

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



MẠNG MÁY TÍNH (THÍ NGHIỆM) (CO3094)

Báo cáo Bài tập lớn

Bài tập lớn 1

DEVELOP A NETWORK APPLICATION

Giảng viên hướng dẫn: Lê Bảo Khánh
Nhóm: Baby Panther
Sinh viên thực hiện: Nguyễn Thanh Hiền (2111203)
Nguyễn Nhật Khải (2111506)
Phạm Văn Nhật Vũ (2110676)
Nguyễn Ngọc Phú (2114417)

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 10 2023

Mục lục

1	TỔNG QUAN ĐỀ TÀI	2
1.1	Thông tin chung	2
1.2	Phạm vi đề tài	2
1.3	Phân công nhiệm vụ	3
2	CƠ SỞ LÝ THUYẾT & GIẢI PHÁP ĐỀ XUẤT	4
2.1	Mô hình Peer-to-Peer (P2P) tập trung	4
2.1.1	Tổng quan	4
2.1.2	Giao thức truyền dữ liệu	5
2.1.3	Phạm vi kết nối	5
2.2	Lập trình ứng dụng	6
2.2.1	Git	6
2.2.2	Github	7
2.2.3	React JS	7
2.2.4	Fastify	8
2.2.5	Prisma	8
2.2.6	PostgreSQL	9
2.2.7	Min.io	9
2.3	Tóm tắt công nghệ sử dụng	10
3	YÊU CẦU SẢN PHẨM	11
3.1	Yêu cầu tính năng	11
3.1.1	Server	11
3.1.1.a	Truy xuất thông tin client (<i>mở rộng</i>)	11
3.1.1.b	Kiểm tra tính khả dụng của client	11
3.1.1.c	Truy xuất tài liệu chia sẻ	12
3.1.1.d	Quản lý cơ sở dữ liệu tập trung (<i>mở rộng</i>)	12
3.1.2	Client	12
3.1.2.a	Tạo tài khoản/Đăng nhập/Đăng xuất	12
3.1.2.b	Công bố/Hủy công bố tài liệu	13
3.1.2.c	Tìm kiếm client công bố tài liệu	14
3.1.2.d	Tải tài liệu từ cộng đồng	14
3.1.2.e	Quản lý kho tài liệu cá nhân (<i>mở rộng</i>)	15
3.2	Yêu cầu phi tính năng	15
3.2.1	Hiệu suất hoạt động	15
3.2.2	Khả năng tương thích	15
3.3	Yêu cầu giao thức	15
3.4	Yêu cầu giao diện	16

3.4.1	Server	16
3.4.2	Client	17
3.5	Yêu cầu dữ liệu	18
3.5.1	Lưu trữ tài liệu	18
3.5.2	Lưu trữ thông tin hệ thống	18
4	THIẾT KẾ HỆ THỐNG	20
4.1	Tổng quan hệ thống	20
4.2	Server	21
4.2.1	Truy xuất thông tin client (<i>mở rộng</i>)	21
4.2.2	Kiểm tra tính khả dụng của client	22
4.2.3	Truy xuất tài liệu chia sẻ	24
4.2.4	Quản lý cơ sở dữ liệu tập trung (<i>mở rộng</i>)	26
4.3	Client	28
4.3.1	Tạo tài khoản/Dăng nhập/Dăng xuất	28
4.3.1.a	Tạo tài khoản	28
4.3.1.b	Dăng nhập	29
4.3.1.c	Dăng xuất	32
4.3.2	Công bố/Hủy công bố tài liệu	33
4.3.2.a	Công bố tài liệu	33
4.3.2.b	Hủy công bố tài liệu	35
4.3.3	Tìm kiếm client công bố tài liệu	37
4.3.3.a	Truy xuất tài liệu khả dụng	37
4.3.3.b	Truy xuất thông tin client theo tên tài liệu	38
4.3.4	Tải tài liệu từ cộng đồng	40
5	KIỂM THỬ	42
5.1	Thiết lập hệ thống	42
5.2	Kiểm tra tính khả dụng của client	42
5.3	Công bố/Hủy công bố tài liệu	43
5.4	Client yêu cầu tải dữ liệu	44
6	HƯỚNG DẪN SỬ DỤNG	45
6.1	Hướng dẫn cài đặt	45
6.1.1	Cài đặt các công cụ cần thiết	45
6.1.2	Cài đặt ứng dụng ở client	45
6.1.3	Cài đặt ứng dụng ở server	47
6.2	Hướng dẫn thao tác trên hệ thống	48
6.2.1	Đối với người quản lý hệ thống	48
6.2.1.a	Dăng nhập hệ thống	48

6.2.1.b	Truy xuất thông tin client (<i>mở rộng</i>)	48
6.2.1.c	Kiểm tra tính khả dụng của client	49
6.2.1.d	Truy xuất tài liệu chia sẻ	49
6.2.1.e	Quản lý cơ sở dữ liệu tập trung (<i>mở rộng</i>)	50
6.2.2	Đối với người dùng thông thường	51
6.2.2.a	Tạo tài khoản/Đăng nhập	51
6.2.2.b	Công bố/Hủy công bố tài liệu	52
6.2.2.c	Tìm kiếm client công bố tài liệu	53
6.2.2.d	Tải tài liệu từ cộng đồng	54
6.2.2.e	Quản lý kho tài liệu cá nhân (<i>mở rộng</i>)	54

7 TÀI LIỆU THAM KHẢO 56

Danh sách hình vẽ

1	Mô hình P2P tập trung (từ Yari et al, 2021)	4
2	Kỹ thuật NAT (từ Kurose, J.F. & Ross, K.W., 2020, trang 345)	5
3	Git	6
4	Github	7
5	ReactJS	7
6	Fastify	8
7	Prisma	8
8	PostgreSQL	9
9	Min.io	9
10	Giao diện: Màn hình chính server	16
11	Giao diện: Cơ sở dữ liệu tập trung	16
12	Giao diện: Màn hình chính client	17
13	Giao diện: Kho tài liệu cá nhân	17
14	Mô hình hệ thống ở mức ý niệm của nhóm	20
15	Use case: Truy xuất thông tin client	21
16	Activity diagram: Truy xuất thông tin client	21
17	Use case: kiểm tra tính khả dụng của client	22
18	Activity diagram: kiểm tra tính khả dụng của client	23
19	Use case diagram cho lệnh tính năng truy xuất tài liệu chia sẻ	24
20	Activity diagram: truy xuất tài liệu chia sẻ	25
21	Use case: Tạo tài khoản	28
22	Activity diagram: Tạo tài khoản	28
23	Use case: Đăng nhập	29
24	Activity diagram: Đăng nhập	30
25	Use case: Đăng xuất	32

26	Activity diagram: Đăng xuất	32
27	Use case: Công bố tài liệu	33
28	Activity diagram: Công bố tài liệu	34
29	Use case: Hủy công bố tài liệu	35
30	Activity diagram: hủy công bố tài liệu	36
31	Use case: Truy xuất tài liệu khả dụng	37
32	Activity diagram: Truy xuất tài liệu khả dụng	37
33	Use case: Tìm kiếm tài liệu bằng tên	38
34	Activity diagram: Tìm kiếm tài liệu bằng tên	39
35	Use case: tải tài liệu từ cộng đồng	40
36	Activity diagram: tải tài liệu từ cộng đồng	40
37	Tạo bucket mới	46
38	Tạo access key và secret key mới	46
39	Chỉnh quyền truy cập PUBLIC cho bucket	47
40	Giao diện: Truy xuất thông tin client	49
41	Giao diện: Kiểm tra tính khả dụng client	49
42	Giao diện: Truy xuất tài liệu chia sẻ	50
43	Giao diện: Các bảng dữ liệu	50
44	Giao diện: Các trường dữ liệu của bảng	51
45	Giao diện: Tạo tài khoản	51
46	Giao diện: Đăng nhập tài khoản	52
47	Giao diện: Công bố tài liệu	53
48	Giao diện: Tìm kiếm client công bố tài liệu	54
49	Giao diện: Tải tài liệu từ cộng đồng	54
50	Giao diện: Quản lý kho tài liệu cá nhân	55

Danh sách bảng

1	Phân công nhiệm vụ	3
2	Giải pháp công nghệ sử dụng	10
3	Lệnh server: Truy xuất cơ sở dữ liệu	11
4	Lệnh server: Kiểm tra tính khả dụng client	11
5	Lệnh server: Truy xuất tài liệu chia sẻ	12
6	Dữ liệu quản lý	12
7	Lệnh client: Đăng xuất	13
8	Lệnh client: Công bố/Hủy công bố tài liệu	13
9	Lệnh client: Tìm kiếm client công bố tài liệu	14
10	Lệnh client: Tải tài liệu từ cộng đồng	14
11	Lưu trữ tài liệu	18
12	Lưu trữ thông tin hệ thống	19



13	Data: User	26
14	Data: Session	26
15	Data: SharedDocument	27
16	Kiểm thử: Thiết lập hệ thống	42
17	Kiểm thử: ping	43
18	Kiểm thử: publish	43
19	Kiểm thử: ls và fetch	44

1 TỔNG QUAN ĐỀ TÀI

1.1 Thông tin chung

- **Tên đề tài:** Ứng dụng mạng P2P tập trung để chia sẻ tài liệu
- **Thời gian thực hiện:** 12/10/2023 - 09/11/2023
- **Số lượng thành viên hiện thực:** 4 sinh viên
- **Mục tiêu đề tài:** Xây dựng một ứng dụng chia sẻ tài liệu đơn giản với mô hình mạng P2P tập trung, sử dụng giao thức TCP/IP.

Để trải nghiệm sản phẩm của nhóm, người dùng sẽ cần thực hiện theo phần [HƯỚNG DẪN SỬ DỤNG](#).

1.2 Phạm vi đề tài

Giải pháp đề xuất để hoạt động với những ràng buộc về phạm vi như sau:

- **Sản phẩm:** Hiện thực một web app có tích hợp giao diện CLI để người dùng sử dụng các câu lệnh được định nghĩa và sử dụng GUI của các ứng dụng đi kèm.
- **Mô hình sử dụng:** P2P với cấu trúc tập trung. Các client, server kết nối cùng một mạng LAN với server được thiết lập mới mỗi lần cần khởi tạo toàn bộ hệ thống.
- **Lưu trữ tài liệu:** lưu trữ cục bộ trên local repository của các client chia sẻ/tải tài liệu, server chỉ lưu trữ thông tin để vận hành hệ thống.

Các chức năng chính được hiện thực phục vụ cho hai đối tượng:

1. **Server:** Trung tâm tập trung thông tin về các tài liệu, máy khách đang khả dụng để thực hiện quá trình truyền/nhận tài liệu.
 - Truy xuất thông tin client (*mở rộng*): `ls clients active`
 - Kiểm tra tính khả dụng của client: `ping hostname`
 - Truy xuất tài liệu chia sẻ: `discover hostname`
 - Quản lý cơ sở dữ liệu tập trung (*mở rộng*)
2. **Client:** Các máy khách lưu trữ tài liệu dưới ổ đĩa cá nhân mong muốn chia sẻ tài liệu với các máy khác trong mạng
 - Tạo tài khoản/Đăng nhập/Đăng xuất

- Công bố/Hủy công bố tài liệu: **publish** fname và **unpublish** fname
- Tìm kiếm client công bố tài liệu (mở rộng): **ls** và **filter** fname
- Tải tài liệu từ cộng đồng: **fetch** fname hostname
- Quản lý kho tài liệu cá nhân (mở rộng)

1.3 Phân công nhiệm vụ

#	Họ và tên	MSSV	Nhiệm vụ	Hoàn thành
1	Nguyễn Thanh Hiền	2111203	- Cơ sở lý thuyết: Lập trình ứng dụng - Đặc tả yêu cầu sản phẩm - Thiết kế giao diện người dùng - Kiểm thử: Xây dựng kịch bản kiểm thử - Thuyết trình sản phẩm: Xây dựng slide và thuyết trình	
2	Nguyễn Nhật Khải	2111506	- Đặc tả yêu cầu sản phẩm - Thiết kế hệ thống: Mô tả - Thiết kế cơ sở dữ liệu - Kiểm thử & Đánh giá kết quả hệ thống	
3	Phạm Văn Nhật Vũ	2110676	- Cơ sở lý thuyết: Mô hình P2P chia sẻ tài liệu - Thiết kế hệ thống: Use case diagram và Activity diagram - Kiểm thử & Đánh giá kết quả hệ thống	
4	Nguyễn Ngọc Phú	2114417	- Hiện thực ứng dụng: Front-end - Hiện thực ứng dụng: Back-end - Hướng dẫn cài đặt hệ thống	

Bảng 1: Phân công nhiệm vụ

2 CƠ SỞ LÝ THUYẾT & GIẢI PHÁP ĐỀ XUẤT

2.1 Mô hình Peer-to-Peer (P2P) tập trung

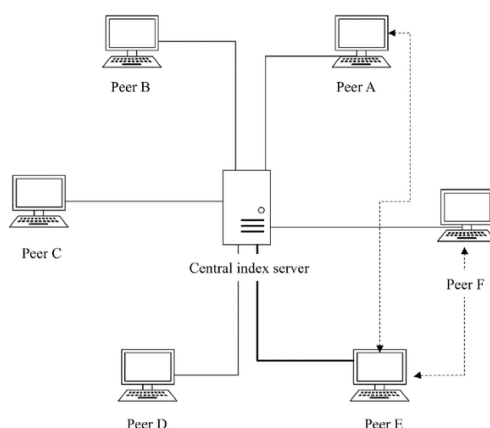
2.1.1 Tổng quan

Trong mạng máy tính, Peer-to-Peer (P2P) là mô hình cho phép thiết lập kết nối trực tiếp giữa các máy tính (được gọi là các **peer**) mà không cần thông qua một máy chủ tập trung (như trong mô hình Client-Server).

Về bản chất, mỗi peer trong mạng P2P đều thực thi một process "*server*" và một process "*client*" để nhận (gửi) các yêu cầu giao tiếp từ (tới) các peer khác trong mạng.

Mô hình P2P tập trung (được sử dụng trong Bài tập lớn này) bao gồm một hoặc nhiều server (được gọi là các **index server**) có nhiệm vụ lưu trữ, quản lý các thông tin (như địa chỉ IP, các tài nguyên đang được sử dụng,...) của các peers trong mạng. Khi người dùng cần tìm kiếm hay truy xuất một tài nguyên nào đó, yêu cầu của người dùng sẽ được gửi tới các index server, các server này sẽ phản hồi lại các yêu cầu đó bằng cách gửi danh sách các peer đang chứa tài nguyên cần truy xuất, người dùng có thể lựa chọn kết nối trực tiếp tới một hoặc nhiều peer trong danh sách để thực hiện thao tác mong muốn.

Tuy nhiên, vì cần một server trung tâm tương tự như mô hình Client-Server nên hệ thống sẽ không còn khả dụng nếu server phát sinh lỗi (single point of failure). Nhưng, cách hiện thực như trên sẽ đơn giản hơn mô hình phi tập trung vì các peer sẽ có thể lấy thông tin truy cập một peer khác từ server. Bên cạnh đó, mỗi client sẽ tự lưu trữ tài liệu (dung lượng lớn) cục bộ, server sẽ chịu trách nhiệm lưu trữ metadata (dung lượng nhỏ) để tập trung cho công tác xử lý và vận hành nên sẽ hiệu quả hơn Client-Server thông thường.



Hình 1: Mô hình P2P tập trung (từ Yari et al, 2021)

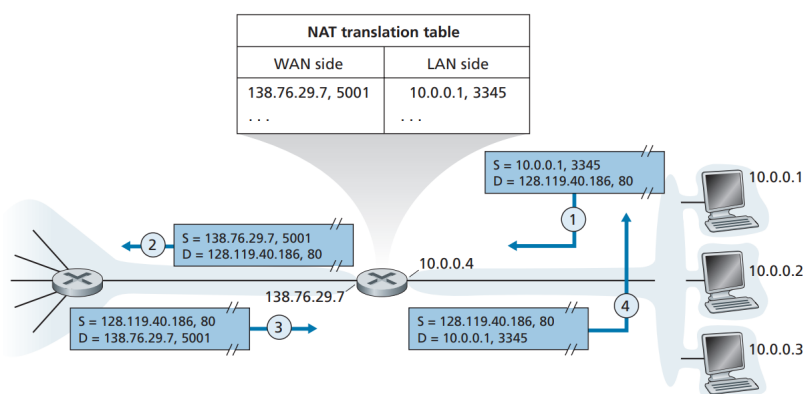
2.1.2 Giao thức truyền dữ liệu

Nhóm sử dụng hoàn toàn giao thức HTTP ở tầng Application để hiện thực một ứng dụng mạng P2P tập trung. Giao thức trên được xây dựng dựa trên giao thức TCP ở tầng Transport, do đó có thể đảm bảo được các thông tin sẽ được truyền đi một cách tin cậy.

HTTP: là giao thức mà trình duyệt thường sử dụng để tải các trang web, giao thức này định nghĩa cấu trúc của các gói tin và cách chúng được truyền nhận giữa trình duyệt và server. Trình duyệt gửi các gói tin HTTP để yêu cầu các đối tượng của trang web cần tải, server phản hồi lại các yêu cầu này bằng cách gửi gói tin HTTP chứa các đối tượng tương ứng về trình duyệt. Cơ chế này phù hợp để hiện thực các tính năng truy xuất tài liệu chia sẻ (lệnh **discover**) của server; tạo tài khoản, đăng nhập, đăng xuất, quản lý tài liệu chia sẻ (lệnh **publish** và **unpublish**) và tải tài liệu từ cộng đồng (lệnh **fetch**) của client.

2.1.3 Phạm vi kết nối

Trên thực tế, kỹ thuật **NAT** (Network Address Translation) thường xuyên được sử dụng để cấu hình địa chỉ IP **private** cho các thiết bị trong cùng một mạng LAN mà không cần phải thuê hoặc mua địa chỉ IP **public** cho từng thiết bị (với chi phí cao và bị giới hạn về số lượng do địa chỉ IPv4 chỉ có độ dài 32 bit). Cụ thể, một router tích hợp NAT sẽ được coi như một thiết bị với một địa chỉ IP duy nhất khi giao tiếp với các thiết bị bên ngoài mạng LAN kết nối bởi router đó. Khi một host nằm trong mạng LAN này muốn giao tiếp với các thiết bị bên ngoài, địa chỉ **private** của host và port của process thực hiện việc giao tiếp thông qua router sẽ được dịch thành địa chỉ **public** của router và một port nhất định để giao tiếp với các thiết bị bên ngoài mạng LAN.



Hình 2: Kỹ thuật NAT (từ Kurose, J.F. & Ross, K.W., 2020, trang 345)

Kỹ thuật này là phù hợp với mô hình Client-Server khi địa chỉ IP của các server thường là địa chỉ IP **public**, client có thể sử dụng trực tiếp địa chỉ này để kết nối tới server và server có thể dựa vào các header **source address** và **destination address** để phản hồi về đúng client. Tuy nhiên, trong mô hình P2P, việc thiết lập kết nối cho các peer thuộc các mạng LAN khác nhau có áp dụng NAT sẽ phức tạp hơn, một số kỹ thuật như *hole punching* có thể được sử dụng để giải quyết vấn đề này. Tuy nhiên, với giới hạn về thời gian và nguồn lực, nhóm quyết định sẽ chỉ xây dựng ứng dụng cho các thiết bị trong cùng mạng LAN để tập trung hiện thực logic của mô hình Peer-to-Peer.

2.2 Lập trình ứng dụng

Để hiện thực ứng dụng chia sẻ file đơn giản, ngoài các giao thức được sử dụng thì nhóm sử dụng các công nghệ sau để lập trình sản phẩm là một ứng dụng web.

- **Quản lý phiên bản, mã nguồn:** Git và GitHub
- **Front-end:** Framework ReactJS
- **Back-end:** Fastify ánh xạ tới cơ sở dữ liệu với Prisma và sử dụng PostgreSQL để quản lý cơ sở dữ liệu tập trung ở server. Còn ở phía mỗi client, nhóm sử dụng Min.io để quản lý tài liệu gốc với local repo.

2.2.1 Git



Hình 3: Git

Git là một hệ thống quản lý phiên bản phân tán nguồn mở, được Linus Torvalds tạo ra vào năm 2005 để giải quyết nhu cầu quản lý mã nguồn hiệu quả trong quá trình phát triển nhân Linux. Hệ thống này đã trở thành một trong những hệ thống quản lý phiên bản phổ biến nhất được sử dụng hiện nay. Git được sử dụng để theo dõi những thay đổi trong mã nguồn trong quá trình phát triển phần mềm và cho phép nhiều lập trình viên làm việc đồng thời trên cùng một cơ sở mã nguồn.

2.2.2 Github



Hình 4: Github

GitHub là một nền tảng web bổ sung cho các chức năng của Git. GitHub cung cấp dịch vụ lưu trữ cho các kho Git (Git repositories) cùng với nhiều tính năng quản lý dự án và cộng tác. Với GitHub, người dùng có thể lưu trữ kho Git của họ trên Internet, giúp chúng có thể truy cập được từ mọi nơi có kết nối Internet. Cách tiếp cận này giúp đơn giản hóa việc cộng tác vì các thành viên trong nhóm có thể dễ dàng sao chép các kho Git, tạo nhánh và gửi “yêu cầu kéo” (Pull Requests) để xem xét và tích hợp mã.

2.2.3 React JS



Hình 5: ReactJS

ReactJS là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng. Đây là một thư viện khai báo, hiệu quả và linh hoạt, chịu trách nhiệm về lớp View của ứng dụng. Nó cho phép các nhà phát triển tạo các thành phần UI có thể tái sử dụng, có thể được lồng ghép với các thành phần khác để xây dựng các ứng dụng phức tạp từ các khối xây dựng đơn giản. ReactJS sử dụng cơ chế dựa trên DOM ảo để điền dữ liệu vào HTML DOM, cơ chế này hoạt động nhanh vì nó chỉ thay đổi các thành phần DOM riêng lẻ thay vì tải lại toàn bộ DOM mỗi lần.

2.2.4 Fastify



Hình 6: Fastify

Fastify là một web framework dành cho Node.js tập trung chủ yếu vào việc cung cấp trải nghiệm tốt lập trình viên với chi phí thấp và một kiến trúc plugin mạnh mẽ. Fastify lấy cảm hứng từ Hapi và Express và là một trong những web framework nhanh nhất hiện tại. Fastify có khả năng mở rộng thông qua các hook, plugin và decorators. Fastify có một trình logging được tích hợp sẵn, từ đó gần như loại bỏ chi phí cho tác vụ logging.

2.2.5 Prisma



Hình 7: Prisma

Prisma là một ORM thế hệ mới giúp đơn giản hóa quy trình làm việc với cơ sở dữ liệu và thay thế các ORM truyền thống. Nó cung cấp khả năng truy cập cơ sở dữ liệu an toàn kiểu (type-safe) với Prisma Client, di chuyển (migrate) dữ liệu với Prisma Migrate cũng như quản lý dữ liệu trực quan với Prisma Studio. Prisma Client có thể được sử dụng để xây dựng các API GraphQL, REST, gRPC và hơn thế nữa. Prisma hiện hỗ trợ nhiều hệ thống quản lý cơ sở dữ liệu như MySQL, PostgreSQL và MongoDB.

2.2.6 PostgreSQL



Hình 8: PostgreSQL

PostgreSQL là một hệ thống cơ sở dữ liệu quan hệ đối tượng nguồn mở sử dụng và mở rộng ngôn ngữ SQL kết hợp với nhiều tính năng lưu trữ và mở rộng quy mô một cách an toàn cho các khối lượng dữ liệu phức tạp nhất. PostgreSQL nổi tiếng bởi kiến trúc đã được chứng minh, độ tin cậy, tính toàn vẹn dữ liệu, bộ tính năng mạnh mẽ, khả năng mở rộng và cộng đồng nguồn mở đẳng cấp sau cung cấp nhất quán các giải pháp sáng tạo và hiệu quả.

2.2.7 Min.io



Hình 9: Min.io

Minio là một object storage server được implement những public API giống như AWS S3, là một dịch vụ tương tự AWS S3 nhưng được host local. Minio là một server lưu trữ object nên có thể được sử dụng để lưu trữ những unstructured data như ảnh, video, log files, backups và container/VM images. Thay vì cấu trúc theo "folder và file", Minio sử dụng "bucket và object". Với "bucket" tương tự như "folder" và "object" tương ứng với "file"

Dung lượng của 1 object có thể dao động từ một vài KB tới tối đa là 5TB. **Tuy nhiên, dung lượng lưu trữ tối đa thực sự của Minio sẽ phụ thuộc**

vào dụng lượng máy tính người dùng vì thực tế, tất cả các dữ liệu đều được lưu trữ trên ổ đĩa người dùng. File cũng được gom lại trong 1 buckets, nó là được chỉ cùng với access key khi dùng app. Minio phù hợp để lưu trữ và truyền khối lượng lớn dữ liệu lớn thông qua HTTP.

2.3 Tóm tắt công nghệ sử dụng

Mục tiêu	Giải pháp
Mô hình kết nối	P2P tập trung
Giao thức truyền dữ liệu	HTTP
Phạm vi kết nối	Các thiết bị cùng một mạng LAN
Quản lý phiên bản, mã nguồn	Git, Github
Front-end	ReactJS
Back-end	Fastify, Prisma, PostgreSQL và Minio

Bảng 2: Giải pháp công nghệ sử dụng

3 YÊU CẦU SẢN PHẨM

3.1 Yêu cầu tính năng

Đối với mỗi vai trò trong hệ thống, ứng dụng sẽ cung cấp đồng thời 2 giao diện (tham khảo phần [Yêu cầu giao diện](#)) để thao tác trên hệ thống:

1. Giao diện chính: GUI mô phỏng hành vi CLI
2. Giao diện hỗ trợ: quản lý kho lưu trữ cá nhân (min.io ở client) hoặc quản lý cơ sở dữ liệu tập trung (prisma ở server)

(*) "**Lệnh ứng dụng**" sẽ là các phần được thao tác trên giao diện chính.

3.1.1 Server

Vai trò trong hệ thống: Trung tâm tập trung thông tin về tài khoản người dùng, các tài liệu, máy khách, phiên hoạt động đang khả dụng để thực hiện quá trình truyền/nhận tài liệu.

3.1.1.a Truy xuất thông tin client (*mở rộng*)

Mô tả: Admin được quyền truy xuất cơ sở dữ liệu lưu trữ của hệ thống. Bên cạnh đó, ứng dụng cũng có khả năng lọc để hiển thị các dữ liệu còn khả dụng.

#	Lệnh ứng dụng	Input	Output
1	<code>ls clients active</code>	Không	Username các client đang hoạt động trong hệ thống

Bảng 3: Lệnh server: Truy xuất cơ sở dữ liệu

3.1.1.b Kiểm tra tính khả dụng của client

Mục đích: Server có thể chủ động liên lạc cụ thể một client để kiểm tra tính khả dụng hiện tại của client và địa chỉ IP của client.

#	Lệnh ứng dụng	Input	Output
2	<code>ping hostname</code>	- hostname: Tên tài khoản của client	Kết quả kiểm tra tính khả dụng của client: - Địa chỉ IP của client - Alive/Dead

Bảng 4: Lệnh server: Kiểm tra tính khả dụng client

3.1.1.c Truy xuất tài liệu chia sẻ

Mục đích: Truy xuất danh sách các tài liệu được chia sẻ bởi một client cụ thể vẫn còn khả dụng để yêu cầu tải về.

#	Lệnh ứng dụng	Input	Output
1	discover hostname	- hostname: Tên đăng nhập của client	Danh sách tài liệu do client hostname công bố: <ul style="list-style-type: none">- Name: Tên tài liệu- Type: Loại tài liệu- Size: Kích thước tài liệu- Shared time: Thời gian công bố- Is available: Đánh dấu tài liệu còn khả dụng

Bảng 5: Lệnh server: Truy xuất tài liệu chia sẻ

3.1.1.d Quản lý cơ sở dữ liệu tập trung (*mở rộng*)

Mục đích: Admin theo dõi, thao tác và đánh giá quá trình vận hành của hệ thống và được thao tác chủ yếu qua giao diện hỗ trợ từ Prisma.

Bảng dữ liệu	data_table
Tài khoản người dùng	User
Phiên hoạt động	Session
Tài liệu chia sẻ	SharedDocument

Bảng 6: Dữ liệu quản lý

3.1.2 Client

Vai trò trong hệ thống: Mỗi máy lưu trữ tài liệu dưới ổ đĩa cá nhân, công bố lên hệ thống rằng tài liệu có thể được tải xuống hoặc yêu cầu tải một tài liệu được chia sẻ trên hệ thống.

3.1.2.a Tạo tài khoản/Đăng nhập/Đăng xuất

Mục đích: Người dùng tạo định danh để đăng nhập/đăng xuất, thao tác với các chức năng trên hệ thống

1. **Tạo tài khoản:** Người dùng thông qua GUI tạo tài khoản mới để truy cập hệ thống, với các thông tin:
 - Họ và tên
 - Tên đăng nhập
 - Mật khẩu
2. **Đăng nhập tài khoản:** Người dùng thông qua GUI với thông tin tài khoản đã tạo trước đó để, hệ thống sẽ tự động publish các tài liệu vẫn còn trong local repo của người dùng. Thông tin đăng nhập cần thiết là:
 - Tên đăng nhập
 - Mật khẩu
3. **Đăng xuất tài khoản:** Người dùng thông qua GUI mô phỏng CLI để đăng xuất khỏi phiên hoạt động, không cho hệ thống truy cập tài liệu đã công bố

#	Lệnh ứng dụng	Input	Output
1	logout	Không	- Đăng xuất người dùng khỏi hệ thống - Đánh dấu các tài liệu không khả dụng

Bảng 7: Lệnh client: Đăng xuất

3.1.2.b Công bố/Hủy công bố tài liệu

Mục đích: Công bố/Hủy công bố các tài liệu chia sẻ công khai, quản lý quyền truy cập để các client khác yêu cầu tải tài liệu.

#	Lệnh ứng dụng	Input	Output
1	publish fname	- fname : Tên tài liệu sẽ công bố (bao gồm đuôi file) - Tài liệu trong ổ đĩa mong muốn công bố	- Tài liệu được công bố với tên fname .
2	unpublish fname	- fname : Tên tài liệu đã công bố (bao gồm đuôi file)	- Hủy công bố tài liệu, xóa trên local repo - Đánh dấu tài liệu không khả dụng

Bảng 8: Lệnh client: Công bố/Hủy công bố tài liệu

3.1.2.c Tìm kiếm client công bố tài liệu

Mục đích: Tìm kiếm thông tin của tất cả các client đã công bố dữ liệu trùng tên để nắm thông tin, sử dụng trong các thao tác tiếp theo.

#	Lệnh ứng dụng	Input	Output
1	ls	Không	Thông tin tài liệu khả dụng trong hệ thống: - Name: Tên tài liệu - Username: Tên đăng nhập của client công bố tài liệu - IP address: Địa chỉ IP client công bố - Type: Loại tài liệu - Size: Kích thước tài liệu - Shared time: Thời gian công bố
	filter fname	- fname: Tên tài liệu đã công bố (bao gồm đuôi file)	- Địa chỉ IP client công bố tài liệu

Bảng 9: Lệnh client: Tìm kiếm client công bố tài liệu

3.1.2.d Tải tài liệu từ cộng đồng

Mục đích: Yêu cầu tải tài liệu được chia sẻ bởi các client khác trên hệ thống vào kho lưu trữ cá nhân. Hệ thống cần được xử lý đa luồng để cho phép nhiều client cùng tải từ một client trong cùng một thời điểm.

#	Lệnh ứng dụng	Input	Output
1	fetch fname hostname	- fname: Tên tài liệu trên cộng đồng - hostname: Địa chỉ IP của client đã công bố	- Tài liệu được tải xuống local repo - Tài liệu đồng thời được công bố lên server

Bảng 10: Lệnh client: Tải tài liệu từ cộng đồng

3.1.2.e Quản lý kho tài liệu cá nhân (*mở rộng*)

Mục đích: Client được phép xem thông tin các tài liệu trong kho dữ liệu cá nhân (local repo) mình đã công bố/hủy công bố hoặc tải về từ cộng đồng thông qua giao diện hỗ trợ của Min.io, với các thông tin về:

- Tên tài liệu
- Kích thước tài liệu
- Lần chỉnh sửa cuối cùng

3.2 Yêu cầu phi tính năng

3.2.1 Hiệu suất hoạt động

Tại mỗi thời điểm, nhiều client có thể yêu cầu tải các tài liệu khác nhau trên cùng một client mục tiêu. Điều này đòi hỏi thao tác truyền/nhận tài liệu được xử lý đa luồng.

3.2.2 Khả năng tương thích

Ứng dụng sẽ được đóng gói bởi Docker để có thể chạy trên nhiều thiết bị máy tính cá nhân của người dùng.

3.3 Yêu cầu giao thức

Ứng dụng sử dụng hoàn toàn giao thức **HTTP** ở tầng Application để truyền/nhận dữ liệu hệ thống. Các port được sử dụng để hiện thực ứng dụng bao gồm:

- **Đối với phía Client:**

1. Giao diện chính: port 3000
2. Giao diện hỗ trợ (Min.io): port 9090
3. Service Back-end: port 8080
4. Service Local repo (Min.io): port 9000

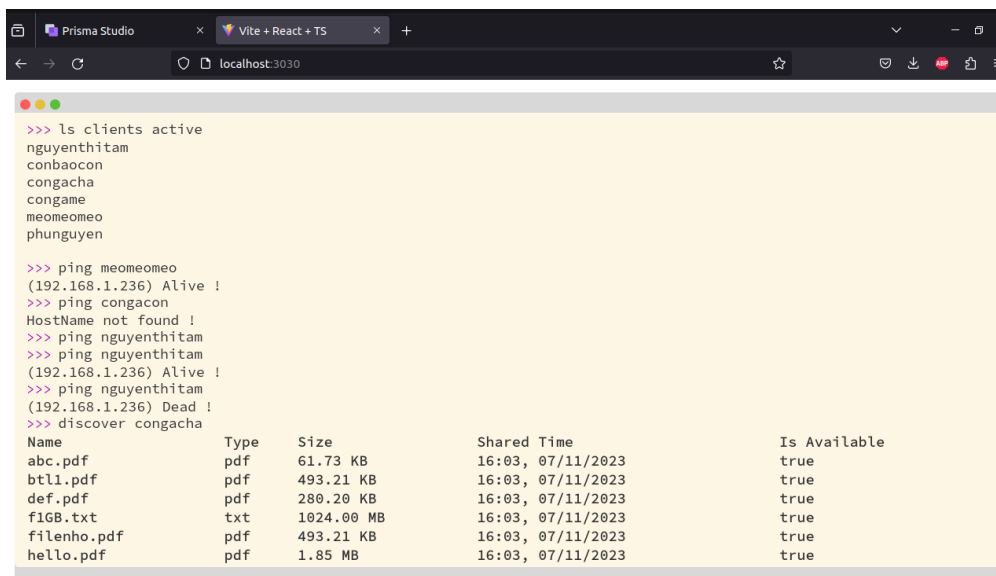
- **Đối với phía Server:**

1. Giao diện chính: port 3030
2. Giao diện hỗ trợ (Prisma): port 5555
3. Service Back-end: port 8000
4. Service Database (Prisma): port 5432

3.4 Yêu cầu giao diện

3.4.1 Server

- Giao diện chính (port 3030): GUI mô phỏng hành vi CLI để thao tác các lệnh ứng dụng.



```
>>> ls clients active
nguyenthitam
conbacon
congacha
congame
meomeomeo
phunguyen

>>> ping meomeomeo
(192.168.1.236) Alive !
>>> ping congacon
HostName not found !
>>> ping nguyenthitam
>>> ping nguyenthitam
(192.168.1.236) Alive !
>>> ping nguyenthitam
(192.168.1.236) Dead !
>>> discover congacha

Name                Type      Size           Shared Time           Is Available
abc.pdf              pdf       61.73 KB       16:03, 07/11/2023     true
bt11.pdf             pdf       493.21 KB      16:03, 07/11/2023     true
def.pdf              pdf       280.20 KB      16:03, 07/11/2023     true
f1GB.txt             txt       1024.00 MB     16:03, 07/11/2023     true
filenho.pdf          pdf       493.21 KB      16:03, 07/11/2023     true
hello.pdf            pdf       1.85 MB        16:03, 07/11/2023     true
```

Hình 10: Giao diện: Màn hình chính server

- Giao diện đi kèm của Prisma (port 5555): Hiển thị cơ sở dữ liệu tập trung.

Prisma Studio

Vite + React + TS

localhost:5555

SharedDocument

User

Session

Filters

None

Fields

All

Showing

100 of 117

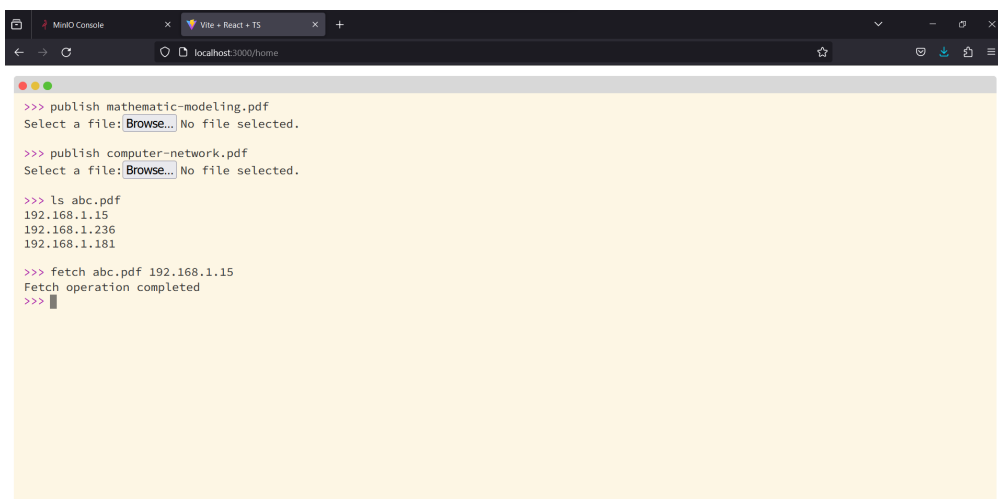
Add record

	id	A	name	A	type	A	size	#	sharedTime	#	isAvailable	↗	session
	cloirktw20003oxdq57g...	2222	pdf	pdf	588021	1699025927	true						Session
	cloiuu06k0005ox328kyt...	2353	pdf	pdf	171618	1699030841	true						Session
	clojnj3dt0004oxikbxvr...	1248	pdf	pdf	305	1699085342	true						Session
	clojml0g20008oxik4oea...	1248	pdf	pdf	131555	1699084547	true						Session
	clojs3l6s0003oxfj5wha...	1248	pdf	pdf	131555	1699087089	true						Session
	clojtr3xo0003ox50qwt2...	image	jpg	jpg	61632	1699091558	true						Session
	clojyd5lc0001oxhh1qui...	db	pdf	pdf	4503353	1699096800	true						Session
	clok071ys0003oxhpe5k...	1911	pdf	pdf	131555	1699099896	true						Session
	clolb6o950000ox7igatz...	1248	pdf	pdf	131555	1699178798	false						Session
	clolb6o960001ox7i0tp6...	1911	pdf	pdf	131555	1699178798	false						Session
	clolb6o960002ox7ihjbd...	db	pdf	pdf	4503353	1699179436	false						Session
	clolb6o960003ox7i4oiw...	image	jpg	jpg	61632	1699192977	false						Session
	clolbip12000qox7i7evh...	1248	pdf	pdf	131555	1699179360	true						Session
	clolbip12000rox7ixn7z...	1911	pdf	pdf	131555	1699179360	true						Session
	clolbip12000sox7iar8w...	db	pdf	pdf	4503353	1699179360	true						Session
	clolbip12000tox7im9kg...	image	jpg	jpg	61632	1699179360	true						Session

Hình 11: Giao diện: Cơ sở dữ liệu tập trung

3.4.2 Client

- Giao diện chính (port 3000): GUI mô phỏng hành vi CLI để thao tác các lệnh ứng dụng.



```
>>> publish mathematic-modeling.pdf
Select a file: Browse... No file selected.

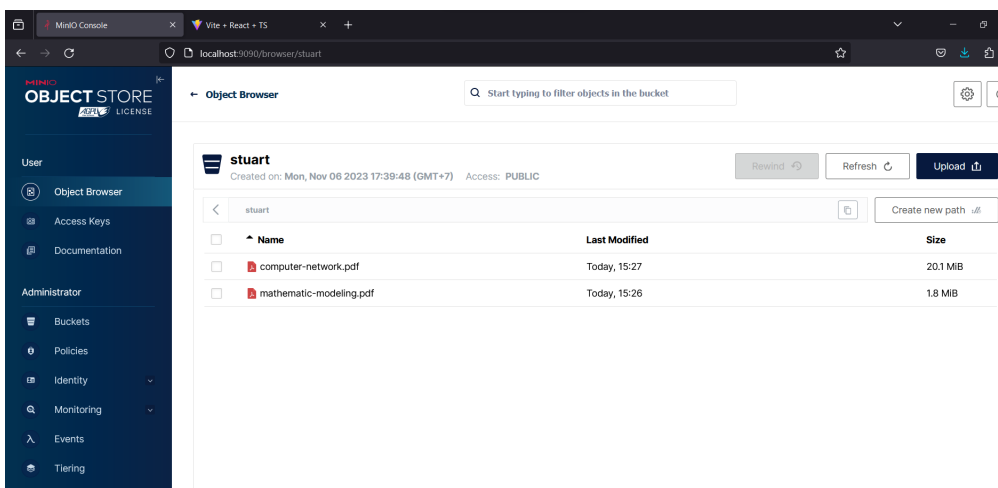
>>> publish computer-network.pdf
Select a file: Browse... No file selected.

>>> ls abc.pdf
192.168.1.15
192.168.1.236
192.168.1.181

>>> fetch abc.pdf 192.168.1.15
Fetch operation completed
>>>
```

Hình 12: Giao diện: Màn hình chính client

- Giao diện đi kèm của Min.io (port 9090): Hiển thị thông tin tài liệu trong kho tài liệu cá nhân đã công bố (local repo).



Hình 13: Giao diện: Kho tài liệu cá nhân

3.5 Yêu cầu dữ liệu

3.5.1 Lưu trữ tài liệu

- **Nơi lưu trữ:** Tại mỗi client
- **Mục đích:** Lưu trữ tài liệu bản gốc, các tài liệu tải về từ hệ thống để giảm dung lượng lưu trữ tại server. Kích thước tối đa của mỗi tài liệu là: **1GB**, tổng kích thước sẽ phụ thuộc vào ổ đĩa của người dùng.
- **Thời lượng lưu trữ:** Mỗi khi người dùng đăng nhập vào hệ thống, ứng dụng sẽ kiểm tra và đăng tải các tài liệu được lưu trong local repo lên server. Nếu người dùng bị thoát khỏi hệ thống (do lỗi phát sinh) mà chưa kịp thoát phiên đăng nhập thì tài liệu local repo vẫn sẽ tồn tại.
- **Tính khả dụng:** Toàn bộ dữ liệu được lưu trữ trong local repo sẽ gắn với thiết bị, không phải phiên đăng nhập. Nếu người dùng vẫn còn khả dụng trên hệ thống thì tất cả các tài liệu được công bố sẽ được phép tải về.

Dữ liệu	Mô tả chung
Thông tin chung	Tên tài khoản, địa chỉ IP và phiên hoạt động hiện tại của người dùng sau khi đăng nhập.
Tài liệu chia sẻ	Tên, loại, kích thước, thời điểm chia sẻ và chính tài liệu đó trong local repository.

Bảng 11: Lưu trữ tài liệu

3.5.2 Lưu trữ thông tin hệ thống

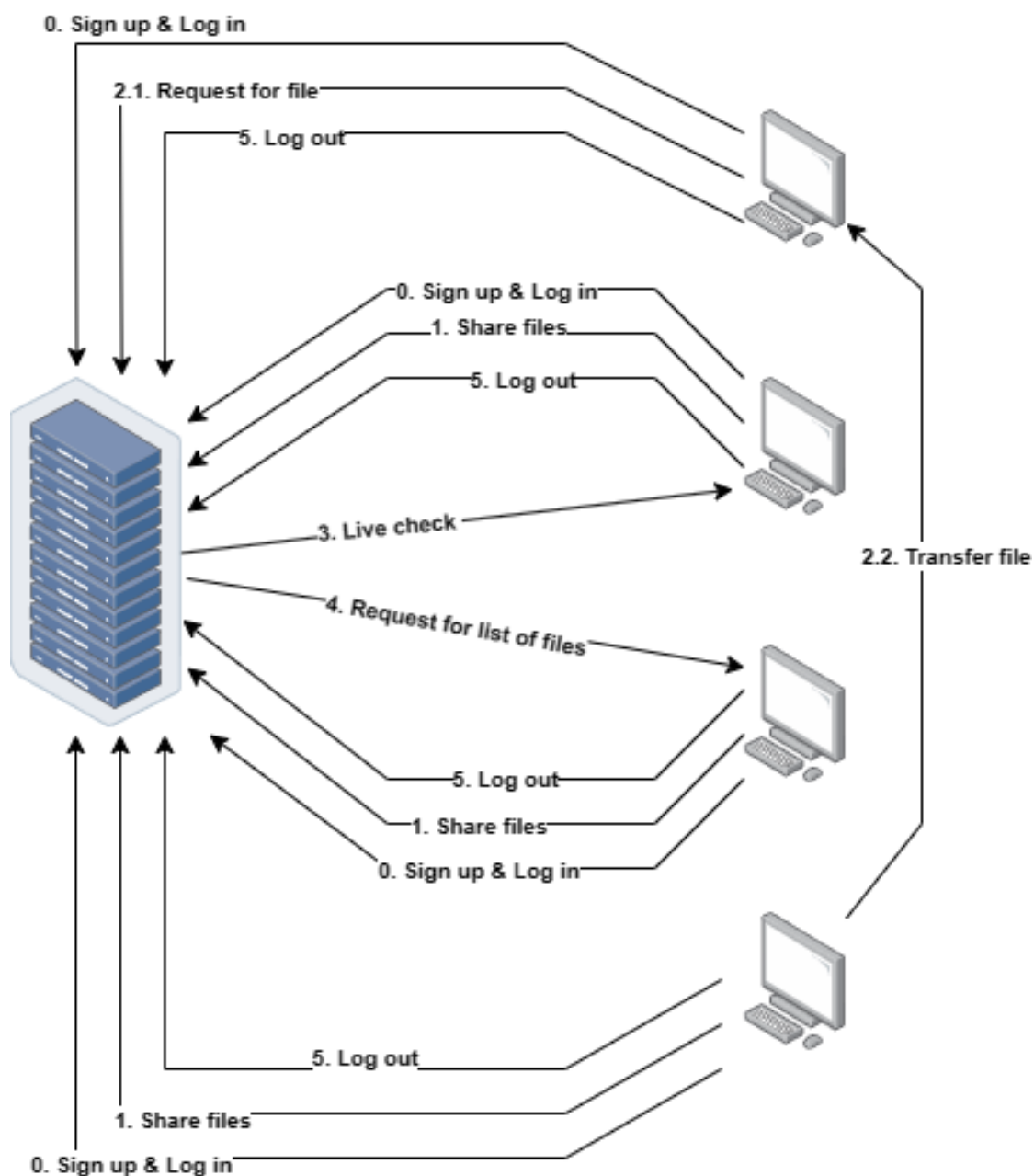
- **Nơi lưu trữ:** Server
- **Mục đích:** Lưu trữ thông tin về hoạt động trong hệ thống nhằm quản lý/vận hành, nhưng không trực tiếp lưu trữ tài liệu.
- **Thời lượng lưu trữ:** Cơ sở dữ liệu tập trung sẽ được khởi tạo mỗi lần thiết lập server và xóa khi server ngừng hoạt động. Hệ thống sẽ cập nhật thường xuyên về tính khả dụng của client và tài liệu được công bố lên hệ thống. Mỗi client sẽ gắn với danh sách tài liệu được công bố, nếu client thoát khỏi hệ thống (đăng xuất, mất kết nối) thì hệ thống sẽ đánh dấu cả client và tài liệu không còn khả dụng.

Dữ liệu	Mô tả chung
Tài khoản người dùng	Tên tài khoản, mật khẩu, tên đầy đủ của người dùng và thông tin kết nối tới phiên hoạt động của tài khoản. Người dùng sẽ cần yêu cầu tạo một tài khoản mới, xác thực danh tính thông qua đăng nhập để truy cập hệ thống. Tại mỗi thời điểm, mỗi tài khoản chỉ có thể hoạt động trên một thiết bị (một phiên hoạt động).
Phiên hoạt động	Thông tin về tài khoản liên kết với phiên đăng nhập, địa chỉ IP và thời điểm bắt đầu/kết thúc phiên. Khi đăng nhập, người dùng sẽ khởi tạo một phiên đăng nhập mới, hệ thống sẽ cập nhật địa chỉ IP nếu thông tin
Tài liệu chia sẻ	Tên, loại, kích thước, thời điểm chia sẻ và ID của phiên hoạt động đã chia sẻ nhằm lưu trữ lại lịch sử các tài liệu đã chia sẻ trên hệ thống.

Bảng 12: Lưu trữ thông tin hệ thống

4 THIẾT KẾ HỆ THỐNG

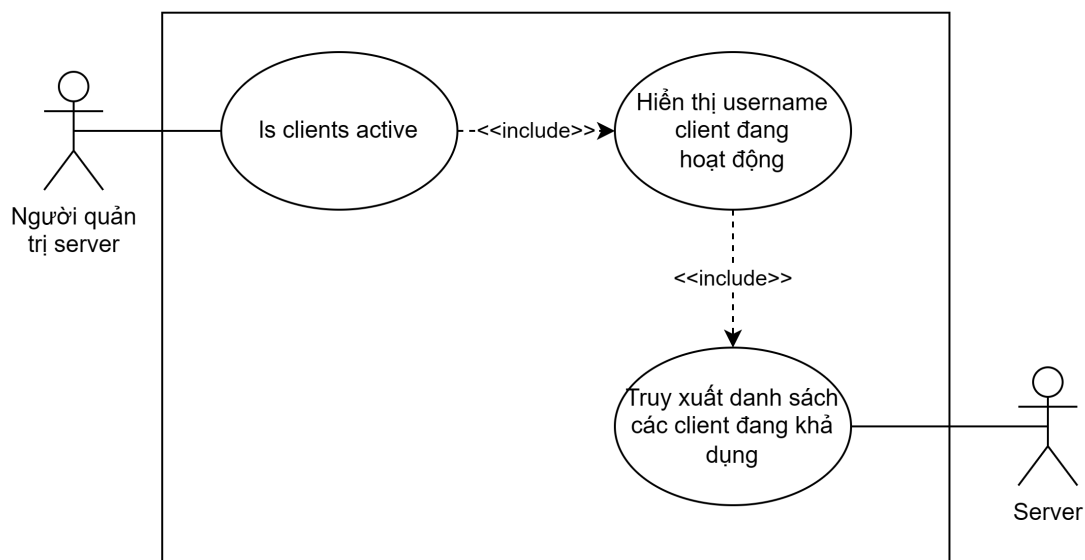
4.1 Tổng quan hệ thống



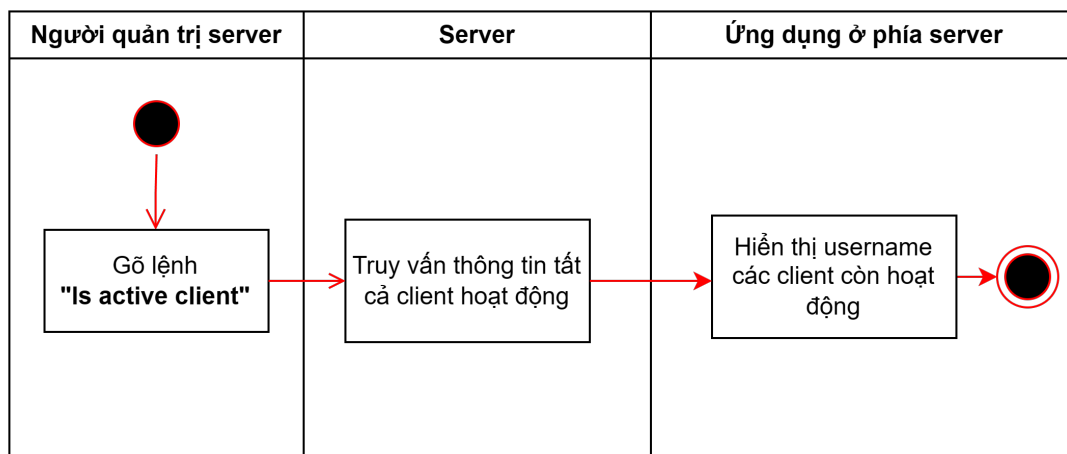
Hình 14: Mô hình hệ thống ở mức ý niệm của nhóm

4.2 Server

4.2.1 Truy xuất thông tin client (mở rộng)



Hình 15: Use case: Truy xuất thông tin client

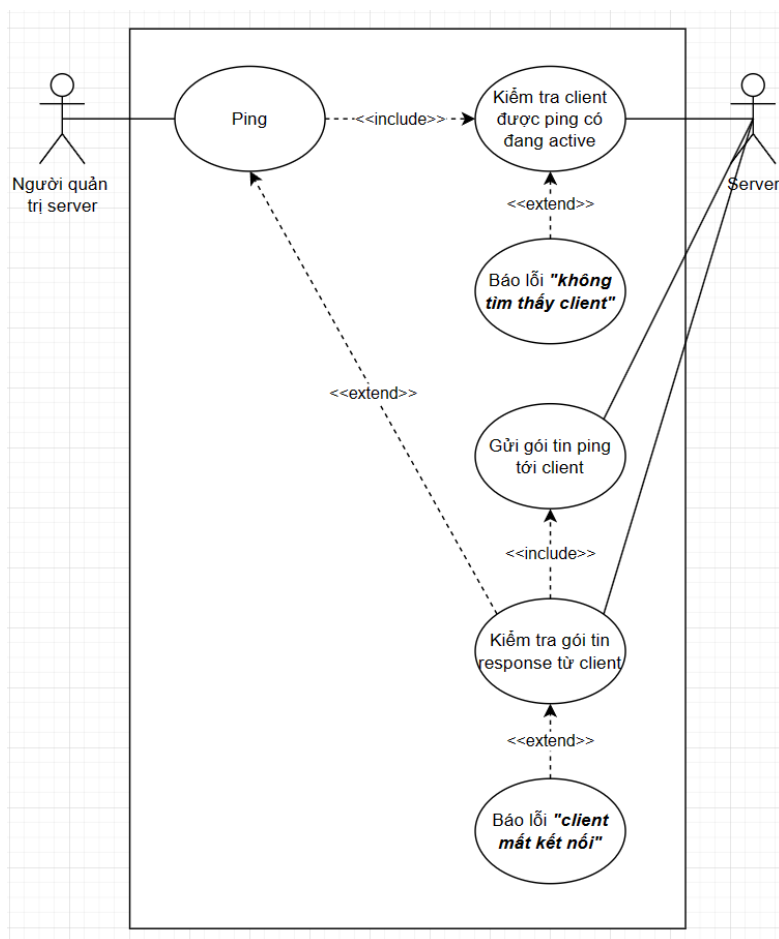


Hình 16: Activity diagram: Truy xuất thông tin client

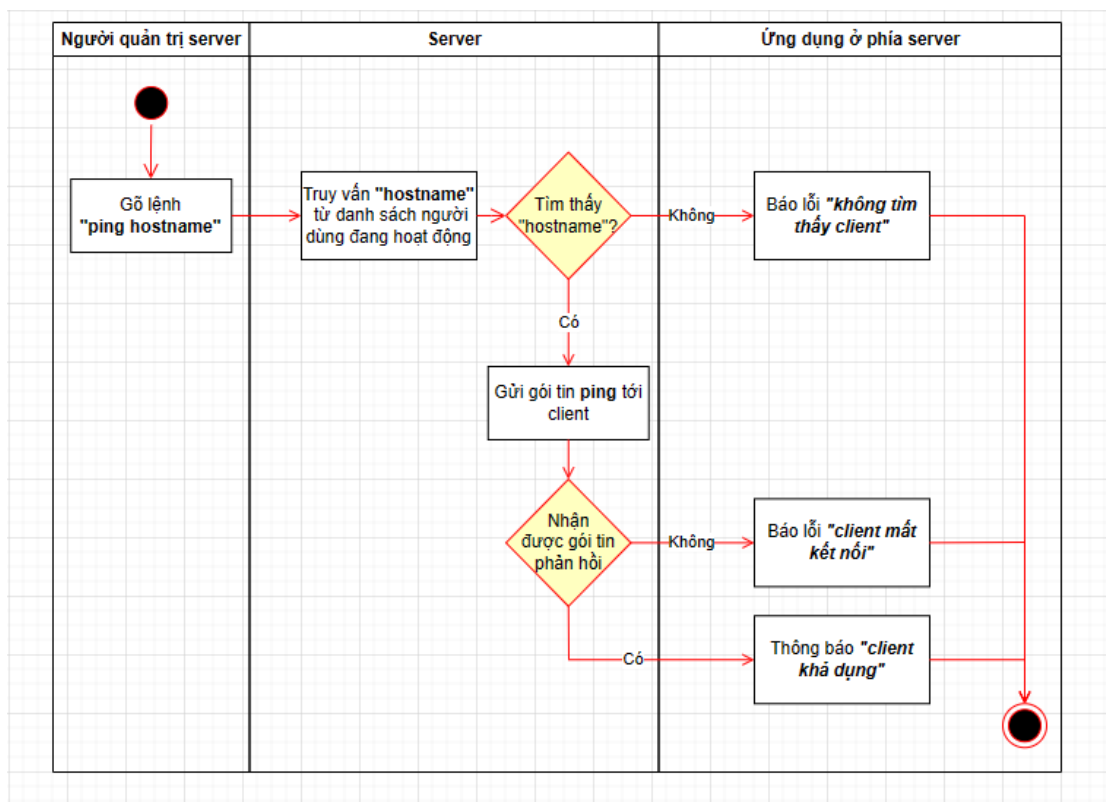
Đặc tả chi tiết:

- **Precondition:** Không
- **Postcondition:** Danh sách các username còn hoạt động trên hệ thống
- **Basic path:**
 1. Thông qua lệnh **ls clients active** trên CLI, người quản trị sẽ tìm kiếm username của tất cả các client vẫn đang hoạt động.
 2. Server truy xuất dữ liệu và hiển thị trên CLI.

4.2.2 Kiểm tra tính khả dụng của client



Hình 17: Use case: kiểm tra tính khả dụng của client

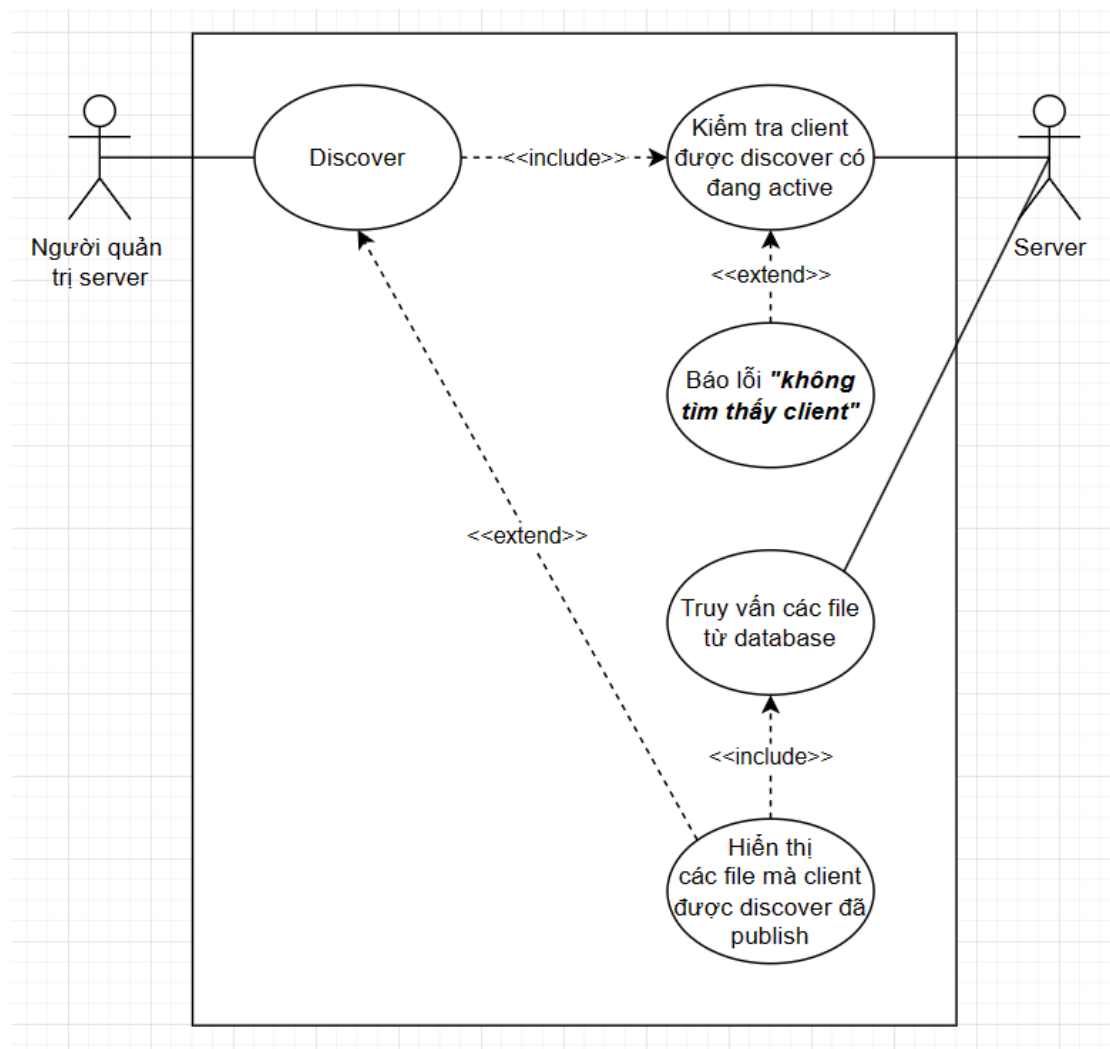


Hình 18: Activity diagram: kiểm tra tính khả dụng của client

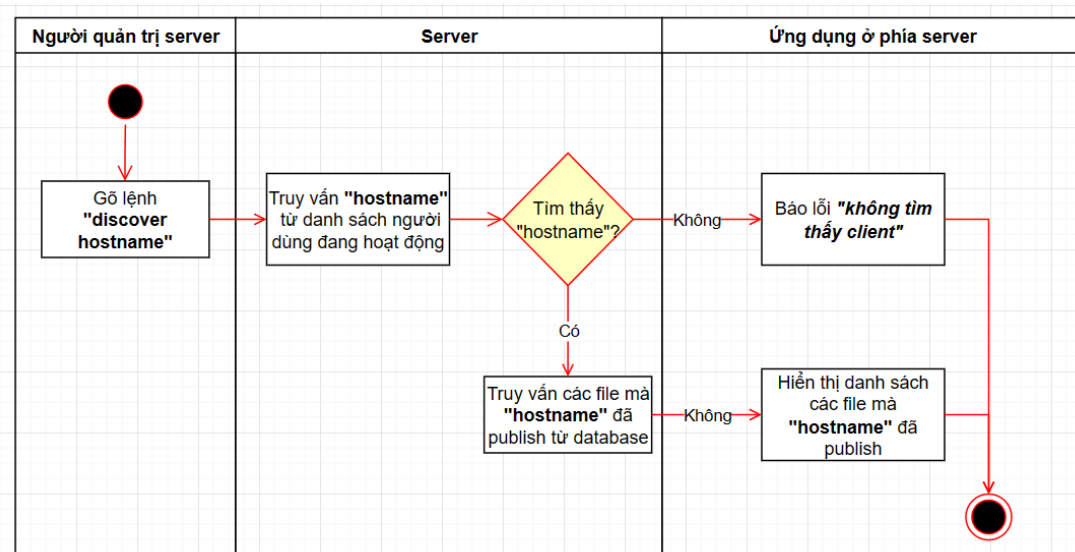
Đặc tả chi tiết:

- **Precondition:** Không
- **Postcondition:** Kết quả tính khả dụng của client được cập nhật lên hệ thống
- **Basic path:** Kiểm tra tính khả dụng client
 1. Thông qua lệnh **ping hostname** trên CLI, người quản trị sẽ tìm kiếm địa chỉ IP client dựa trên **hostname**.
 2. Server gửi tín hiệu tới client để kiểm tra tính khả dụng của client.
 3. Client nhận được thông báo sẽ phản hồi với server.
 4. Server nhận được phản hồi sẽ cập nhật thông tin client đang hoạt động trên cơ sở dữ liệu tập trung. Nếu sau một khoảng thời gian, server vẫn không nhận được tín hiệu phản hồi thì sẽ đăng xuất client khỏi hệ thống, kết thúc phiên hoạt động.

4.2.3 Truy xuất tài liệu chia sẻ



Hình 19: Use case diagram cho lệnh tính năng truy xuất tài liệu chia sẻ



Hình 20: Activity diagram: truy xuất tài liệu chia sẻ

Đặc tả chi tiết:

- **Precondition:** Client cần truy xuất khả dụng.
- **Postcondition:** Hiển thị danh sách các tài liệu được lưu trữ cục bộ.
- **Basic path:**
 1. Thông qua lệnh **discover** hostname trên CLI, người quản lý sẽ yêu cầu truy xuất tất cả tài liệu còn khả dụng của client **hostname**.
 2. Server sẽ truy xuất thông tin về tài liệu trên cơ sở dữ liệu tập trung dựa theo **hostname** và trả về hiển thị trên giao diện của người quản lý (nếu **hostname** còn khả dụng).
- **Exceptional path:** Nếu **hostname** cần truy xuất không tồn tại thì hệ thống sẽ thông báo cho server và hủy yêu cầu truy xuất.

4.2.4 Quản lý cơ sở dữ liệu tập trung (mở rộng)

Đặc tả chi tiết:

Người quản lý sẽ có quyền truy cập trực tiếp tới tất cả các dữ liệu được lưu trữ trên cơ sở dữ liệu tập trung, thông qua giao diện hỗ trợ của Prisma. Cơ sở dữ liệu sẽ tập trung lưu trữ thông tin để vận hành hệ thống, không bao gồm tài liệu gốc được công bố.

Dữ liệu	Loại	Mô tả	Điều kiện
ID	String	Khóa chính	
username	String	Tên đăng nhập	Độc nhất
password	String	Mật khẩu	
fullName	String	Họ và tên người dùng	
isAvailable	Boolean	Tính khả dụng của tài liệu	Mặc định: False
session	Session[]	Danh sách tất cả phiên hoạt động của người dùng	

Bảng 13: Data: User

Dữ liệu	Loại	Mô tả	Điều kiện
ID	String	Khóa chính	
loginTime	Number	Thời điểm đăng nhập	
logoutTime	Number	Thời điểm đăng xuất	Nếu <i>null</i> sẽ biểu thị người dùng đang hoạt động
ipAddress	String	Địa chỉ IP mới nhất trong phiên đăng nhập	Hệ thống cập nhật mỗi lần client tương tác
userId	String	Người dùng liên kết tới phiên đăng nhập	
sharedDocument	SharedDocument[]	Danh sách tất cả tài liệu được người dùng chia sẻ trong phiên đăng nhập	

Bảng 14: Data: Session

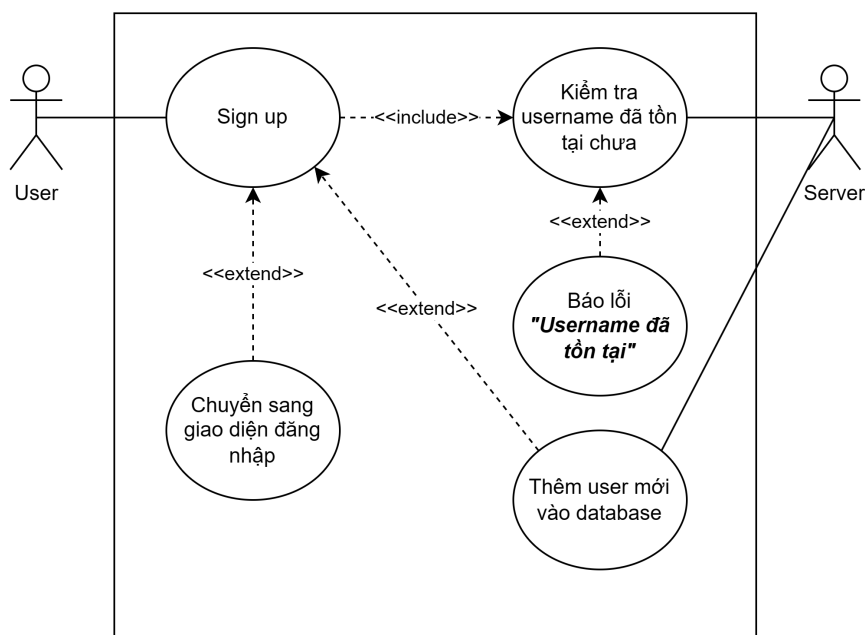
Dữ liệu	Loại	Mô tả	Điều kiện
ID	String	Khóa chính	
Name	String	Tên tài liệu fname	Nhiều tài liệu trùng tên, nhưng không được cùng trùng phiên hoạt động
Type	String	Loại tài liệu	
Size	Number	Kích thước tài liệu	
SharedTime	Number	Thời gian gần nhất tài liệu với tên fname được chia sẻ trong phiên hoạt động	
isAvailable	Boolean	Tính khả dụng của tài liệu	Mặc định: True
sessionId	String	Phiên hoạt động liên kết với tài liệu	

Bảng 15: Data: SharedDocument

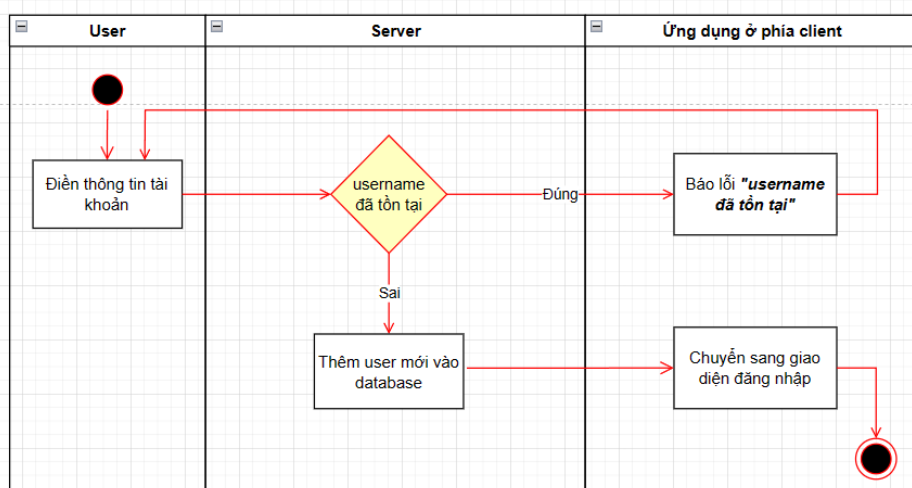
4.3 Client

4.3.1 Tạo tài khoản/Đăng nhập/Đăng xuất

4.3.1.a Tạo tài khoản



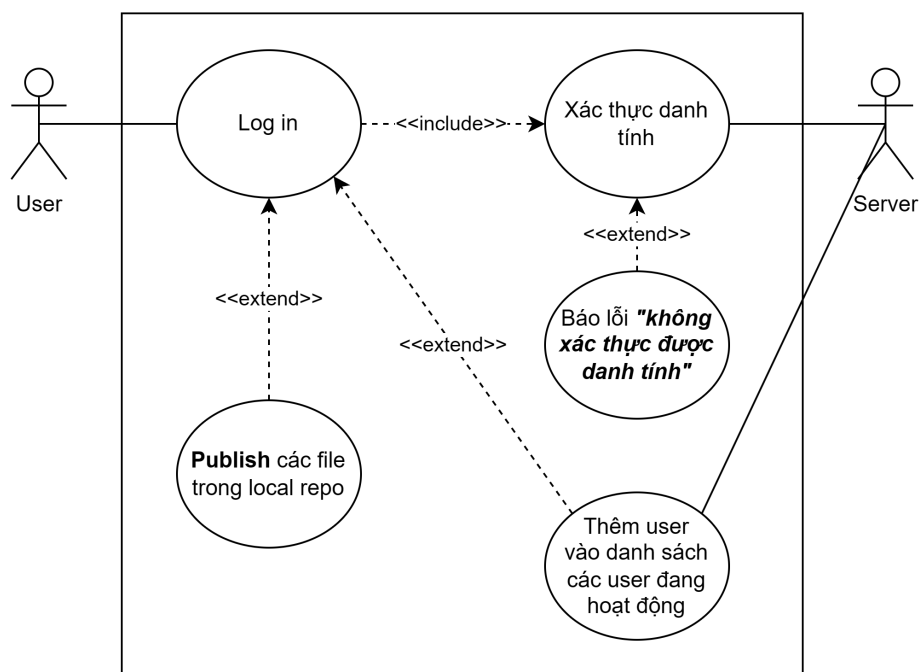
Hình 21: Use case: Tạo tài khoản



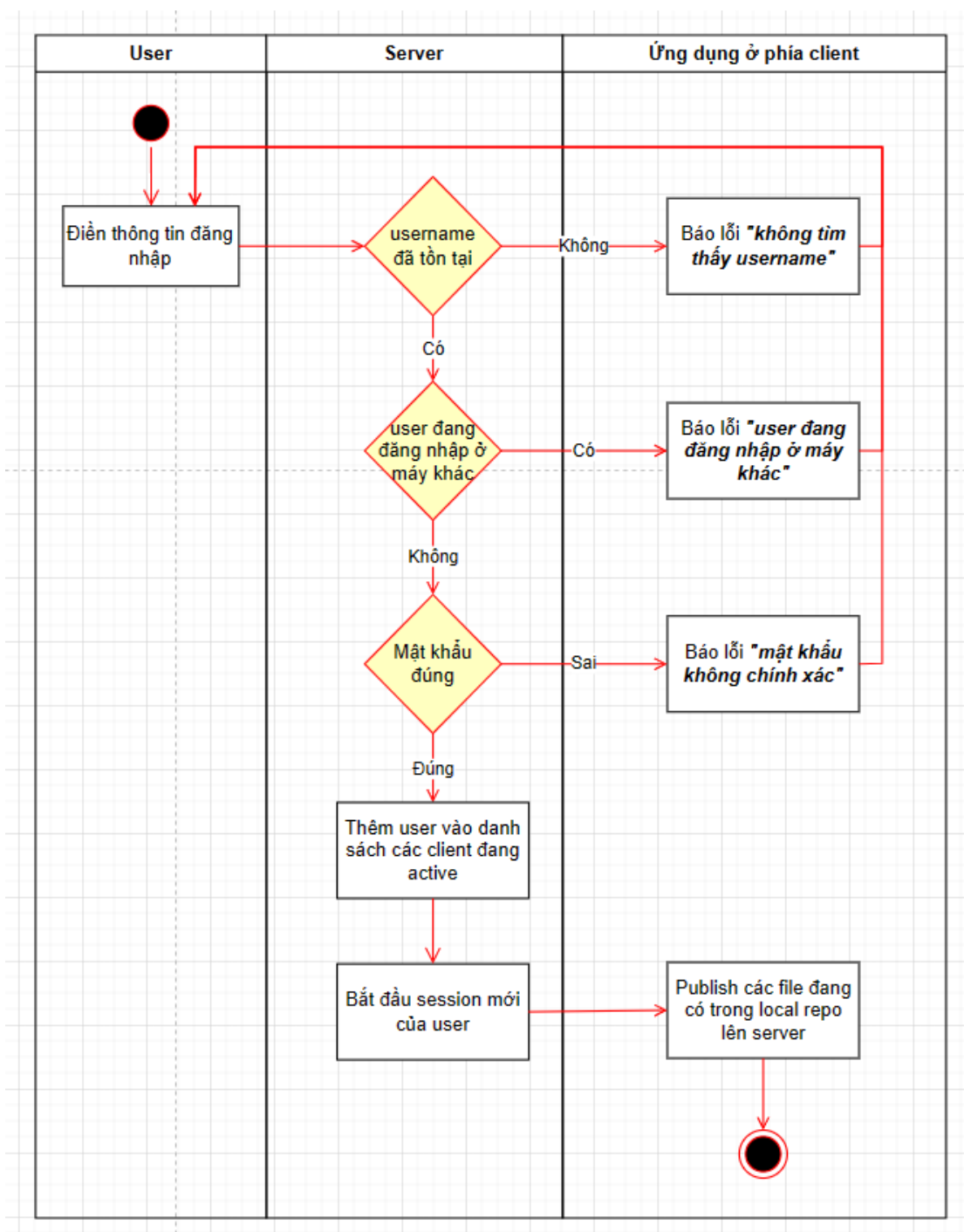
Hình 22: Activity diagram: Tạo tài khoản

- **Precondition:** Thông tin tài khoản chưa tồn tại trong hệ thống.
- **Postcondition:** Lưu trữ thông tin tài khoản, điều hướng người dùng tới giao diện đăng nhập.
- **Basic path:**
 1. Người dùng nhập thông tin muốn đăng ký: fullName, username, password.
 2. Hệ thống xác thực thông tin tài khoản hợp lệ, kiểm tra nếu tài khoản đã tồn tại trước đó và lưu trữ thông tin tài khoản mới vào cơ sở dữ liệu. Tài khoản được tạo thành công, hiển thị thông báo cho người dùng.
 3. Điều hướng người dùng quay trở lại giao diện đăng nhập.
- **Exceptional path:** Tại bước thứ 2, nếu tài khoản đã tồn tại trong hệ thống thì sẽ hiển thị thông báo cho người dùng và điều hướng trở lại giao diện đăng nhập.

4.3.1.b Đăng nhập



Hình 23: Use case: Đăng nhập

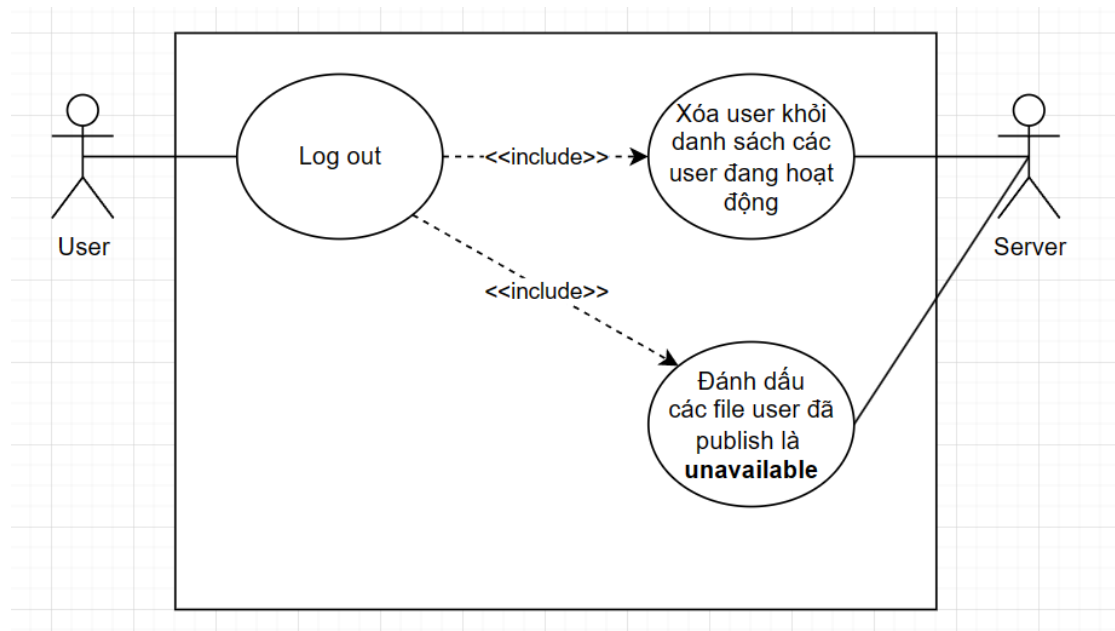


Hình 24: Activity diagram: Đăng nhập

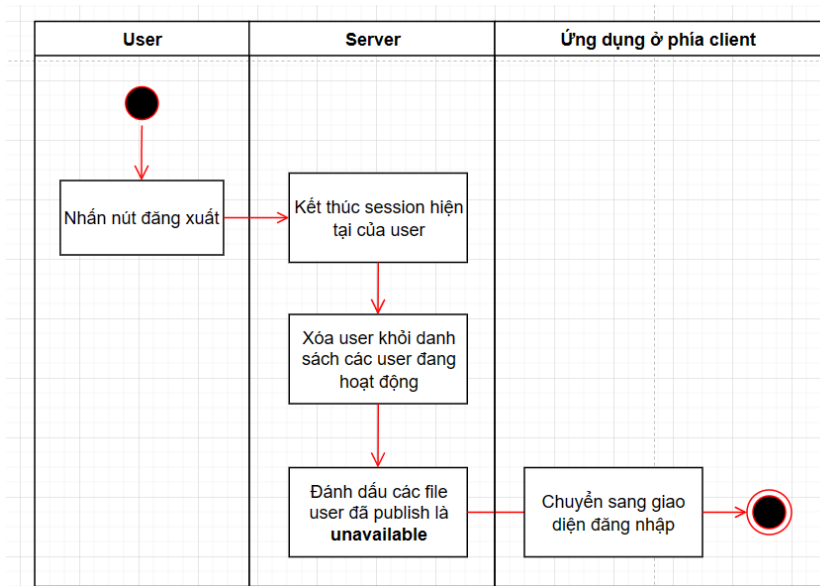
Đặc tả chi tiết:

- **Precondition:** Thông tin đăng nhập tồn tại.
- **Postcondition:** Danh tính người dùng được xác thực, khởi động phiên hoạt động mới và điều hướng vào giao diện chính của hệ thống.
- **Basic path:**
 1. Người dùng nhập thông tin tài khoản để đăng nhập: tên đăng nhập, mật khẩu.
 2. Hệ thống kiểm tra tính chính xác của thông tin đăng nhập.
 3. Sau khi xác nhận tài khoản hợp lệ, hệ thống sẽ tiếp tục kiểm tra nếu đang có thiết bị khác đăng nhập bằng tài khoản trên.
 - Nếu thiết bị vẫn đang hoạt động: Yêu cầu đăng nhập của người dùng sẽ bị từ chối. Đăng nhập thất bại, hiển thị thông báo cho người dùng. Hệ thống sẽ điều hướng người dùng trở lại trang đăng nhập.
 - Nếu thiết bị không còn hoạt động: Hệ thống sẽ đăng xuất thiết bị cũ, đăng nhập thiết bị mới. Đăng nhập thành công, hiển thị thông báo cho người dùng. Hệ thống điều hướng người dùng tới trang chính của ứng dụng. Hệ thống kiểm tra tất cả tài liệu đang lưu trữ trong local repo và công bố lên server.
- **Exceptional path:** Tại bước thứ 2, nếu thông tin đăng nhập không hợp lệ thì sẽ hiển thị thông báo cho người dùng, đăng nhập thất bại.

4.3.1.c Đăng xuất



Hình 25: Use case: Đăng xuất



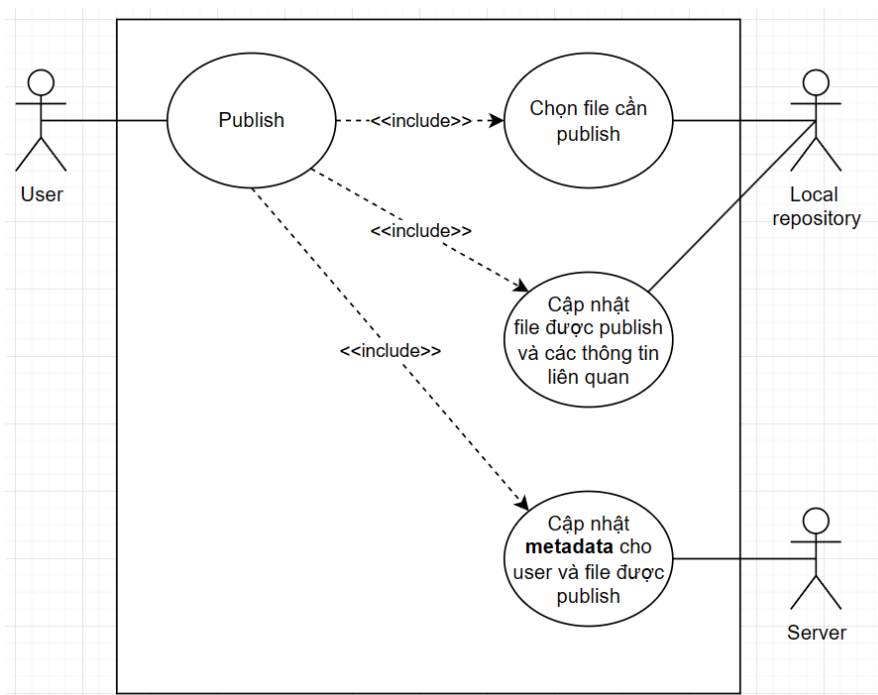
Hình 26: Activity diagram: Đăng xuất

Đặc tả chi tiết:

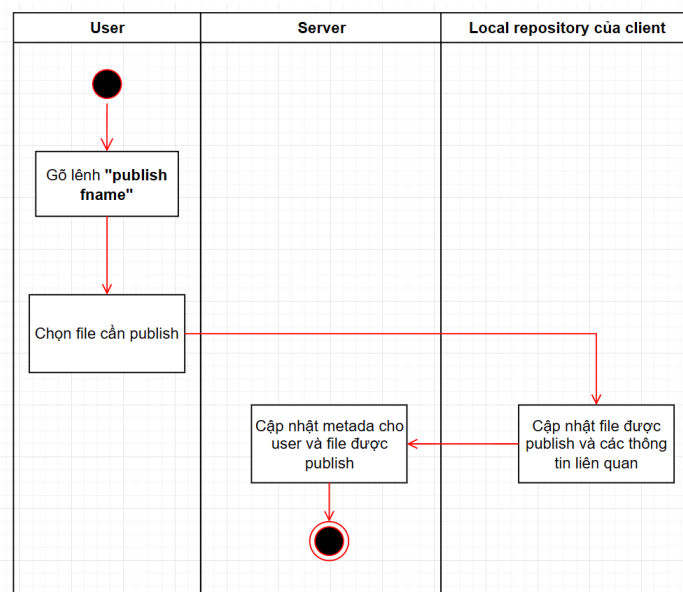
- **Precondition:** Người dùng vẫn trong phiên đăng nhập
- **Postcondition:** Kết thúc phiên đăng nhập của người dùng, hủy công bố các tài liệu trên hệ thống.
- **Basic path:**
 1. Người dùng lựa chọn đăng xuất khỏi hệ thống.
 2. Hệ thống đánh dấu thời gian kết thúc của phiên hoạt động, đánh dấu các tài liệu được công bố của client không còn khả dụng và đánh dấu chính client không còn khả dụng.
 3. Đăng xuất thành công, hiển thị thông báo cho người dùng. Hệ thống điều hướng người dùng trở lại trang đăng nhập.

4.3.2 Công bố/Hủy công bố tài liệu

4.3.2.a Công bố tài liệu



Hình 27: Use case: Công bố tài liệu

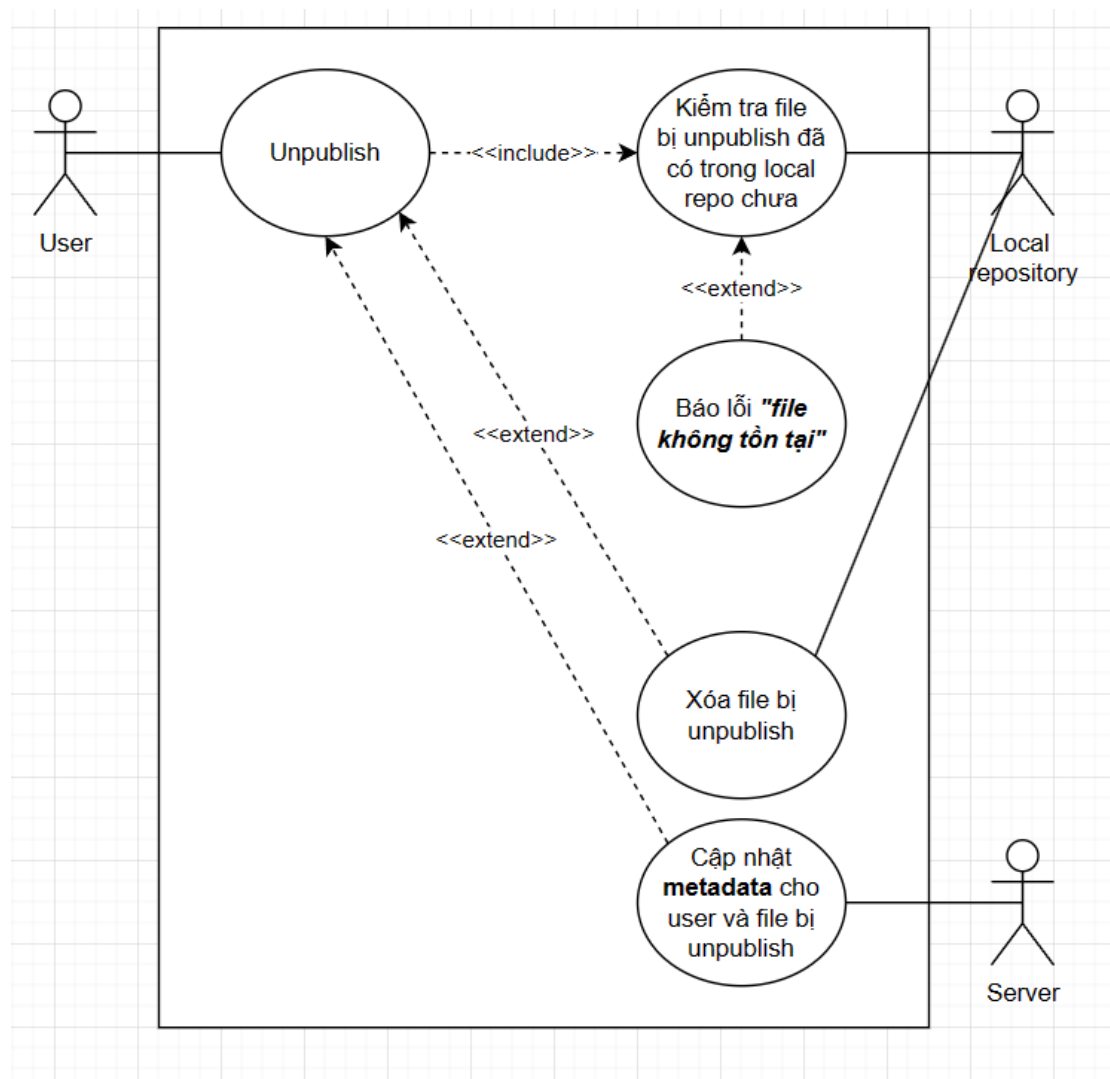


Hình 28: Activity diagram: Công bố tài liệu

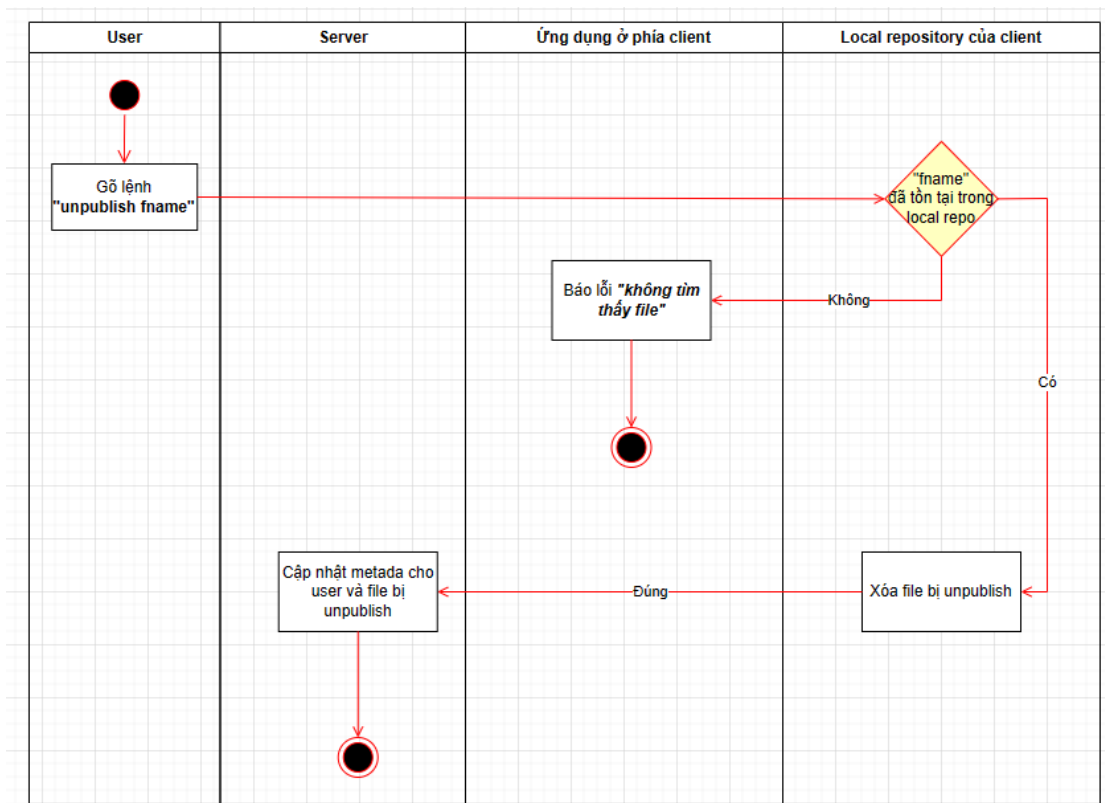
Đặc tả chi tiết:

- **Precondition:** Người dùng vẫn trong phiên đăng nhập và có tài liệu cần chia sẻ.
- **Postcondition:** Tài liệu được công bố, có thể được tải xuống bởi một client khác
- **Basic path:**
 1. Thông qua lệnh **publish fname**, người dùng lựa chọn tài liệu muốn chia sẻ cho hệ thống, lưu dưới tên **fname**.
 - Nếu không có tài liệu trùng tên trên local repo: Ứng dụng sẽ tạo một bản sao chép của tài liệu vào local repo của người dùng và gửi thông tin của tài liệu lên hệ thống: tên, loại và kích thước tài liệu.
 - Nếu có tài liệu trùng tên trên local repo, hệ thống sẽ ghi đè dữ liệu.
 2. Hệ thống cập nhật tài liệu thành tài liệu khả dụng, liên kết với user tương ứng.
- **Exceptional path:** Tại bước 2, nếu client đã từng chia sẻ 1 tài liệu trùng tên, hệ thống sẽ cho phép người dùng lựa chọn ghi đè dữ liệu cũ hoặc không thực hiện lệnh công bố tài liệu này.

4.3.2.b Hủy công bố tài liệu



Hình 29: Use case: Hủy công bố tài liệu



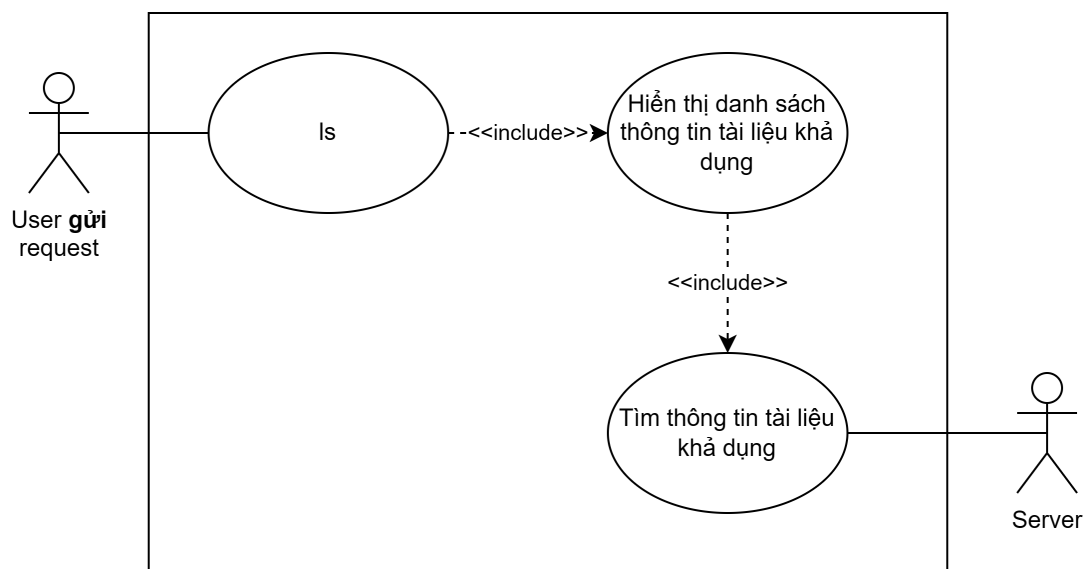
Hình 30: Activity diagram: hủy công bố tài liệu

Đặc tả chi tiết:

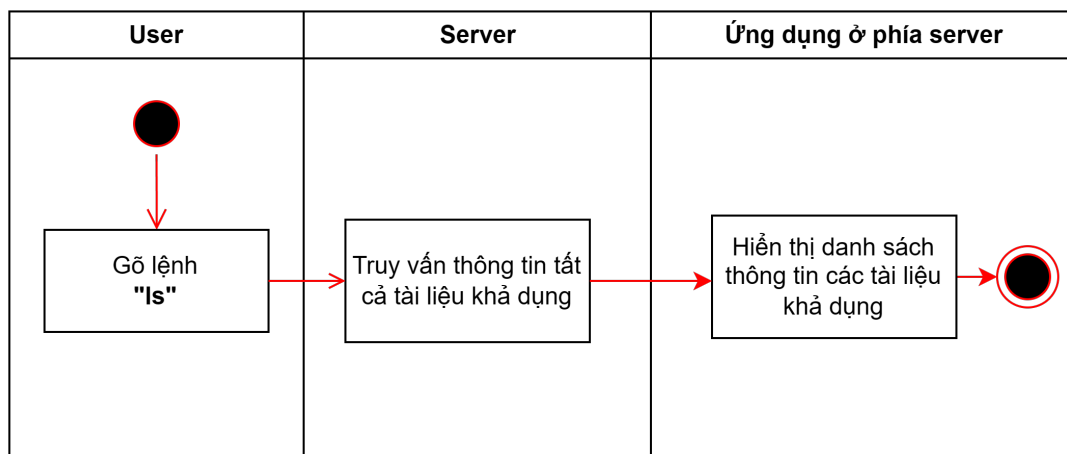
- **Precondition:** Thông tin tài liệu client công bố tồn tại trên hệ thống.
- **Postcondition:** Thông tin tài liệu không còn khả dụng trên hệ thống.
- **Basic path:**
 1. Thông qua lệnh **unpublish fname**, người dùng lựa chọn tài liệu đã chia sẻ trước đó để hủy chia sẻ tài liệu.
 2. Hệ thống kiểm tra thông tin tài liệu trên local repo, gửi yêu cầu cập nhật tính khả dụng của tài liệu lên server và xóa tài liệu khỏi local repo. Hủy công bố tài liệu thành công, hiển thị thông báo cho người dùng.
- **Exceptional path:** Tại bước 2, nếu hệ thống không truy xuất được thông tin tài liệu thì sẽ hủy yêu cầu và gửi thông báo cho người dùng.

4.3.3 Tìm kiếm client công bố tài liệu

4.3.3.a Truy xuất tài liệu khả dụng



Hình 31: Use case: Truy xuất tài liệu khả dụng

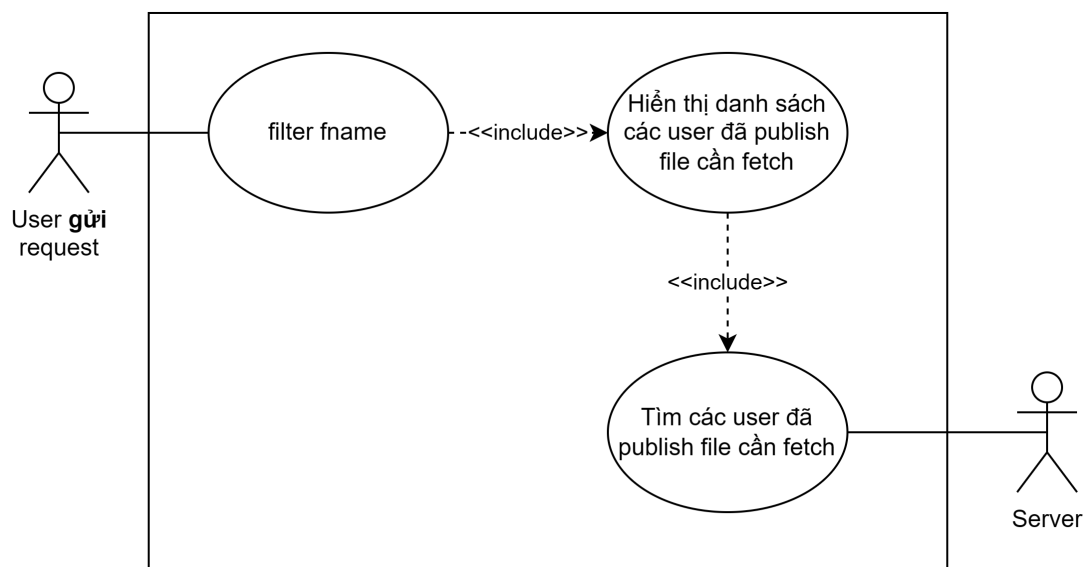


Hình 32: Activity diagram: Truy xuất tài liệu khả dụng

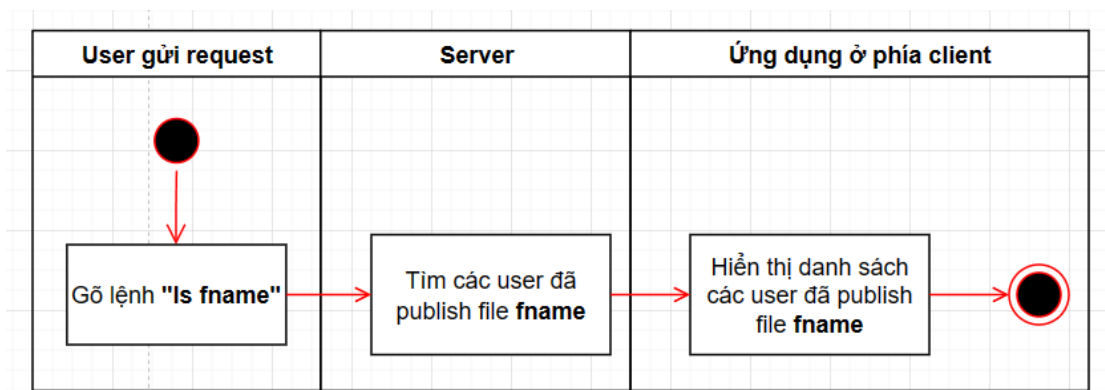
Đặc tả chi tiết:

- **Precondition:** Không
- **Postcondition:** Danh sách thông tin tất cả các tài liệu đang khả dụng
- **Basic path:**
 1. Thông qua lệnh **ls** để tìm kiếm thông tin tất cả các tài liệu còn khả dụng
 2. Hệ thống hiển thị thông tin tất cả các tài liệu khả dụng với: tên tài liệu, username, địa chỉ IP, kích thước và loại tài liệu chia sẻ, thời gian công bố
- **Exceptional path:** Tại bước 2, nếu hệ thống không truy xuất được thông tin tài liệu thì sẽ không hiển thị dữ liệu.

4.3.3.b Truy xuất thông tin client theo tên tài liệu



Hình 33: Use case: Tìm kiếm tài liệu bằng tên

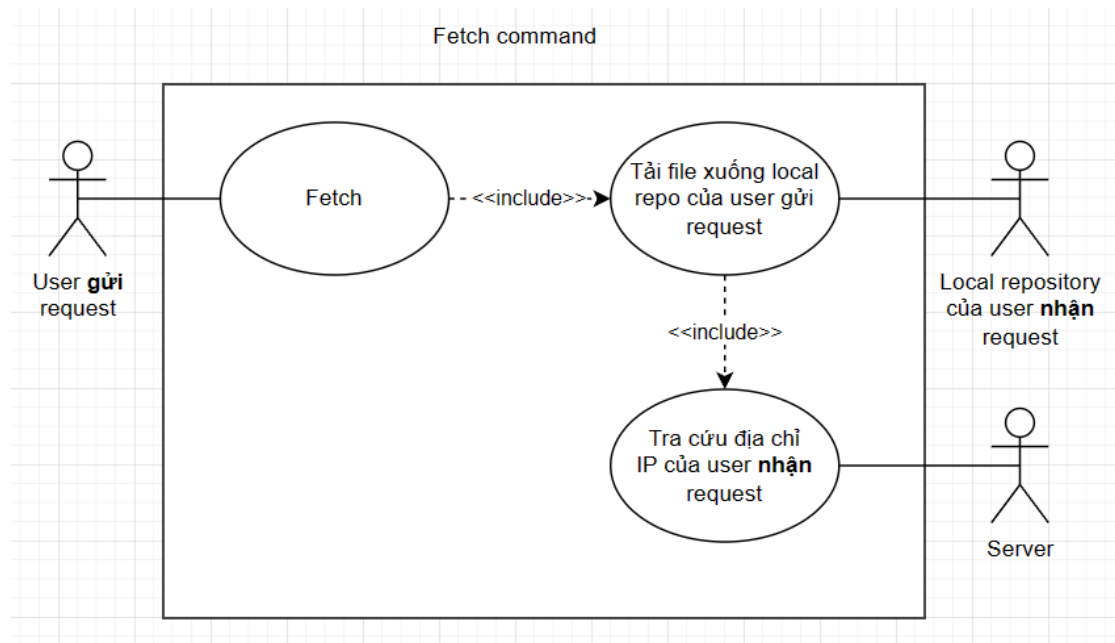


Hình 34: Activity digram: Tìm kiếm tài liệu bằng tên

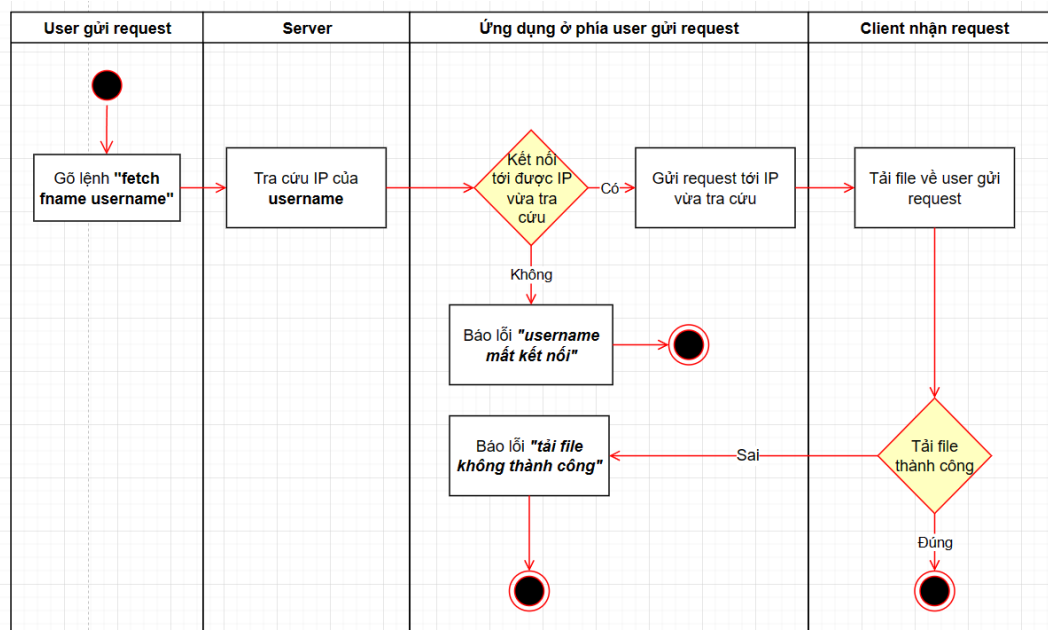
Đặc tả chi tiết:

- **Precondition:** Thông tin tài liệu client công bố tồn tại trên hệ thống.
- **Postcondition:** Danh sách username đang lưu trữ tài liệu.
- **Basic path:**
 1. Thông qua lệnh **filter fname** để tìm kiếm các tài liệu cùng tên vẫn còn khả dụng.
 2. Hệ thống hiển thị các tài liệu trùng tên với các thông tin cơ bản về tài liệu và **hostname** đã đăng tải.
- **Exceptional path:** Tại bước 2, nếu hệ thống không truy xuất được thông tin tài liệu thì sẽ không hiển thị dữ liệu.

4.3.4 Tải tài liệu từ cộng đồng



Hình 35: Use case: tải tài liệu từ cộng đồng



Hình 36: Activity diagram: tải tài liệu từ cộng đồng

Đặc tả chi tiết:

- **Precondition:** Hệ thống lưu trữ thông tin tài liệu client cần tải. Client công bố tài liệu và tài liệu còn khả dụng.
- **Postcondition:** Tài liệu được chia sẻ bởi các client
- **Basic path:**
 1. Thông qua lệnh `fetch fname hostname`, người dùng yêu cầu tải tài liệu còn khả dụng trên hệ thống với thông tin tương ứng.
 2. Hệ thống kiểm tra thông tin về tài liệu tồn tại và còn hữu dụng.
 3. Ứng dụng sẽ kiểm tra liệu có tài liệu `fname` trùng tên trên local repo và ghi đè.
 4. Hệ thống gửi địa chỉ IP của thiết bị đang lưu trữ tài liệu cho máy người dùng, hai thiết bị tự thiết lập kết nối và gửi tài liệu vào local repo của bên yêu cầu tải.
- **Exceptional path:** Tại bước số 2, nếu không tồn tại tài liệu với thông tin tương ứng thì yêu cầu sẽ bị hủy, hiển thị thông báo cho người dùng

5 KIỂM THỬ

5.1 Thiết lập hệ thống

Mục đích: Thực hiện và kiểm tra các bước tiên quyết khởi tạo hệ thống, kiểm tra việc client truy cập thành công vào hệ thống. Đồng thời, hệ thống kiểm tra các ràng buộc về: chỉ tồn tại một client đăng nhập cùng tài khoản tại một thời điểm, cập nhật metadata đối với mỗi local repo người dùng đăng nhập.

#	Trường hợp	Kỳ vọng	Kết quả
SU-01	Admin thiết lập server tập trung	- Server thiết lập thành công	- Server(Prisma) - Server(UI)
SU-02	Client tạo tài khoản mới	- Client: Tài khoản được tạo thành công - Server: Tạo mới tài khoản người dùng	- Client - Server
SU-03	Client đăng nhập vào hệ thống (khi local repo rỗng)	- Client: Truy cập được hệ thống. - Server: Ghi nhận thông tin client đã kết nối	- Client (MinIO) - Client(UI) - Server
SU-04	Client đăng nhập vào hệ thống (khi local repo có tài liệu)	- Client: Truy cập được hệ thống, tài liệu lưu trong local repo được công bố. - Server: Ghi nhận thông tin client đã kết nối và tài liệu khả dụng	- Client (MinIO) - Client(UI) - Server
SU-05	Client đăng nhập (trùng tài khoản đang hoạt động)	- Client đăng nhập không thành công	- Client

Bảng 16: Kiểm thử: Thiết lập hệ thống

5.2 Kiểm tra tính khả dụng của client

Mục đích: Kiểm tra tính đúng đắn của lệnh `ping hostname`.

#	Trường hợp	Kỳ vọng	Kết quả
PC-01	Admin sử dụng ping hostname (hostname khả dụng)	- Server cập nhật client và các tài liệu vẫn khả dụng	- Server
PC-02	Admin sử dụng ping hostname (hostname không khả dụng)	- Server cập nhật client và các tài liệu liên quan không còn khả dụng	- Server

Bảng 17: Kiểm thử: ping

5.3 Công bố/Hủy công bố tài liệu

#	Trường hợp	Kỳ vọng	Kết quả
PD-01	Client sử dụng publish fname	<ul style="list-style-type: none">- Client yêu cầu người dùng chọn file cần publish từ máy tính- Client sao chép tài liệu vào local repo với tên fname (có thể ghi đè nếu trùng)- Client gửi metadata tài liệu để cập nhật danh sách tài liệu đã công bố- Server cập nhật tính khả dụng và thông tin tài liệu	<ul style="list-style-type: none">- Client (người dùng chọn file)- Client (sao chép tài liệu vào local repo)- Server

Bảng 18: Kiểm thử: publish

5.4 Client yêu cầu tải dữ liệu

#	Trường hợp	Kỳ vọng	Kết quả
FD-01	Client sử dụng lệnh ls	- Server truy xuất và trả về thông tin tất cả tài liệu còn khả dụng	- Client hiển thị thông tin các tài liệu
FD-02	Client sử dụng lệnh filter fname	- Server truy xuất và trả về địa chỉ IP các client đang lưu trữ fname	- Client hiển thị các địa chỉ IP
FD-03	Client sử dụng lệnh fetch fname hostname (tồn tại tài liệu)	<ul style="list-style-type: none"> - Server truy xuất và trả về thông tin client đang lưu trữ tài liệu - Client thiết lập kết nối và nhận tài liệu từ client lưu trữ - Client thành công tải tài liệu vào local repo - Client gửi kết quả quá trình gửi tài liệu lên server - Server lưu lịch sử tải tài liệu, kết quả và metadata tài liệu vừa tải 	<ul style="list-style-type: none"> - Client (UI) - Client (MinIO) - Server
FD-04	Client sử dụng lệnh fetch fname hostname (không tồn tại tài liệu)	<ul style="list-style-type: none"> - Server truy xuất và trả về yêu cầu thất bại - Client tải tài liệu không thành công 	- Client
FD-05	Client sử dụng lệnh fetch fname hostname (hostname không chứa filename)	<ul style="list-style-type: none"> - Server truy xuất và trả về yêu cầu thất bại - Client tải tài liệu không thành công 	- Client
FD-06	Client sử dụng lệnh fetch fname hostname (hostname mất kết nối)	<ul style="list-style-type: none"> - Server truy xuất và trả về yêu cầu thất bại - Client tải tài liệu không thành công 	- Client

Bảng 19: Kiểm thử: ls và fetch

6 HƯỚNG DẪN SỬ DỤNG

6.1 Hướng dẫn cài đặt

Để sử dụng hệ thống, người dùng cần phải đáp ứng các yêu cầu kỹ thuật sau:

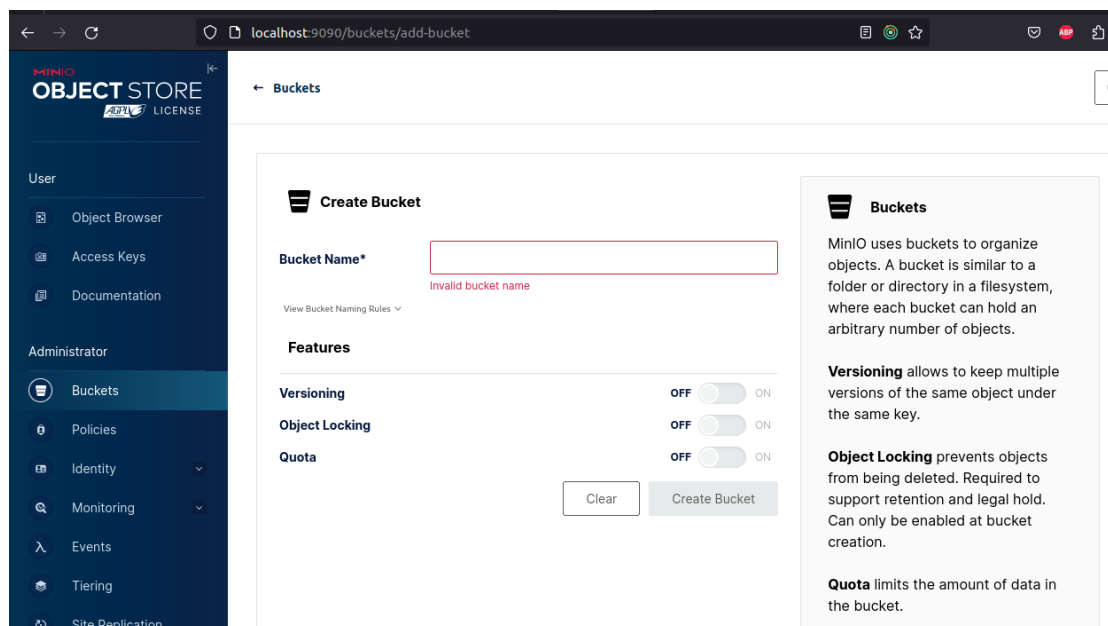
- **Thiết bị:** Tối thiểu 3 thiết bị trong hệ thống (hoặc 4 thiết bị để kiểm tra trường hợp xử lý đa luồng), với 1 thiết bị đóng vai trò máy chủ để các thiết bị còn lại là máy khách truy cập hệ thống. Người quản lý sẽ cần thiết lập 1 thiết bị làm máy chủ khi muốn khởi động hệ thống.
- **Kết nối:** Tất cả các thiết bị sử dụng chung một mạng LAN
- **Ứng dụng:** Các thiết bị phải tải thêm các ứng dụng được yêu cầu theo hướng dẫn.

6.1.1 Cài đặt các công cụ cần thiết

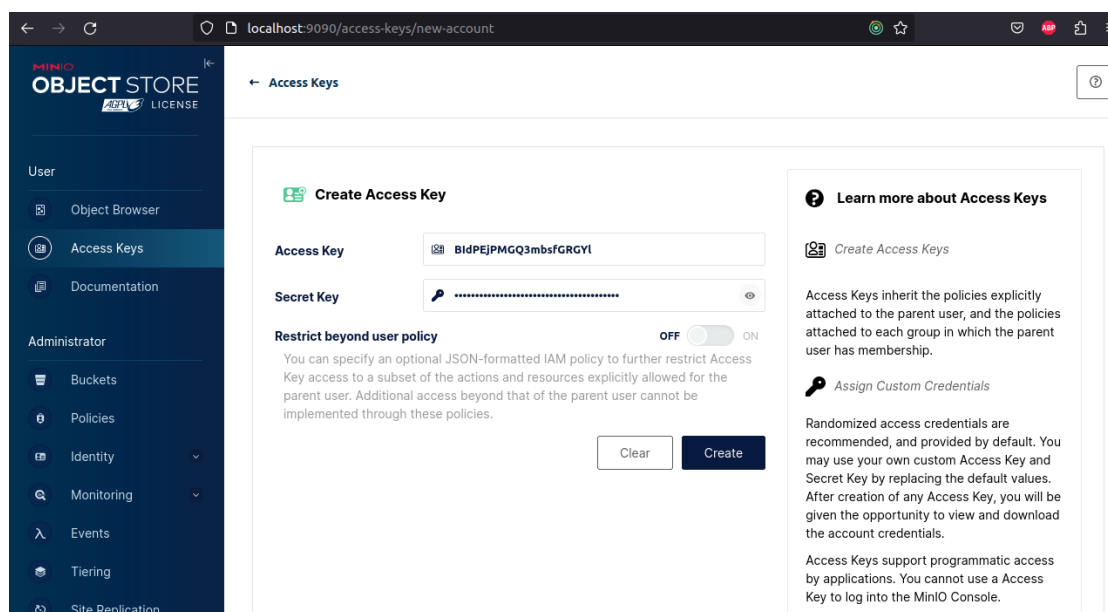
- **node** \geq v18.15.0
- **yarn** \geq v1.22.19
- **docker** \geq v23.0.1
- **docker compose** \geq v2.16.0

6.1.2 Cài đặt ứng dụng ở client

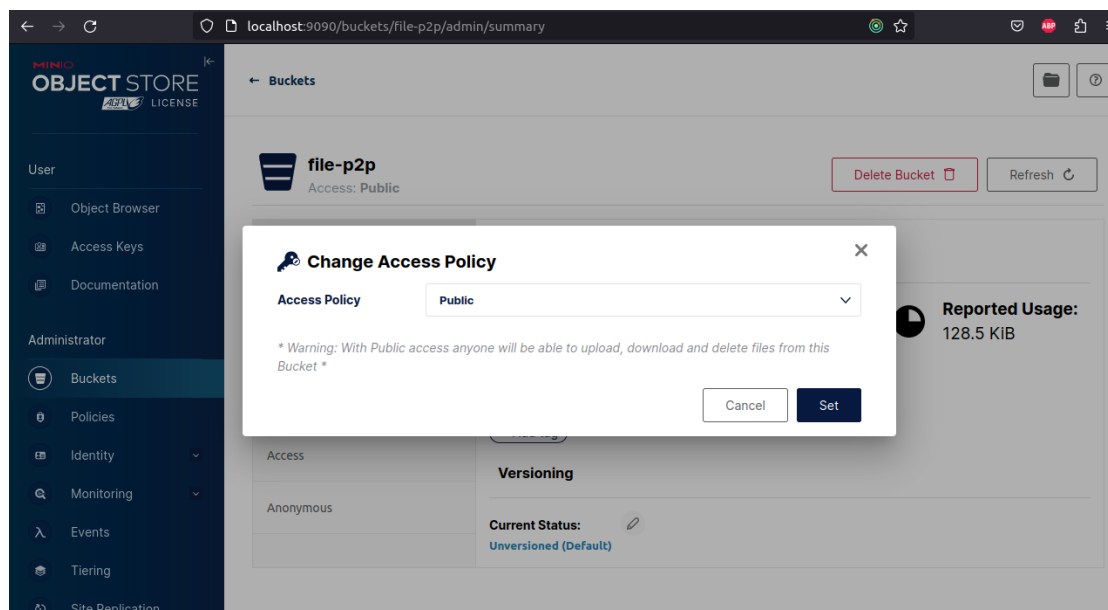
1. Trên hệ điều hành Unix/Linux, grant quyền **sudo** cho docker engine. Trên hệ điều hành Windows, mở Docker Desktop.
2. Từ thư mục "file-sharing-client" của source code, di chuyển đến thư mục "object-storage", cài đặt các dependencies cần thiết bằng lệnh: **yarn**.
3. Khởi tạo local repository bằng lệnh: **yarn start:docker**.
4. Vào localhost:9000 trên trình duyệt (Firefox), nhập Username: **admin** và Password: **123456789** để đăng nhập local repo.
5. Tạo bucket mới, access key, secret key và chỉnh quyền truy cập PUBLIC cho bucket như các hình dưới đây:



Hình 37: Tạo bucket mới



Hình 38: Tạo access key và secret key mới



Hình 39: Chỉnh quyền truy cập PUBLIC cho bucket

6. Sau đó nhập tên bucket, access key, secret key vào các biến môi trường trong file ".env":

- **MINIO_BUCKET_NAME**=<bucket-name>
- **MINIO_ACCESS_KEY**=<access-key>
- **MINIO_SECRET_KEY**=<secret-key>

7. Khởi chạy service bằng lệnh: **yarn start**.

8. Chuyển đường dẫn làm việc đến thư mục "ui", cài đặt các dependencies cần thiết bằng lệnh: **yarn**.

9. Tùy chỉnh địa chỉ IP của server ở biến môi trường **VITE_TRACKER_SERVER** (không thay đổi port).

10. Khởi chạy GUI + CLI bằng lệnh: **yarn start**.

11. Vào localhost:3000 để sử dụng ứng dụng.

6.1.3 Cài đặt ứng dụng ở server

1. Từ thư mục "file-sharing-server" của source code, di chuyển đến thư mục "server", cài đặt các dependencies cần thiết bằng lệnh: **yarn**.

2. Khởi tạo database và chạy dữ liệu mẫu bằng lệnh: **yarn bootstrap**.

3. Khởi chạy GUI bằng lệnh: **yarn db:studio**. Vào localhost:5555 để xem dữ liệu được lưu trữ trên server.
4. Khởi chạy service bằng lệnh: **yarn start**.
5. Đi đến thư mục "ui", cài đặt các dependencies cần thiết bằng lệnh: **yarn**.
6. Tùy chỉnh địa chỉ IP của server ở biến môi trường **VITE_TRACKER_SERVER** (không thay đổi port).
7. Khởi chạy CLI bằng lệnh: **yarn start**.
8. Vào localhost:3030 để sử dụng máy chủ.

6.2 Hướng dẫn thao tác trên hệ thống

6.2.1 Đối với người quản lý hệ thống

6.2.1.a Đăng nhập hệ thống

Sau khi thiết lập một máy chủ tập trung, người quản lý có thể thao tác trực tiếp trên máy chủ để truy cập hệ thống mà không cần đăng nhập. Cụ thể, người quản lý được thao tác trên hai giao diện:

- Giao diện chính (GUI mô phỏng hành vi CLI) tại <http://localhost:3030/>: Thực hiện các thao tác lệnh ứng dụng
- Giao diện hỗ trợ (GUI có sẵn của Prisma) tại <http://localhost:5555/>: Là giao diện của admin để quản lý cơ sở dữ liệu tập trung

6.2.1.b Truy xuất thông tin client (*mở rộng*)

1. Admin truy cập giao diện chính (<http://localhost:3030/>) và sử dụng lệnh: **ls clients active**.
2. CLI sẽ hiển thị danh sách username của các client đang hoạt động trong hệ thống.

```
>>> ls clients active
nguyenthitam
conbaocon
congacha
congame
meomeomeo
phunguyen
```

Hình 40: Giao diện: Truy xuất thông tin client

6.2.1.c Kiểm tra tính khả dụng của client

1. Admin truy cập giao diện chính (<http://localhost:3030/>) và sử dụng lệnh: **ping** hostname, với hostname là username truy xuất được từ tính năng **Truy xuất thông tin client (mở rộng)** để truy xuất thông tin các client còn hoạt động trong hệ thống.
2. CLI sẽ hiển thị kết quả kiểm tra client còn hoạt động trong hệ thống với: Địa chỉ IP của client và kết quả (Alive: còn hoạt động, Dead: không còn hoạt động do timeout sau 5s).

```
>>> ping meomeomeo
(192.168.1.236) Alive !
>>> ping congacon
HostName not found !
>>> ping nguyenthitam
>>> ping nguyenthitam
(192.168.1.236) Alive !
>>> ping nguyenthitam
(192.168.1.236) Dead !
```

Hình 41: Giao diện: Kiểm tra tính khả dụng client

6.2.1.d Truy xuất tài liệu chia sẻ

1. Admin truy cập giao diện chính (<http://localhost:3030/>) và sử dụng lệnh: **discover** hostname để truy xuất thông tin các tài liệu còn khả dụng được chia sẻ bởi hostname.

2. CLI sẽ hiển thị danh sách các tài liệu còn khả dụng của client đó với thông tin về: tên, kích thước, loại, thời điểm chia sẻ

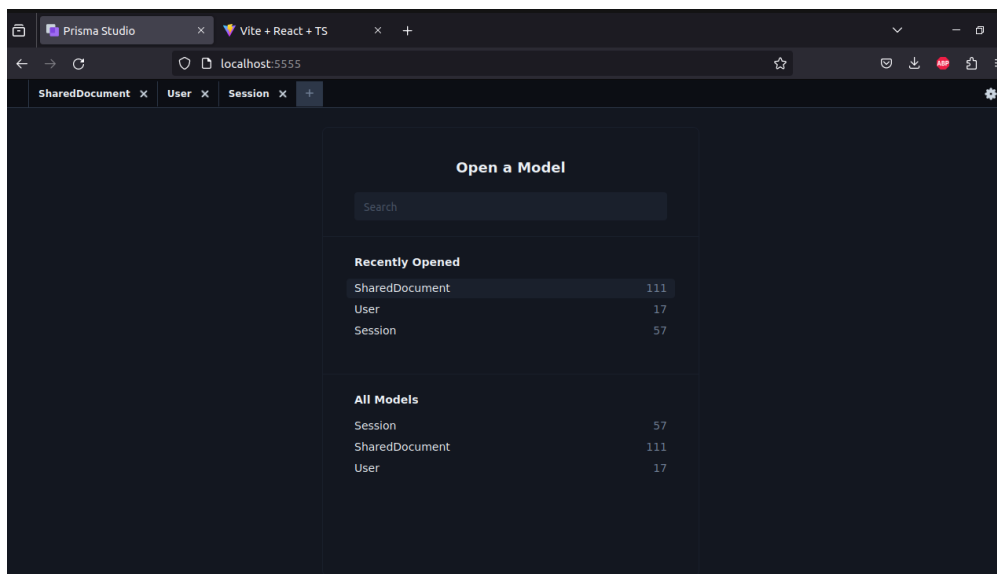
```
>>> discover congacha
```

Name	Type	Size	Shared Time	Is Available
abc.pdf	pdf	61.73 KB	16:03, 07/11/2023	true
bt11.pdf	pdf	493.21 KB	16:03, 07/11/2023	true
def.pdf	pdf	280.20 KB	16:03, 07/11/2023	true
f1GB.txt	txt	1024.00 MB	16:03, 07/11/2023	true
filenho.pdf	pdf	493.21 KB	16:03, 07/11/2023	true
hello.pdf	pdf	1.85 MB	16:03, 07/11/2023	true

Hình 42: Giao diện: Truy xuất tài liệu chia sẻ

6.2.1.e Quản lý cơ sở dữ liệu tập trung (*mở rộng*)

1. Admin truy cập giao diện hỗ trợ (<http://localhost:5555/>) để truy cập cơ sở dữ liệu tập trung, với các bảng dữ liệu: User, Session, SharedDocument. Admin chọn bảng dữ liệu muốn theo dõi.



Hình 43: Giao diện: Các bảng dữ liệu

2. Sau khi truy cập vào một bảng dữ liệu, các trường dữ liệu sẽ tương tự như đã được đề cập tại phần [Quản lý cơ sở dữ liệu tập trung \(*mở rộng*\)](#)

Prisma Studio

Vite + React + TS

localhost:5555

SharedDocument

User

Session

Filters

None

Fields

All

Showing

100 of 117

Add record


id	A	name	A	type	A	size	#	sharedTime	#	isAvailable		sessi
cloirktw20083oxdq57g...	2222	.pdf	pdf	588021	1699025927	true		Sessi				
cloiuaosk0085ox328kyt...	2353	.pdf	pdf	171618	1699030841	true		Sessi				
clojmj3dt0084oxikbxvr...	1248	.pdf	pdf	305	1699085342	true		Sessi				
clojml0g20080oxik4oea...	1248	.pdf	pdf	131555	1699084547	true		Sessi				
clojs3l6s0083oxfj5wha...	1248	.pdf	pdf	131555	1699087089	true		Sessi				
clojtr3xo0083ox50qwt2...		image.jpg	jpg	61632	1699091558	true		Sessi				
clojyd5lc0081oxhhlqu1...		db.pdf	pdf	4503353	1699096800	true		Sessi				
clok07lys0083oxhpe5k...	1911	.pdf	pdf	131555	1699099896	true		Sessi				
clolb60950080ox7lgatz...	1248	.pdf	pdf	131555	1699178798	false		Sessi				
clolb60960801ox7i0tp6...	1911	.pdf	pdf	131555	1699178798	false		Sessi				
clolb60960802ox7ihjbd...		db.pdf	pdf	4503353	1699179436	false		Sessi				
clolb60960803ox7i4oiw...		image.jpg	jpg	61632	1699192977	false		Sessi				
clolbipi2008qox7i7evh...	1248	.pdf	pdf	131555	1699179360	true		Sessi				
clolbipi2008rox7ixn7z...	1911	.pdf	pdf	131555	1699179360	true		Sessi				
clolbipi2008sox7iar8w...		db.pdf	pdf	4503353	1699179360	true		Sessi				
clolbipi2008tox7im9kg...		image.jpg	jpg	61632	1699179360	true		Sessi				

Hình 44: Giao diện: Các trường dữ liệu của bảng

6.2.2 Đối với người dùng thông thường

6.2.2.a Tạo tài khoản/Đăng nhập

- Client truy cập vào giao diện chính (<http://localhost:3000>), thao tác trên GUI và tạo mới tài khoản hoặc đăng nhập với thông tin về: họ và tên, tên tài khoản và mật khẩu.



Sign Up

Nice to meet you! Enter your details to register.

Full Name

Username

Password

SIGN UP

Already have an account? [Sign In](#)

Hình 45: Giao diện: Tạo tài khoản

- Sau đó, người dùng có thể sử dụng thông tin đã tạo để đăng nhập vào hệ

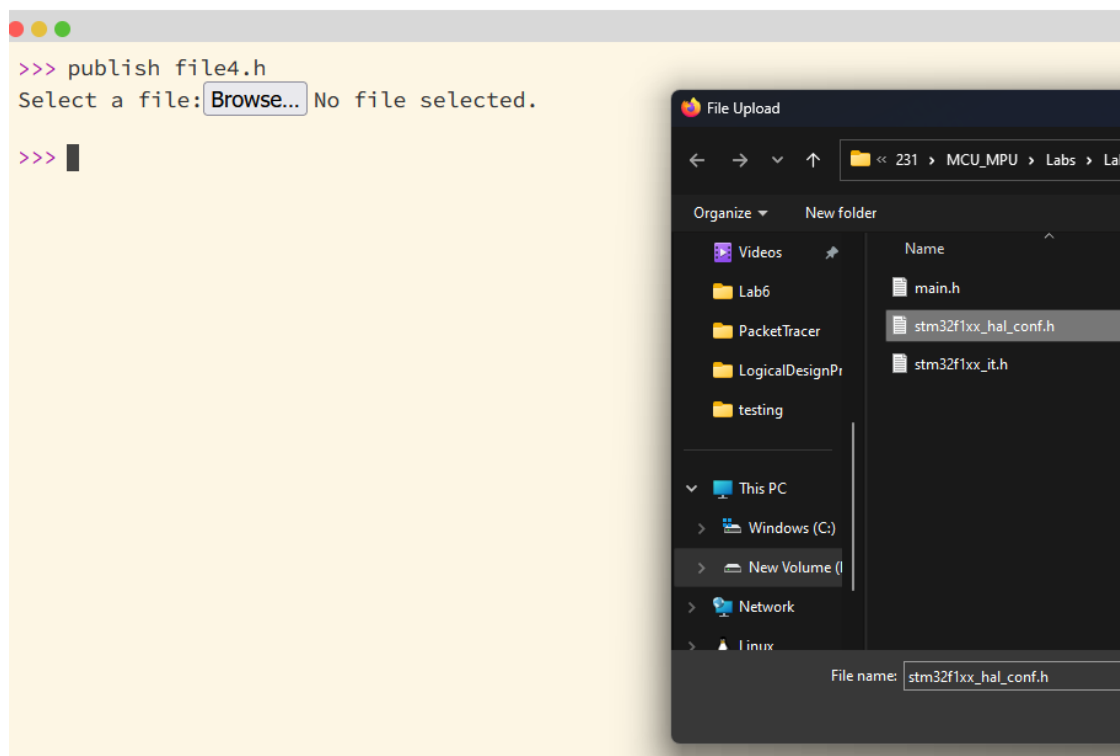
thống. Mỗi tài khoản chỉ có thể đăng nhập trên một thiết bị tại một thời điểm.



Hình 46: Giao diện: Đăng nhập tài khoản

6.2.2.b Công bố/Hủy công bố tài liệu

1. Client truy cập vào giao diện chính (<http://localhost:3000>), sử dụng thông qua các lệnh để công bố/hủy công bố tài liệu.
2. Hệ thống sẽ ghi nhận thông tin dựa trên lệnh ứng dụng được nhập:
 - Công bố tài liệu (**publish fname**): Công bố tài liệu cho phép các client khác tải về, đồng thời lưu vào local repo của client. Người dùng sẽ nhập tên file (bao gồm cả đuôi file) muốn lưu **fname** và lựa chọn tài liệu tải lên từ máy cá nhân.

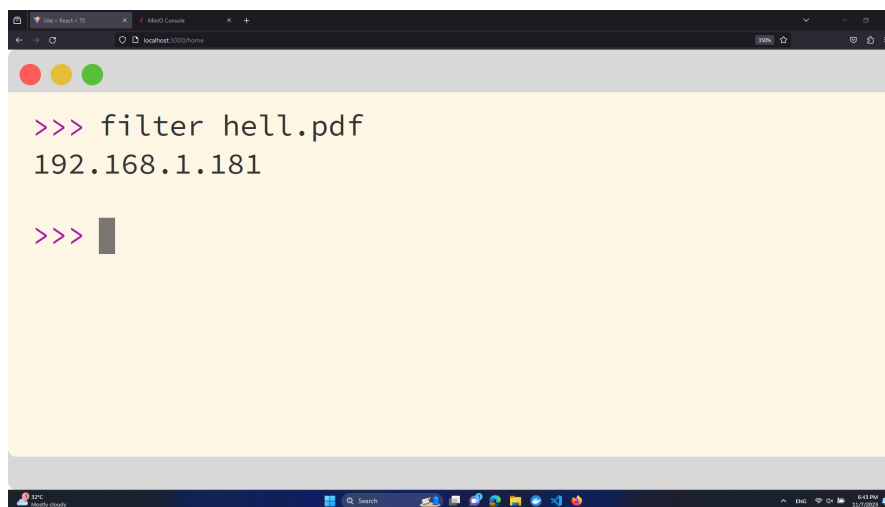


Hình 47: Giao diện: Công bố tài liệu

- Hủy công bố tài liệu (**unpublish** **fname**): Hủy công bố tài liệu tên **fname** không cho phép các client khác tải về, đồng thời xóa khỏi local repo của client.

6.2.2.c Tìm kiếm client công bố tài liệu

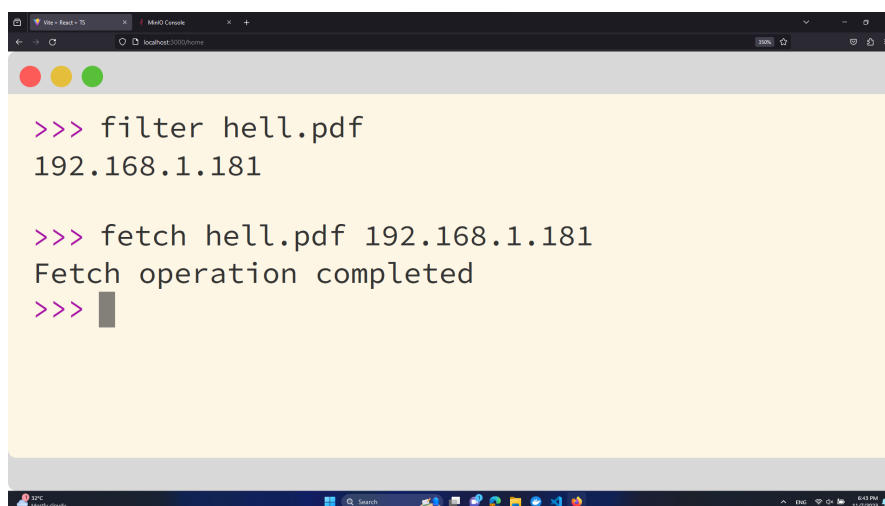
1. Client truy cập vào giao diện chính (<http://localhost:3000>)
2. Người dùng gõ lệnh **filter** **filename**, hệ thống sẽ hiển thị địa chỉ IP của các client đã công bố tài liệu **filename**.



Hình 48: Giao diện: Tìm kiếm client công bố tài liệu

6.2.2.d Tải tài liệu từ cộng đồng

1. Client truy cập vào giao diện chính (<http://localhost:3000>)
2. Người dùng gõ lệnh **fetch** filename hostname, tài liệu filename (nếu có) sẽ được tải từ hostname về máy tính của người dùng.

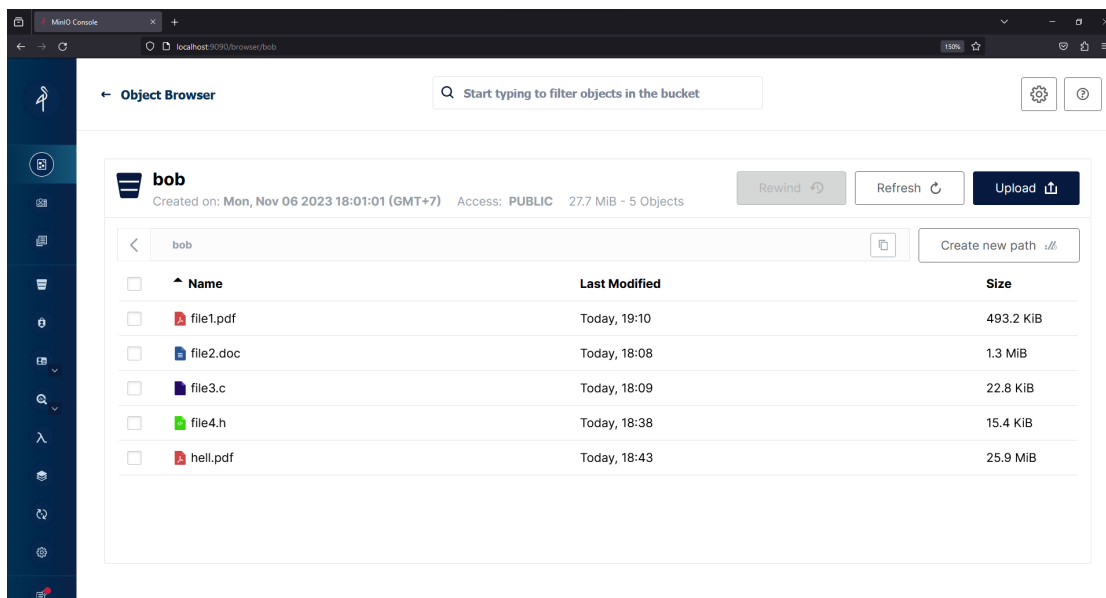


Hình 49: Giao diện: Tải tài liệu từ cộng đồng

6.2.2.e Quản lý kho tài liệu cá nhân (*mở rộng*)

1. Client truy cập vào giao diện chính (<http://localhost:9090>)

2. Người dùng có thể quan sát được những tài liệu đã **publish** hoặc **fetch** thông qua giao diện được cung cấp bởi MinIO.



Hình 50: Giao diện: Quản lý kho tài liệu cá nhân

7 TÀI LIỆU THAM KHẢO

1. Abdullahi Yari, Imrana & Dehling, Tobias & Kluge, Felix & Geck, Juergen & Sunyaev, Ali & Eskofier, Bjoern. (2021). *Security Engineering of Patient-Centered Health Care Information Systems in Peer-to-Peer Environments: Systematic Review*.
2. Ding, C. H., Nutanong, S., & Buyya, R. (2005). Peer-to-Peer Networks for Content Sharing. In R. Subramanian & B. Goodman (Eds.), *Peer-to-Peer Computing: The Evolution of a Disruptive Technology*. IGI Global. <https://doi.org/10.4018/978-1-59140-429-3.ch002>
3. Do Van Nam (2018). *Minio – Object storage server như AWS S3*. <https://viblo.asia/p/minio-object-storage-server-nhu-aws-s3-LzD5d0AW5jY>
4. Kurose, J.F. & Ross, K.W. *Computer Networks, A Top-down Approach*. (2020). 8th ed. https://www.researchgate.net/figure/The-centralized-peer-to-peer-P2P-system-A-peer-E-sends-a-message-to-the-central-server_fig3_356245976