HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

----ꕥ📖ꕥ----

BÀI BÁO CÁO MÔN CÁC HỆ THỐNG PHÂN TÁN

ĐỀ TÀI:
HỆ THỐNG QUẢN LÝ PHÒNG KHÁM

Giảng viên : **TS. KIM NGỌC BÁCH**

Học viên thực hiện : **NGUYỄN ĐÌNH HÀ - B24CHHT067**

NGUYỄN HỒNG PHONG - B24CHHT088

TRỊNH QUANG TÙNG - B24CHHT097

Lớp : **M24CQHT02-B**

MỤC LỤC

CHƯƠNG I. MỞ ĐẦU

1. Bối cảnh và lý do nghiên cứu

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, các hệ thống quản lý y tế truyền thống dần bộc lộ nhiều hạn chế như thiếu tính linh hoạt, khó mở rộng và khả năng truy cập dữ liệu còn hạn chế. Sự bùng nổ của công nghệ điện toán đám mây và kiến trúc hệ thống phân tán đã mở ra cơ hội mới cho việc phát triển các phần mềm quản lý phòng khám hiện đại, cho phép lưu trữ dữ liệu trên nhiều máy chủ, đảm bảo tính sẵn sàng và khả năng mở rộng.

Việc xây dựng một hệ thống quản lý phòng khám theo mô hình phân tán không chỉ đáp ứng nhu cầu truy cập nhanh chóng, an toàn mà còn hỗ trợ tích hợp các dịch vụ nâng cao như lưu trữ đám mây, xác thực bảo mật, và phân quyền người dùng. Đây chính là lý do nghiên cứu và phát triển một giải pháp phần mềm dựa trên kiến trúc phân tán cho quản lý phòng khám.

2. Vấn đề và khoảng trống nghiên cứu

Các hệ thống quản lý phòng khám hiện nay chủ yếu triển khai theo mô hình tập trung, gây ra một số hạn chế:

- Khó mở rộng khi số lượng người dùng và dữ liệu tăng nhanh.

- Rủi ro mất dữ liệu nếu máy chủ trung tâm gặp sự cố.

- Khó tích hợp dịch vụ đám mây để lưu trữ file, ảnh y tế.

Mặc dù đã có một số hệ thống y tế ứng dụng mô hình phân tán, nhưng phần lớn giải pháp này tập trung cho bệnh viện quy mô lớn, chưa tối ưu cho các phòng khám vừa và nhỏ với yêu cầu triển khai nhanh, chi phí hợp lý. Khoảng trống nghiên cứu là cần một kiến trúc phân tán đơn giản, hiệu quả, dễ triển khai cho các phòng khám thông thường.

3. Mục tiêu nghiên cứu

- Xây dựng mô hình hệ thống quản lý phòng khám dựa trên kiến trúc hệ thống phân tán.

- Ứng dụng các công nghệ hiện đại như MongoDB, Cloudinary, Node.js, React để triển khai phần mềm.

- Đảm bảo các yêu cầu về bảo mật, phân quyền, hiệu năng và khả năng mở rộng.

- Đề xuất giải pháp bảo mật và tối ưu hiệu năng trong môi trường phân tán.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

Kiến trúc phân tán áp dụng trong hệ thống quản lý dữ liệu y tế.

Các kỹ thuật bảo mật và tối ưu hiệu năng trong hệ thống phân tán.

Phạm vi nghiên cứu: Xây dựng phần mềm quản lý phòng khám với các chức năng cơ bản:

- Quản lý người dùng (Admin, Doctor, Patient).
- Đăng nhập và phân quyền qua JWT middleware.
- Quản lý bệnh nhân, bác sĩ và hồ sơ khám bệnh.
- Lưu trữ dữ liệu trên MongoDB và file trên Cloudinary.

5. Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học:

Góp phần minh chứng tính ứng dụng của kiến trúc phân tán trong quản lý y tế quy mô nhỏ.

Đưa ra mô hình kiến trúc hệ thống phân tán kết hợp lưu trữ đám mây cho bài toán thực tế.

Ý nghĩa thực tiễn:

Giúp các phòng khám tối ưu quy trình quản lý hồ sơ bệnh nhân và bác sĩ.

Nâng cao tính bảo mật và khả năng truy cập dữ liệu ở mọi nơi, mọi thời điểm.

Giảm chi phí triển khai nhờ tận dụng dịch vụ đám mây và kiến trúc mở rộng linh hoạt.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN

1. Khái niệm hệ thống phân tán

Hệ thống phân tán (Distributed System) là một tập hợp các thành phần tính toán độc lập, được kết nối với nhau thông qua mạng và phối hợp hoạt động như một hệ thống duy nhất. Người dùng hoặc ứng dụng khi tương tác với hệ thống phân tán sẽ có cảm giác đang làm việc trên một hệ thống thống nhất, mặc dù về thực chất dữ liệu và xử lý được phân tán trên nhiều nút (node) khác nhau.

Các hệ thống này không bị ràng buộc bởi vị trí địa lý, có thể bao gồm các máy chủ, dịch vụ và thiết bị ở nhiều khu vực khác nhau, kết nối thông qua mạng Internet hoặc mạng chuyên dụng. Hệ thống phân tán có mặt trong nhiều lĩnh vực như thương mai điện tử, mạng xã hội, xử lý dữ liệu lớn, dịch vụ tài chính và hệ thống IoT.

2. Đặc trưng của hệ thống phân tán

Hệ thống phân tán sở hữu những đặc trưng nổi bật sau:

(1) Tính trong suốt (Transparency)

Trong suốt về vị trí: Người dùng không biết dữ liệu nằm ở máy chủ nào.

Trong suốt về truy cập: Cách truy cập tài nguyên giống như trên hệ thống cục bộ.

Trong suốt về nhân bản (Replication): Dữ liệu có thể được sao chép mà người dùng không nhận ra.

(2) Tính mở rộng (Scalability)

Hệ thống có thể dễ dàng mở rộng bằng cách thêm máy chủ hoặc dịch vụ mà không ảnh hưởng đến kiến trúc tổng thể.

(3) Khả năng chịu lỗi (Fault tolerance)

Hệ thống tiếp tục hoạt động ngay cả khi một số thành phần bị lỗi, thông qua cơ chế dự phòng (redundancy) hoặc tự động chuyển đổi (failover).

(4) Tính đồng bộ và nhất quán dữ liệu

Các nút trong hệ thống phải đồng bộ dữ liệu theo cơ chế strong consistency hoặc eventual consistency để đảm bảo tính chính xác.

(5) Đa dạng nền tảng (Heterogeneity)

Các thành phần có thể chạy trên nhiều nền tảng phần cứng và hệ điều hành khác nhau.

(6) Tính bảo mật

Do dữ liệu được phân tán trên nhiều nút và truyền qua mạng, hệ thống phân tán cần cơ chế xác thực (authentication), mã hóa (encryption) và quản lý phân quyền (authorization) chặt chẽ.

3. So sánh hệ thống tập trung và phân tán

Tiêu chí	**Hệ thống tập trung**	**Hệ thống phân tán**
Kiến trúc	Một máy chủ chính	Nhiều node liên kết
Khả năng mở rộng	Hạn chế	Cao (thêm node dễ dàng)
Tính sẵn sàng	Thấp (Single Point of Failure)	Cao nhờ nhân bản và dự phòng
Chi phí ban đầu	Thấp hơn	Cao hơn
Quản lý	Đơn giản	Phức tạp hơn

4. Mô hình kiến trúc hệ thống phân tán

Client–Server: Phổ biến, client gửi yêu cầu, server xử lý và phản hồi.

Peer-to-Peer (P2P): Các node bình đẳng, vừa đóng vai trò client vừa làm server.

Microservices: Chia hệ thống thành các dịch vụ nhỏ, độc lập, giao tiếp qua API hoặc message broker.

Event-driven Architecture: Các thành phần phản ứng theo sự kiện, thích hợp cho xử lý bất đồng bộ.

5. Giao tiếp trong hệ thống phân tán

Giao thức đồng bộ: RESTful API qua HTTP/HTTPS, gRPC.

Giao thức bất đồng bộ: Message Queue (RabbitMQ, Kafka), Pub/Sub (Google Pub/Sub, AWS SNS).

Quản lý dịch vụ: Service Discovery (Eureka, Consul), API Gateway (Kong, NGINX, AWS API Gateway).

6. Bảo mật trong hệ thống phân tán

Các thách thức bảo mật bao gồm xác thực, phân quyền, mã hóa dữ liệu, chống tấn công DDoS và bảo vệ API. Một số giải pháp:

Xác thực & phân quyền: OAuth2, JWT, Multi-Factor Authentication (MFA).

Mã hóa: TLS/SSL cho truyền dữ liệu, AES-256 cho lưu trữ.

Tường lửa ứng dụng web (WAF): Chặn tấn công SQL Injection, XSS.

7. Ứng dụng của hệ thống phân tán trong quản lý phòng khám

Hệ thống phân tán được ứng dụng rộng rãi trong nhiều lĩnh vực, trong đó có y tế và quản lý phòng khám, nhằm giải quyết các vấn đề về lưu trữ, xử lý dữ liệu và tính khả dụng của hệ thống. Một số ứng dụng cụ thể trong phần mềm quản lý phòng khám bao gồm:

Quản lý hồ sơ bệnh án điện tử (EMR): Các hồ sơ bệnh án của bệnh nhân được lưu trữ trên cơ sở dữ liệu phân tán (như MongoDB) để đảm bảo tính toàn vẹn và truy cập nhanh chóng. Dữ liệu có thể được sao chép trên nhiều node để tránh mất mát khi xảy ra sự cố.

Lưu trữ và xử lý file y tế trên dịch vụ đám mây: Các file như kết quả xét nghiệm, hình ảnh X-quang, đơn thuốc được lưu trữ trên nền tảng lưu trữ đám mây (ví dụ: Cloudinary). Điều này giúp giảm tải cho máy chủ chính và cho phép người dùng truy cập file mọi lúc, mọi nơi thông qua liên kết bảo mật.

Phân quyền và xác thực người dùng: Hệ thống phân tán cho phép triển khai nhiều dịch vụ xác thực và phân quyền khác nhau để đảm bảo bảo mật và tránh truy cập trái phép. Ví dụ, sử dụng JWT middleware giúp bảo vệ API ở lớp backend trong môi trường phân tán.

Khả năng mở rộng quy mô: Khi số lượng người dùng (bác sĩ, bệnh nhân) tăng lên, hệ thống phân tán có thể mở rộng bằng cách thêm nhiều máy chủ ứng dụng hoặc node cơ sở dữ liệu mà không ảnh hưởng đến hoạt động chung của hệ thống.

Truy cập từ xa và tính sẵn sàng cao: Nhờ kiến trúc phân tán, dữ liệu được lưu trữ trên nhiều máy chủ hoặc dịch vụ đám mây, đảm bảo hệ thống hoạt động liên tục ngay cả khi một máy chủ gặp sự cố. Điều này đặc biệt quan trọng đối với các phòng khám phục vụ nhiều chi nhánh hoặc cần truy cập từ xa.

CHƯƠNG 3. KIẾN TRÚC PHÂN TÁN

1. Định hướng kiến trúc

Kiến trúc của hệ thống quản lý phòng khám được thiết kế theo hướng Client-Server kết hợp dịch vụ đám mây, đảm bảo:

- Phân tán về lưu trữ:

  - Dữ liệu văn bản (người dùng, hồ sơ khám) lưu trữ trong MongoDB.
  - File đính kèm (ảnh, PDF, kết quả xét nghiệm) lưu trữ trên Cloudinary.

Phân tán về triển khai:

- Frontend (React + Vite) triển khai độc lập, giao tiếp với backend thông qua REST API.
- Backend (Node.js + Express) cung cấp API cho tất cả các chức năng.
- Có thể triển khai backend trên nhiều node để đảm bảo khả năng mở rộng.

Định hướng bảo mật:

- Sử dụng JWT middleware để xác thực, bảo vệ API.
- Áp dụng HTTPS cho toàn bộ giao tiếp giữa client và server.

Khả năng mở rộng và tương tác:

- Cho phép tích hợp thêm các dịch vụ nâng cao như: đặt lịch khám trực tuyến, thanh toán điện tử, tư vấn trực tuyến (video call).

Tóm lại, định hướng kiến trúc tập trung vào phân tán lưu trữ và xử lý, đảm bảo tính linh hoạt, mở rộng và bảo mật cho hệ thống.

2. Thành phần kiến trúc

Hệ thống được cấu trúc thành 4 thành phần chính:

(1) Lớp Giao diện (Frontend)

Xây dựng bằng React + Vite để tối ưu tốc độ tải và trải nghiệm người dùng.

Quản lý trạng thái đăng nhập bằng Context API, điều hướng bằng React Router.

Giao tiếp với backend thông qua Axios qua các API bảo mật.

Hiển thị các chức năng chính:

Đăng nhập/đăng xuất.

Quản lý thông tin bệnh nhân, bác sĩ.

Quản lý hồ sơ khám bệnh.

(2) Lớp Dịch vụ (Backend)

Viết bằng Node.js + Express.

Chịu trách nhiệm:

Xử lý logic nghiệp vụ (business logic).

Kiểm tra phân quyền dựa trên role (admin, doctor, patient).

Quản lý xác thực thông qua JWT middleware.

Cung cấp các API:

- /users: quản lý người dùng
- /patients: quản lý bệnh nhân
- /doctors: quản lý bác sỹ
- /medical-records: quản lý hồ sơ khám bệnh
- /login: đăng nhập và cấp token
- /reset-password: khởi tạo lại mật khẩu

(3) Lớp Cơ sở dữ liệu (MongoDB)

Lưu trữ:

- Thông tin người dùng: tên, username, mật khẩu (mã hóa bằng bcrypt), role.
- Hồ sơ khám bệnh: triệu chứng, chẩn đoán, đơn thuốc, ghi chú, ngày khám.

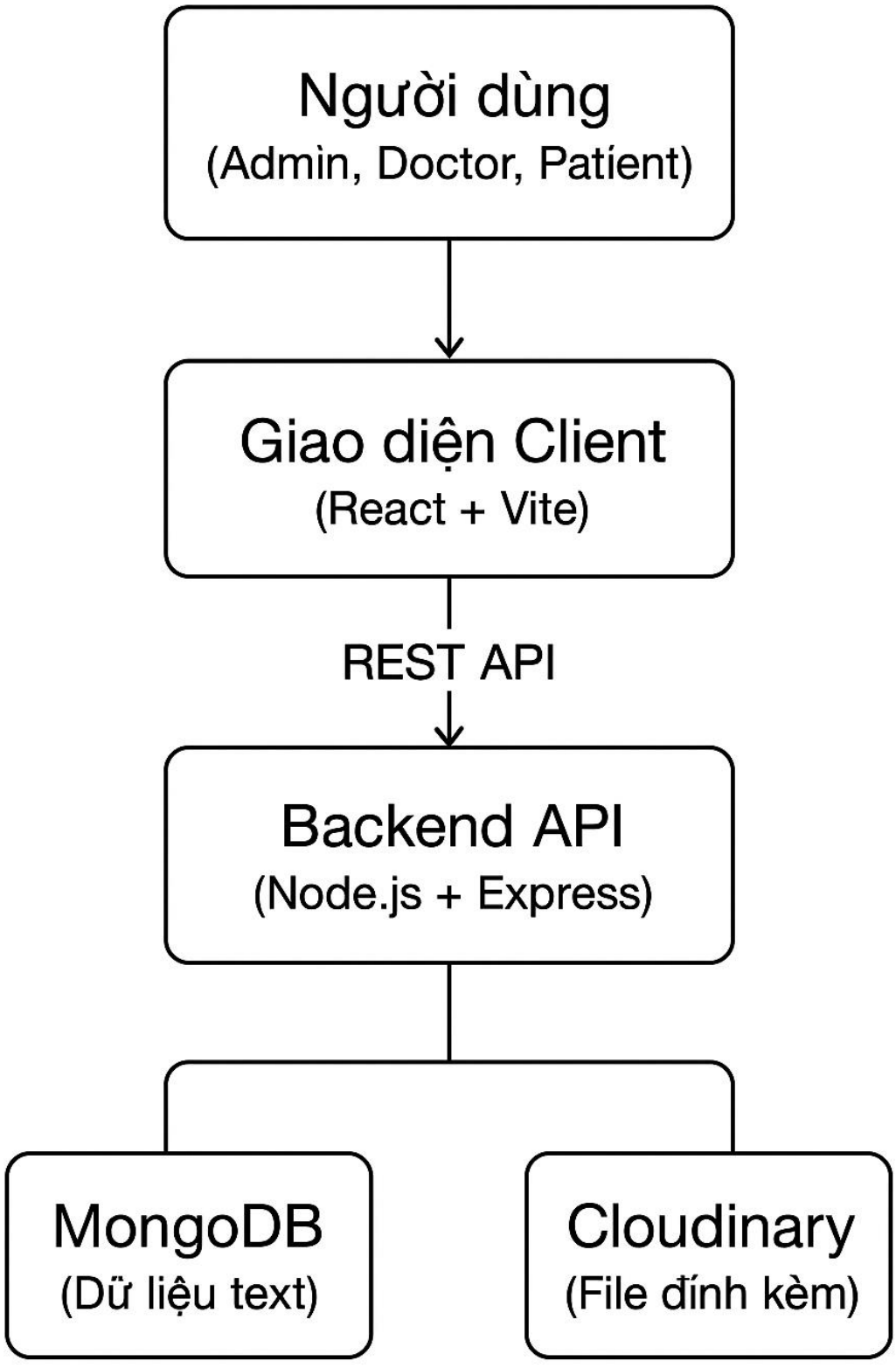
Hỗ trợ Indexing cho tìm kiếm nhanh theo tên, ngày khám.

(4) Lớp Lưu trữ đám mây (Cloudinary)

Lưu trữ các file đính kèm: hình ảnh xét nghiệm, đơn thuốc dạng ảnh hoặc PDF.

Trả về URL bảo mật để hiển thị trên giao diện hoặc tải xuống.

3. Sơ đồ kiến trúc tổng thể


Người dùng
(Admin, Doctor, Patient)
Giao diện Client
(React + Vite)
REST API
Backend API
(Node.js + Express)
MongoDB
(Dữ liệu text)
Cloudinary
(File đính kèm)

4. Cơ chế hoạt động

Hệ thống quản lý phòng khám hoạt động dựa trên nguyên tắc Client – Server và tích hợp dịch vụ đám mây. Cơ chế chính gồm các bước:

(1) Xác thực và phân quyền

Người dùng gửi yêu cầu đăng nhập kèm thông tin tài khoản đến API /auth/login.

Backend xác thực thông tin, nếu hợp lệ sẽ trả về JWT token.

Client lưu token (thường trong localStorage hoặc sessionStorage) và gửi token trong header cho mỗi yêu cầu API.

Middleware JWT tại backend kiểm tra token và phân quyền:

- Admin: toàn quyền quản lý (users, records).
- Doctor: quản lý bệnh nhân và hồ sơ mình phụ trách.
- Patient: chỉ xem thông tin cá nhân và lịch sử khám.

(2) Quản lý bệnh nhân và bác sĩ

Khi Admin hoặc Doctor thêm/sửa bệnh nhân:

Client gửi dữ liệu qua API /users (POST hoặc PATCH).

Backend lưu thông tin vào MongoDB.

Khi tìm kiếm bệnh nhân:

Client gọi API /patients?search=....

Backend trả về danh sách phù hợp từ MongoDB.

(3) Quản lý hồ sơ khám bệnh

Khi tạo mới hồ sơ khám:

Người dùng (Admin/Doctor) nhập dữ liệu, upload file (nếu có).

File được gửi tới Cloudinary và trả về URL.

Backend lưu URL cùng dữ liệu hồ sơ vào MongoDB.

Khi xem danh sách hồ sơ:

Client gửi yêu cầu đến API /medical-records.

Backend trả về dữ liệu đã lọc (theo tên, ngày khám, role).

(4) Lưu trữ file đính kèm

File tải lên được gửi từ frontend đến backend.

Backend gọi API của Cloudinary để lưu file.

Cloudinary trả về secure URL và lưu vào MongoDB.

Khi hiển thị, frontend sử dụng URL này để hiển thị ảnh hoặc link tải.

(5) Đồng bộ và phân tán

Các yêu cầu được xử lý độc lập trên backend, có thể triển khai trên nhiều node.

MongoDB hỗ trợ replica set để đảm bảo dữ liệu không mất khi một node lỗi.

Cloudinary hoạt động phân tán toàn cầu, giúp truy xuất file nhanh chóng.

5. Ưu điểm của kiến trúc

Khả năng mở rộng cao:

- Có thể dễ dàng mở rộng số lượng server backend khi số lượng người dùng tăng.
- MongoDB và Cloudinary hỗ trợ scale ngang, đáp ứng lượng dữ liệu lớn.

Hiệu năng tốt:

- Frontend và Backend tách biệt → dễ tối ưu và triển khai CDN cho frontend.
- Cloudinary giảm tải cho backend nhờ xử lý và lưu trữ file bên ngoài.

Bảo mật được tăng cường:

- JWT giúp phân quyền rõ ràng, ngăn chặn truy cập trái phép.
- Giao tiếp qua HTTPS bảo vệ dữ liệu khỏi bị đánh cắp.

Khả năng truy cập linh hoạt:

- Người dùng (bác sĩ, bệnh nhân) có thể truy cập từ mọi nơi qua internet.
- Cloudinary cung cấp đường dẫn toàn cầu, tối ưu tốc độ tải file.

Độ tin cậy cao:

- Nhờ cơ chế replica của MongoDB và hạ tầng Cloudinary, hệ thống hoạt động ổn định ngay cả khi một node gặp sự cố.

Dễ bảo trì và nâng cấp:

- Kiến trúc module hóa: frontend, backend, cơ sở dữ liệu và lưu trữ file độc lập, dễ nâng cấp từng phần.

CHƯƠNG 4. BẢO MẬT VÀ HIỆU NĂNG TRONG HỆ THỐNG PHÂN TÁN

1. Thách thức bảo mật trong hệ thống phân tán

Hệ thống phân tán mang lại nhiều lợi ích về mở rộng và hiệu năng, nhưng cũng tiềm ẩn các rủi ro bảo mật, đặc biệt trong lĩnh vực y tế – nơi dữ liệu bệnh nhân là thông tin nhạy cảm. Một số thách thức chính:

(1) Rò rỉ thông tin bệnh nhân

Dữ liệu cá nhân, hồ sơ khám bệnh, đơn thuốc có thể bị truy cập trái phép nếu không được mã hóa hoặc bảo vệ đúng cách.

Vi phạm quy định bảo mật y tế (ví dụ: HIPAA ở Mỹ) có thể gây hậu quả pháp lý nghiêm trọng.

(2) Tấn công trung gian (Man-in-the-Middle)

Giao tiếp giữa frontend – backend hoặc backend – dịch vụ lưu trữ (Cloudinary) có nguy cơ bị chèn mã độc nếu không dùng HTTPS hoặc không xác thực token.

(3) Tấn công JWT Token

JWT hết hạn hoặc bị đánh cắp (qua XSS, CSRF) có thể cho phép hacker truy cập dữ liệu nhạy cảm.

Không triển khai cơ chế token refresh hoặc blacklist token sẽ làm tăng nguy cơ.

(4) Injection & XSS

Dữ liệu nhập từ người dùng (form khám bệnh, tên, địa chỉ) nếu không được lọc có thể dẫn đến NoSQL Injection hoặc Cross-Site Scripting.

(5) Upload file độc hại

Tính năng upload hồ sơ khám bệnh có thể bị lợi dụng để tải lên file chứa mã độc, nếu không kiểm tra loại file và quét virus.

2. Chiến lược bảo mật đề xuất cho hệ thống

Để đảm bảo an toàn cho dữ liệu và hạn chế rủi ro, hệ thống quản lý phòng khám áp dụng các giải pháp sau:

(1) Mã hóa dữ liệu và giao thức bảo mật

HTTPS cho toàn bộ giao tiếp giữa client – server.

Mã hóa mật khẩu bằng bcrypt trước khi lưu vào MongoDB.

Sử dụng TLS khi kết nối đến Cloudinary để tránh rò rỉ dữ liệu trong quá trình upload.

(2) Quản lý và bảo vệ JWT

Sử dụng JWT với thời gian sống ngắn (ví dụ: 15 phút) và refresh token để hạn chế nguy cơ token bị lợi dụng lâu dài.

Lưu token trong HTTP-only cookie thay vì localStorage để giảm nguy cơ XSS.

Triển khai middleware kiểm tra token ở tất cả API quan trọng.

(3) Kiểm soát phân quyền nghiêm ngặt

Xây dựng cơ chế Role-based Access Control (RBAC):

Admin: toàn quyền hệ thống.

Doctor: chỉ xem và sửa hồ sơ bệnh nhân của mình.

Patient: chỉ xem hồ sơ cá nhân.

Các API kiểm tra role trước khi xử lý yêu cầu.

(4) Ngăn chặn tấn công Injection & XSS

Sử dụng mongoose với cơ chế truy vấn an toàn để tránh NoSQL Injection.

Lọc và mã hóa dữ liệu đầu vào bằng thư viện như validator.js.

Sử dụng helmet middleware để bảo vệ khỏi XSS.

(5) Bảo mật upload file

Chỉ cho phép upload định dạng ảnh/pdf.

Quét file bằng dịch vụ antivirus trước khi lưu vào Cloudinary.

Kích hoạt signed upload trong Cloudinary để xác thực yêu cầu hợp lệ.

3. Thách thức hiệu năng trong hệ thống phân tán

Một số vấn đề có thể ảnh hưởng đến hiệu suất của hệ thống:

(1) Tải cao khi số lượng người dùng lớn

Khi nhiều bệnh nhân và bác sĩ truy cập cùng lúc (đặt lịch, xem hồ sơ), backend có thể bị quá tải.

(2) Truy vấn dữ liệu phức tạp

Truy vấn lọc bệnh nhân hoặc tìm kiếm hồ sơ có thể chậm nếu không có chỉ mục.

(3) Upload và hiển thị file lớn

Hồ sơ có nhiều file ảnh hoặc PDF khiến tốc độ tải chậm nếu không tối ưu.

(4) Độ trễ mạng

Hệ thống phân tán có nhiều điểm (client, server, Cloudinary, MongoDB cluster), dẫn đến độ trễ khi xử lý.

4. Kỹ thuật tối ưu hiệu năng cho hệ thống

Để giảm tải và tăng tốc độ phản hồi, có thể áp dụng các kỹ thuật sau:

(1) Sử dụng kiến trúc microservices

Tách API thành các service nhỏ: User Service, Medical Record Service, File Service để dễ mở rộng độc lập.

(2) Tối ưu cơ sở dữ liệu

Tạo index cho các trường hay tìm kiếm (tên bệnh nhân, số điện thoại).

Sử dụng aggregate pipeline cho báo cáo, tránh tải dữ liệu không cần thiết.

(3) Cache dữ liệu

Áp dụng Redis để cache kết quả truy vấn thường xuyên (ví dụ: danh sách bác sĩ, lịch khám).

Cache token xác thực để giảm thời gian kiểm tra.

(4) Tối ưu upload và hiển thị file

Dùng Cloudinary transformation để resize ảnh khi hiển thị.

Giảm dung lượng file trước khi lưu.

(5) Load balancing và scaling

Dùng Nginx hoặc AWS ELB để cân bằng tải backend.

Triển khai backend dạng container (Docker) để dễ scale.

5. Liên hệ tới hệ thống quản lý phòng khám

Áp dụng các giải pháp trên giúp hệ thống:

Đảm bảo an toàn cho dữ liệu bệnh nhân (tránh rò rỉ thông tin y tế).

Tăng khả năng chịu tải khi nhiều người dùng đồng thời (đặt lịch, upload hồ sơ).

Giảm thời gian phản hồi (từ 2–3 giây xuống dưới 1 giây cho các thao tác cơ bản).

Giữ trải nghiệm mượt mà ngay cả khi tích hợp nhiều tính năng mới trong tương lai (AI chẩn đoán, chatbot hỗ trợ).

CHƯƠNG 5. ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

1. Đánh giá hệ thống hiện tại

Sau quá trình thiết kế và triển khai theo kiến trúc phân tán, hệ thống quản lý phòng khám hiện tại có một số ưu điểm nổi bật và tồn tại một số hạn chế.

Ưu điểm:

Kiến trúc phân tán linh hoạt: Hệ thống backend sử dụng Node.js với Express, frontend dùng React (Vite) cho hiệu năng cao, dữ liệu được lưu trên MongoDB, file được lưu trữ an toàn trên Cloudinary.

Khả năng mở rộng: Có thể triển khai theo mô hình microservices và dễ dàng scale ngang khi lượng người dùng tăng.

Bảo mật tương đối tốt: Đăng nhập qua JWT, phân quyền theo role (admin, doctor, patient), mã hóa mật khẩu bằng bcrypt, giao tiếp HTTPS.

Tính năng đầy đủ cho quản lý phòng khám cơ bản:

Quản lý bệnh nhân, bác sĩ, phiếu khám (thêm, sửa, xóa).

Phân quyền theo vai trò.

Giao diện hiện đại và dễ sử dụng: Sử dụng React + Vite cho tốc độ load nhanh.

Hạn chế:

Chưa có cơ chế tối ưu tải lớn: Chưa triển khai caching (Redis), load balancing hoặc horizontal scaling thực tế.

Chưa có tính năng nâng cao: Đặt lịch khám trực tuyến, gửi thông báo tự động (SMS/Email), thanh toán online.

Giới hạn về bảo mật nâng cao: Chưa có cơ chế refresh token hoàn chỉnh, chưa áp dụng xác thực hai yếu tố (2FA).

Chưa có giám sát và logging chuyên sâu: Thiếu các công cụ như ELK stack hoặc Prometheus + Grafana để theo dõi hiệu năng.

2. Định hướng phát triển

Để đáp ứng nhu cầu thực tế và đảm bảo hệ thống vận hành tốt khi số lượng người dùng tăng, cần mở rộng theo các hướng sau:

(1) Nâng cao bảo mật

Triển khai refresh token + token blacklist.

Thêm xác thực hai lớp (2FA) cho tài khoản admin và bác sĩ.

Áp dụng CORS chặt chẽ và bảo vệ API chống CSRF.

(2) Cải thiện hiệu năng

Áp dụng Redis caching cho các truy vấn phổ biến.

Sử dụng CDN cho Cloudinary để tối ưu phân phối ảnh và tài liệu.

Triển khai kiến trúc microservices và load balancing bằng Nginx hoặc Kubernetes.

(3) Thêm tính năng nâng cao

Đặt lịch hẹn trực tuyến và gửi thông báo tự động qua Email/SMS.

Thanh toán trực tuyến qua ví điện tử hoặc thẻ ngân hàng.

Tích hợp AI hỗ trợ chẩn đoán sơ bộ dựa trên triệu chứng bệnh nhân nhập vào.

(4) Tối ưu trải nghiệm người dùng

Ứng dụng Progressive Web App (PWA) cho phép bệnh nhân đặt lịch ngay trên di động.

Hỗ trợ đa ngôn ngữ (Tiếng Việt, Tiếng Anh) để phục vụ khách hàng quốc tế.

(5) Giám sát và phân tích dữ liệu

Triển khai Prometheus + Grafana để giám sát hiệu năng.

Tích hợp ELK stack (Elasticsearch, Logstash, Kibana) để phân tích log.

Áp dụng machine learning để phân tích dữ liệu lịch sử khám chữa bệnh.

CHƯƠNG 6. KẾT LUẬN

1. Tóm tắt nội dung

Báo cáo đã nghiên cứu và triển khai một hệ thống quản lý phòng khám dựa trên kiến trúc phân tán, với các thành phần và công nghệ chủ đạo:

- Backend: Node.js + Express.
- Frontend: React + Vite cho hiệu năng cao.
- Cơ sở dữ liệu: MongoDB lưu trữ dữ liệu dạng document.
- Lưu trữ file: Cloudinary để quản lý hình ảnh và tài liệu.
- Bảo mật: JWT cho xác thực, phân quyền theo 3 vai trò (Admin, Doctor, Patient).
- Chức năng chính: Quản lý bệnh nhân, bác sĩ, phiếu khám; đăng nhập phân quyền.

Trong báo cáo, chúng ta đã:

- Phân tích cơ sở lý thuyết và đặc điểm của hệ thống phân tán.
- Xây dựng kiến trúc phân tán cho phần mềm quản lý phòng khám.
- Đề xuất các chiến lược bảo mật và tối ưu hiệu năng.
- Đưa ra đánh giá hệ thống hiện tại và định hướng phát triển trong tương lai.

2. Đóng góp khoa học và thực tiễn

Đóng góp khoa học:

Minh họa cách ứng dụng kiến trúc phân tán vào một hệ thống thực tế (quản lý phòng khám).

Là ví dụ cụ thể về tích hợp nhiều công nghệ (Node.js, React, MongoDB, Cloudinary) trong một kiến trúc linh hoạt.

Đóng góp thực tiễn:

Hệ thống có thể giảm tải quy trình quản lý phòng khám thủ công, tăng hiệu quả làm việc.

Hỗ trợ bác sĩ, bệnh nhân, và admin tương tác trực tuyến nhanh chóng và bảo mật.

Có khả năng mở rộng quy mô để phục vụ nhiều phòng khám hoặc bệnh viện.

3. Hạn chế của nghiên cứu

Tính năng còn hạn chế: Thiếu các tính năng như đặt lịch khám online, hỗ trợ thanh toán trực tuyến, chat trực tuyến với bác sỹ…

Khả năng chịu tải chưa tối ưu: Chưa triển khai cân bằng tải, chưa có cơ chế caching và microservices thực tế.

Bảo mật nâng cao chưa hoàn thiện: Chưa có xác thực hai yếu tố (2FA), chưa triển khai IDS/IPS.

4. Hướng nghiên cứu và phát triển tiếp theo

Trong tương lai, hệ thống sẽ được nâng cấp theo các hướng:

Tích hợp các tính năng nâng cao: Đặt lịch khám, nhắc lịch tự động, thanh toán online.

Triển khai microservices và cân bằng tải để hỗ trợ hàng nghìn người dùng đồng thời.

Nâng cấp bảo mật với 2FA, quản lý token nâng cao, giám sát bảo mật thời gian thực.

Ứng dụng AI để hỗ trợ chẩn đoán sơ bộ và phân tích dữ liệu bệnh nhân.