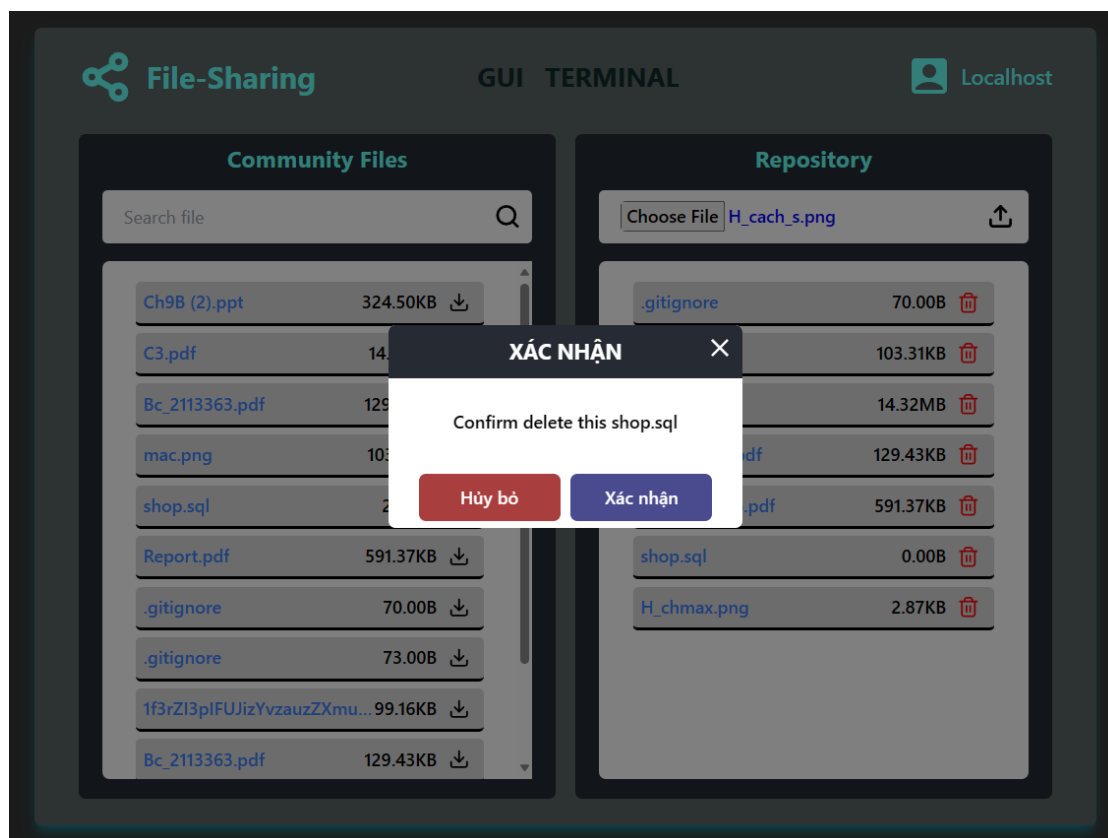
Hình 21: Tính năng publish file trong GUI



Hình 22: Tính năng publish file trong CLI

7.2.4 Tính năng delete file

Ứng dụng cho phép người dùng xóa file trong local repository bằng cách nhấn vào biểu tượng thùng rác ở bên phải tên file trong GUI hoặc dùng lệnh "delete fname" trong CLI.



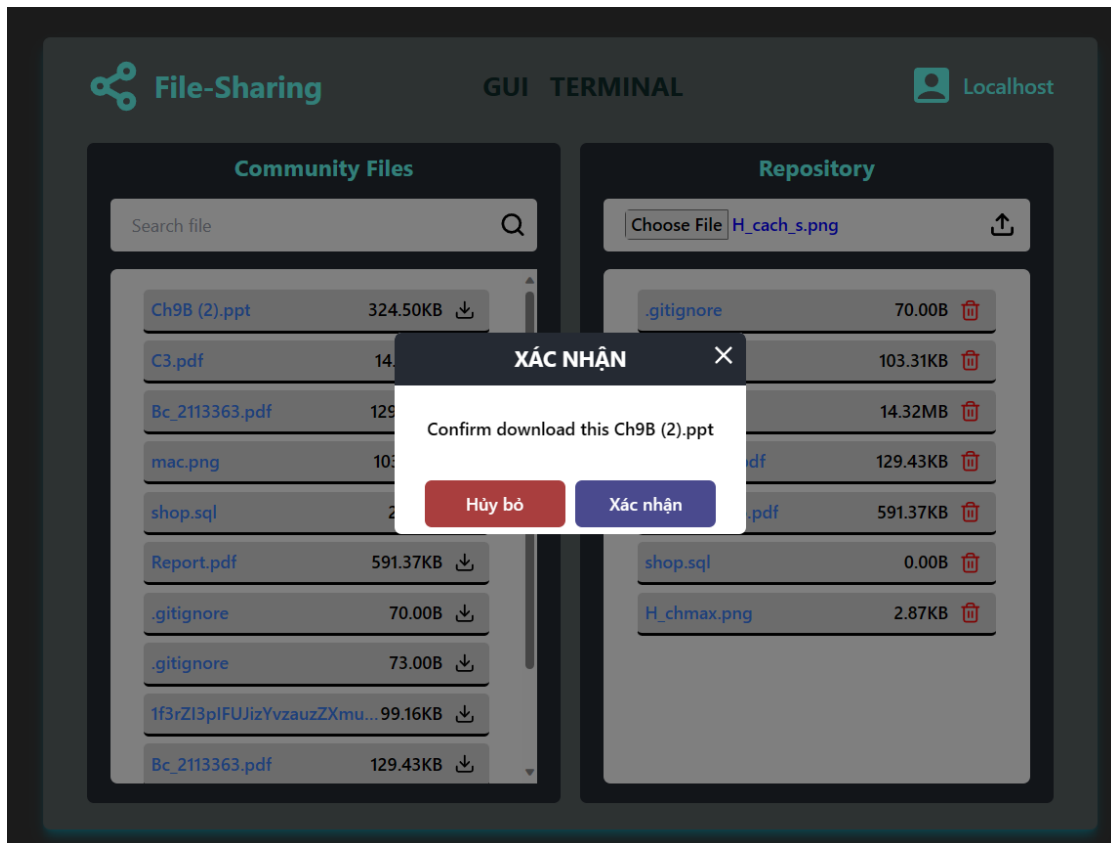
Hình 23: Tính năng delete file trong GUI



Hình 24: Tính năng delete file trong CLI

7.2.5 Tính năng fetch file

Người dùng có thể tải các tệp tin của các client khác có trong hệ thống thông qua việc xem các file được chia sẻ và bấm nút tải về trong GUI. Ngoài ra trong CLI, người dùng có thể dùng lệnh "fetch fname" hoặc "fetch fname peerIP" (với fname là tên file và peerIP là IP của client khác).



Hình 25: Tính năng fetch file trong GUI

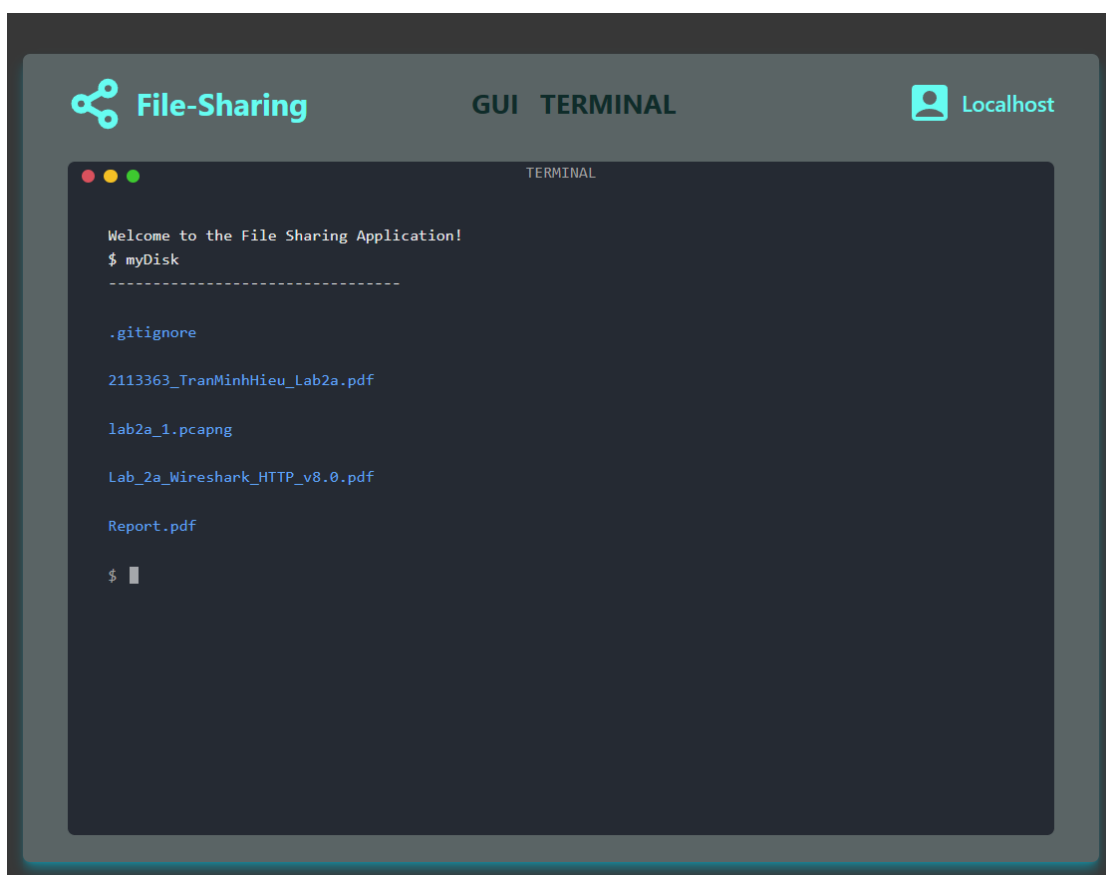


Hình 26: Tính năng fetch file trong CLI

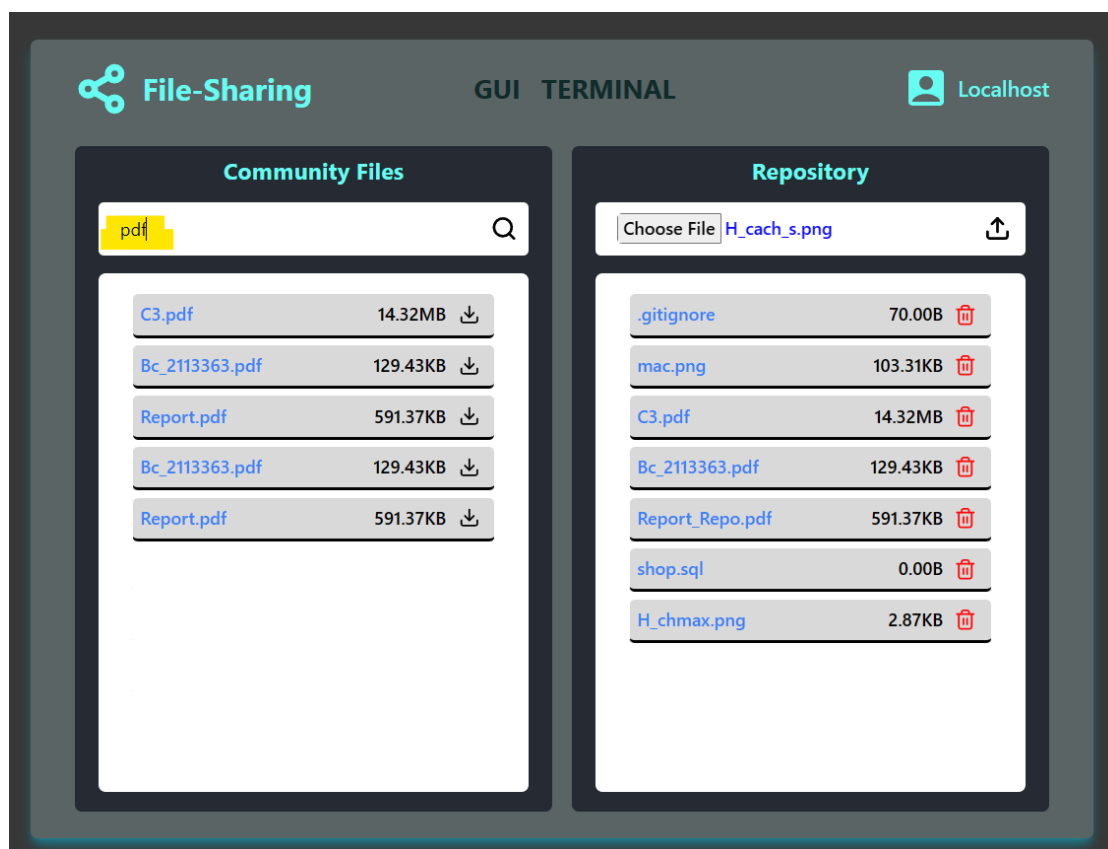
7.2.6 Các tính năng khác của client

Ngoài các tính năng chính như trên, ứng dụng còn cung cấp một số chức năng khác cho client như:

- Lệnh "myDisk" trong CLI cho phép người dùng xem được các file trong thư mục Disk (tương trưng cho các file nằm ngoài local repository trên máy người dùng).
- Khung tìm kiếm trên GUI cho phép người dùng tìm kiếm các file được chia sẻ trong hệ thống.



Hình 27: Tính năng xem các file bên ngoài local repository

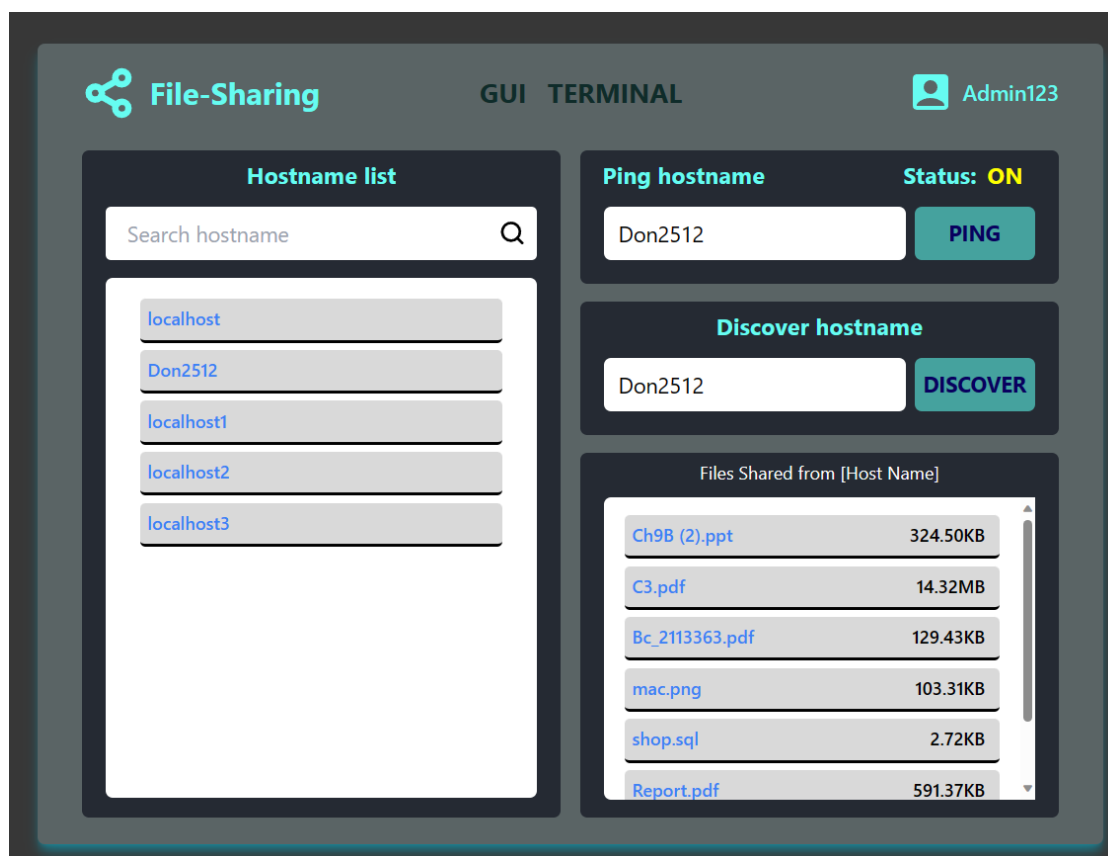


Hình 28: Tính năng tìm kiếm file

7.3 Các tính năng của server

Server có giao diện hỗ trợ các tính năng như:

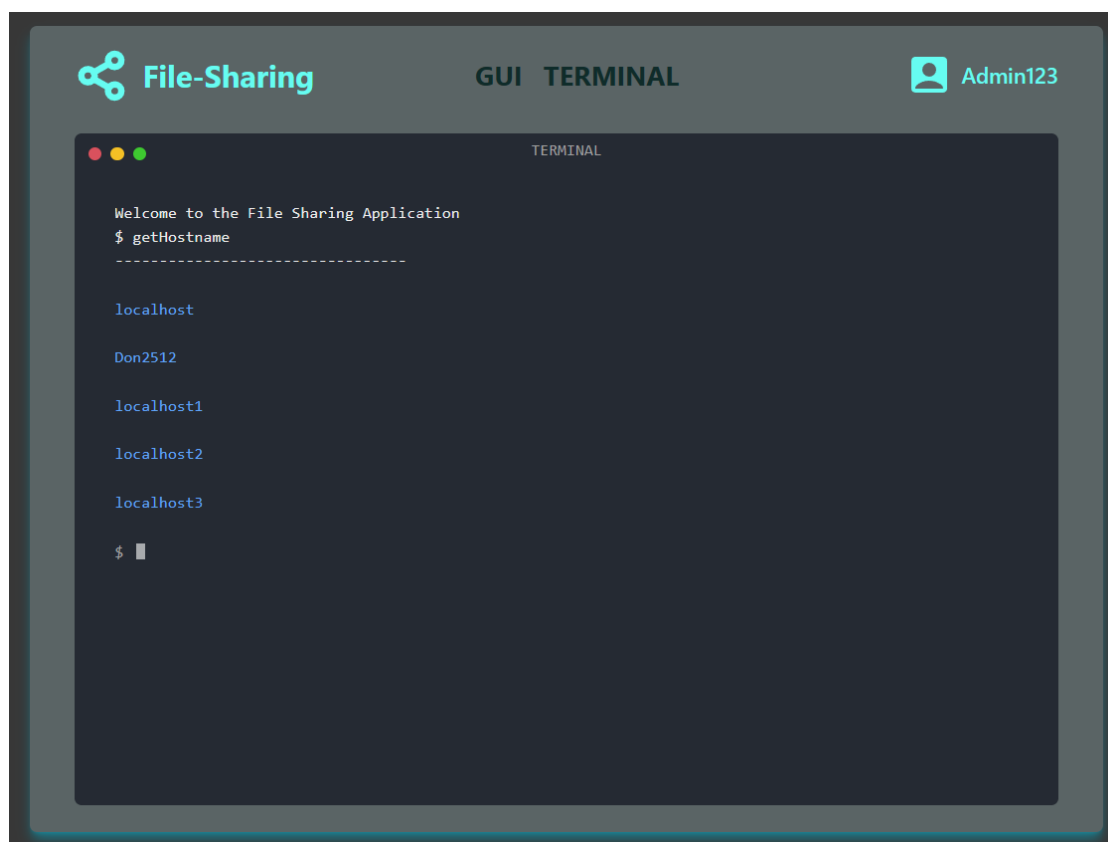
- getHostname: Xem danh sách các hostname của client trong hệ thống.
- ping hostname: Xem trạng thái của một client trong hệ thống khi biết hostname.
- discover hostname: Xem danh sách các tệp tin mà client đó đang chia sẻ.



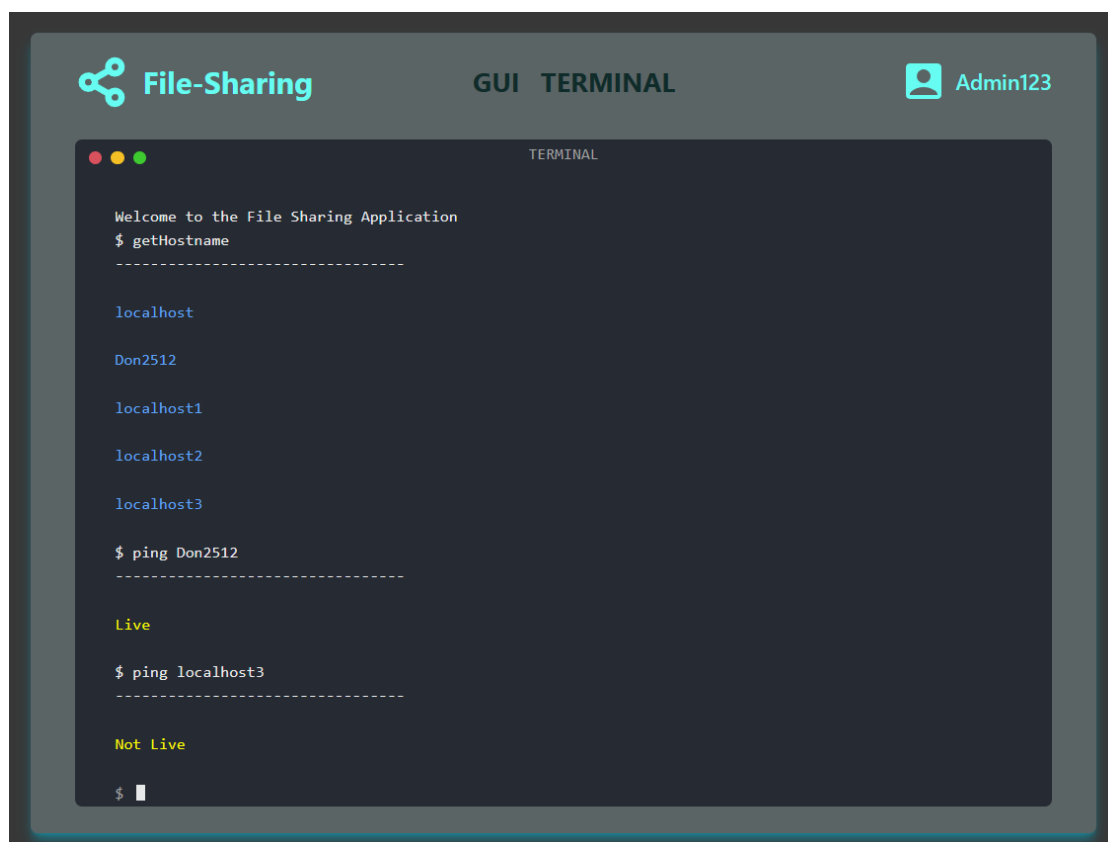
Hình 29: Các tính năng của server được tích hợp trong GUI



Hình 30: Tính năng discover hostname trong CLI



Hình 31: Tính năng getHostname trong CLI



Hình 32: Tính năng ping hostname trong CLI

8 Implementation Code

Link Code GitHub : [Tại Đây](#)

Trong *url* https://github.com/MinhHieu212/ASS1_MMT_FileSharing

9 Hướng dẫn sử dụng

(A) Phía Client

1. Tải file P2P

- B1: Kích hoạt server và các peer.
- B2: Các peer thực hiện việc đăng nhập vào hệ thống, phía server sẽ chịu trách nhiệm việc xác thực.
- B3: Sau khi đăng nhập các peer dùng lệnh "communityFile" để lấy danh sách các files hiện có trong hệ thống.
- B4: Các peer sẽ dùng lệnh "fetch fname" hoặc lệnh "fetch fname peerIP" để tải về file mong muốn.

2. Publish file vào repository

- B1: Kích hoạt server và peer.
- B2: Peer thực hiện việc đăng nhập vào hệ thống, phía server sẽ chịu trách nhiệm việc xác thực.
- B3: Sau khi đăng nhập peer dùng lệnh "myDisk" để lấy danh sách các file hiện có trong máy.
- B4: Các peer sẽ dùng lệnh "fetch fname" hoặc lệnh "fetch fname peerIP" để tải về file mong muốn.
- B5: Dùng lệnh "myRepository" để kiểm tra các file có trong repository .
Dùng lệnh "delete fname" nếu muốn xóa file trong repository.

(B) Phía Server

1. Ping tới các client

- B1: Kích hoạt server
- B2: Đăng nhập vào hệ thống
- B3: Dùng lệnh "discover hostname" để lấy danh sách các hostname đã đăng kí vào hệ thống
- B4: Dùng lệnh "ping hostname" để kiểm tra các hostname hiện đang hoạt động



10 Tài liệu tham khảo

Tài liệu

- [1] Client-Server and Peer-to-Peer Network: <https://www.geeksforgeeks.org/difference-between-client-server-and-peer-to-peer-network/>
- [2] MERN Stack Explained: <https://www.mongodb.com/mern-stack>
- [3] Introduction Socket.IO: <https://socket.io/docs/v4/>
- [4] Net | Node.js v21.1.0 Documentation: <https://nodejs.org/api/net.html>