BỘ CÔNG AN

**TRƯỜNG ĐẠI HỌC KỸ THUẬT - HẬU CẦN CAND**

**ĐỒ ÁN TỐT NGHIỆP**

**NGHIÊN CỨU VÀ XÂY DỰNG WEBSITE HỖ TRỢ CHẨN ĐOÁN BỆNH DA LIỄU DỰA TRÊN TRIỆU CHỨNG**

**Học viên: Nguyễn Thị Xuân Hồng**

**Ngành/chuyên ngành: AN TOÀN THÔNG TIN CAND**

**Niên khoá: 2021-2026**

**Bắc Ninh, năm 2025**

BỘ CÔNG AN

**TRƯỜNG ĐẠI HỌC KỸ THUẬT - HẬU CẦN CAND**

**NGHIÊN CỨU VÀ XÂY DỰNG WEBSITE HỖ TRỢ CHẨN ĐOÁN BỆNH DA LIỄU DỰA TRÊN TRIỆU CHỨNG**

**Bắc Ninh, năm 2025**

**Ngành/chuyên ngành: Kỹ thuật CAND**

**Người hướng dẫn: ThS. Phạm Anh Tuấn**

**Học viên thực hiện: Nguyễn Thị Xuân Hồng**

**Lớp: B1D11**

**Khoá: D11**

**Hệ: Đại học chính quy**

**ĐỒ ÁN TỐT NGHIỆP**

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả được trình bày trong Đồ án/Khoá luận tốt nghiệp là công trình nghiên cứu của tôi dưới sự hướng dẫn của thầy Thạc sĩ Phạm Anh Tuấn. Các số liệu, kết quả trong Đồ án/Khoá luận tốt nghiệp là hoàn toàn trung thực và chưa được công bố trong bất kỳ công trình nào trước đây. Các kết quả được sử dụng để tham khảo đều đã được trích dẫn đầy đủ và theo đúng quy định. Tôi cũng xin cam đoan rằng trong bài luận này không có tài liệu chứa bí mật nhà nước hoặc thông tin nội bộ không được phép công bố./.

|  |  |
| --- | --- |
|  | *Bắc Ninh, ngày…tháng…năm 2025*  **Tác giả**  *(Ký, ghi rõ họ tên)*  Nguyễn Thị Xuân Hồng |

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thiện Đồ án/Khoá luận tốt nghiệp này, tác giả đã nhận được nhiều sự giúp đỡ và đóng góp quý báu:

Đầu tiên, tác giả xin bày tỏ lòng cảm ơn sâu sắc tới thầy giáo hướng dẫn là **Thiếu tá, ThS Phạm Anh Tuấn,** Giảng viên khoa Công nghệ và An toàn thông tin và các thầy cô giáo trong **Khoa Công nghệ và An toàn thông tin**. Các thầy cô đã luôn tận tình giúp đỡ, hỗ trợ cũng như tạo mọi điều kiện tốt nhất cho tác giả trong suốt quá trình nghiên cứu và hoàn thiện Đồ án/Khoá luận tốt nghiệp này.

Tác giả xin chân thành cảm ơn **Trường Đại học Kỹ thuật - Hậu cần CAND** đã tạo điều kiện thuận lợi để tác giả hoàn thành nhiệm vụ. Tác giả cũng xin cảm ơn **Cục Cảnh sát quản lý hành chính về trật tự xã hội** đã tạo điều kiện cho phép tác giả được tham gia nghiên cứu trong thời gian làm Đồ án/Khoá luận tốt nghiệp.

Cuối cùng, tác giả xin bày tỏ lòng cảm ơn đến gia đình, anh em, bạn bè, các đồng chí đã động viên và cổ vũ tác giả trong suốt thời gian nghiên cứu.

Xin trân trọng cảm ơn!

MỞ ĐẦU

Trong bối cảnh **công nghệ số phát triển mạnh mẽ**, **trí tuệ nhân tạo (AI)** đang dần trở thành công cụ hỗ trợ hiệu quả trong nhiều lĩnh vực của đời sống, đặc biệt là trong **ngành y học**. Theo nghiên cứu được công bố trên tạp chí Lancet Digital Health, các mô hình học sâu (deep learning) có khả năng **phân tích triệu chứng và chẩn đoán chính xác 87% ca bệnh**, với **tỷ lệ điều trị thành công đạt 93%**; trong khi đó, các bác sĩ con người đạt tương ứng **86% và 91%**.  
Những con số này cho thấy **AI hoàn toàn có thể trở thành công cụ hỗ trợ đắc lực cho đội ngũ y bác sĩ** trong việc **chẩn đoán ban đầu** và **đề xuất hướng điều trị phù hợp**.

Từ thực tiễn đó, đề tài **“Nghiên cứu và xây dựng website hỗ trợ chẩn đoán bệnh da liễu dựa trên triệu chứng”** được thực hiện nhằm **ứng dụng công nghệ trí tuệ nhân tạo và web** vào lĩnh vực **da liễu**.  
Hệ thống hướng đến mục tiêu:

Cung cấp thông tin đáng tin cậy về các bệnh da liễu thông qua các bài viết tổng hợp từ nguồn y khoa uy tín.

Xây dựng giao diện thân thiện, dễ sử dụng, giúp người dùng tra cứu triệu chứng, đặt câu hỏi và nhận tư vấn sơ bộ.

Góp phần nâng cao nhận thức cộng đồng về các bệnh da liễu, hỗ trợ phòng ngừa và điều trị hợp lý.

# TỔNG QUAN VÀ CƠ SỞ LÝ THUYẾT

## Tổng quan

### Đặt vấn đề

Trong thời đại công nghệ số phát triển mạnh mẽ, trí tuệ nhân tạo (Artificial Intelligence - AI) đang trở thành một trong những công nghệ nền tảng, có khả năng hỗ trợ, thay thế con người trong nhiều lĩnh vực khác nhau.

Đặc biệt trong ngành y học, AI được ứng dụng rộng rãi trong việc phân tích dữ liệu, nhận dạng hình ảnh y khoa, chẩn đoán bệnh và hỗ trợ điều trị.

Theo công bố trên tạp chí Lancet Digital Health, các mô hình học sâu (deep learning) có thể chẩn đoán chính xác 87% ca bệnh và tỷ lệ điều trị thành công đạt 93%, trong khi các bác sĩ đạt tương ứng 86% và 91%.

Từ thực tế này, có thể thấy việc áp dụng AI trong hỗ trợ chẩn đoán ban đầu các bệnh lý, đặc biệt là bệnh da liễu, là hoàn toàn khả thi và mang lại hiệu quả cao.

Đề tài “Nghiên cứu và xây dựng website hỗ trợ chẩn đoán bệnh da liễu dựa trên triệu chứng” được thực hiện nhằm ứng dụng công nghệ trí tuệ nhân tạo và nền tảng web trong việc hỗ trợ người dùng nhận biết, tra cứu và phòng ngừa bệnh da liễu một cách nhanh chóng, khoa học và thuận tiện.

### Mục tiêu nghiên cứu

Đề tài hướng đến các mục tiêu chính sau:

1. Nghiên cứu tổng quan về trí tuệ nhân tạo và các ứng dụng trong lĩnh vực y học, đặc biệt là chẩn đoán bệnh da liễu.
2. Xây dựng hệ thống website có khả năng:
   1. Cung cấp thông tin đáng tin cậy về các bệnh da liễu phổ biến.
   2. Cho phép người dùng nhập triệu chứng ban đầu để hệ thống gợi ý khả năng mắc bệnh và hướng xử lý sơ bộ.
   3. Tích hợp chức năng hỏi đáp, chia sẻ kiến thức giữa người dùng và chuyên gia y tế.
3. Đánh giá hiệu quả thực tiễn của hệ thống trong việc hỗ trợ người dân tra cứu và nhận biết sớm bệnh da liễu.

### Phạm vi và giới hạn nghiên cứu

Phạm vi nghiên cứu:

Nghiên cứu, phân tích và ứng dụng AI trong chẩn đoán sơ bộ bệnh da liễu dựa trên triệu chứng mô tả hoặc hình ảnh minh họa.

Xây dựng website hỗ trợ tra cứu và tương tác giữa người dùng và hệ thống AI.

Nguồn dữ liệu được lấy từ các tài liệu y khoa, website chính thống, và các công trình nghiên cứu khoa học có độ tin cậy cao.

Giới hạn nghiên cứu:

Hệ thống chỉ hỗ trợ chẩn đoán sơ bộ, không thay thế cho ý kiến chuyên môn của bác sĩ.

Phạm vi bệnh lý giới hạn trong một số bệnh da liễu phổ biến (ví dụ: viêm da, mụn trứng cá, nấm da, chàm, vảy nến,...).

Kết quả dựa trên mô hình AI thử nghiệm và dữ liệu được thu thập ở mức tham khảo, chưa mang tính lâm sàng.

## CƠ SỞ LÝ THUYẾT

### Kiến trúc hệ thống

Hệ thống được xây dựng theo mô hình kiến trúc client-server hiện đại, kết hợp microservices cho các chức năng chuyên biệt. Kiến trúc này đảm bảo tính mô-đun hóa cao, khả năng bảo trì và mở rộng dễ dàng.

**Các thành phần chính:**

1. **Frontend:**

Ứng dụng web được xây dựng bằng ReactJS kết hợp Vite nhằm tối ưu tốc độ tải trang, hiệu năng render và trải nghiệm người dùng.

1. **Backend chính:**

Dịch vụ RESTful API được phát triển bằng Spring Boot, đóng vai trò trung tâm xử lý nghiệp vụ, xác thực người dùng và giao tiếp với cơ sở dữ liệu cũng như các dịch vụ khác.

1. **Cơ sở dữ liệu:**

Sử dụng MySQL để lưu trữ dữ liệu có cấu trúc (structured data) như thông tin người dùng, bài viết y tế, danh mục và lịch sử tương tác.

1. **AI Service:**

Dịch vụ Chatbot được phát triển độc lập bằng Python, tích hợp mô hình Gemini 2.0 của Google để xử lý ngôn ngữ tự nhiên, hỗ trợ người dùng tư vấn và tra cứu thông tin y tế.

|  |
| --- |
|  |
| **Hình 1.1: Sơ đồ kiến trúc tổng quan** |

Các module trao đổi dữ liệu thông qua API calls sử dụng giao thức HTTP/HTTPS, đảm bảo tính bảo mật và khả năng tích hợp cao giữa các hệ thống.

**Luồng dữ liệu tổng quát:**

Người dùng thao tác và gửi yêu cầu thông qua giao diện web ReactJS.

Frontend gửi yêu cầu đến Spring Boot API để thực hiện các chức năng như đăng nhập, đăng bài, hoặc truy xuất nội dung.

Spring Boot xử lý nghiệp vụ, sau đó tương tác với MySQL để lưu trữ hoặc truy vấn dữ liệu.

Khi người dùng đặt câu hỏi cho chatbot, Spring Boot chuyển tiếp yêu cầu sang Python AI Service.

Dịch vụ AI sử dụng Gemini 2.0 để phân tích, sinh phản hồi và gửi kết quả trở lại cho Frontend, hiển thị trực quan cho người dùng.

### Giới thiệu về công nghệ Spring Boot

#### Spring Boot là gì?

#### Spring Boot là một framework mã nguồn mở được xây dựng trên nền tảng Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển và triển khai các ứng dụng Java độc lập, sẵn sàng cho production.

#### Khác với Spring truyền thống vốn yêu cầu nhiều cấu hình XML phức tạp, Spring Boot cho phép lập trình viên xây dựng **ứng dụng độc lập (standalone)**, **sẵn sàng triển khai (production-ready)** chỉ với một vài cấu hình đơn giản.

#### Spring Boot tích hợp sẵn **máy chủ web nhúng** (như Tomcat, Jetty, hoặc Undertow), giúp ứng dụng có thể **chạy trực tiếp dưới dạng tệp JAR** mà không cần triển khai lên máy chủ bên ngoài. Ngoài ra, Spring Boot còn cung cấp các **Starter Dependencies** - tập hợp các thư viện được cấu hình sẵn, hỗ trợ lập trình viên nhanh chóng khởi tạo và phát triển ứng dụng theo từng mục đích cụ thể (ví dụ: spring-boot-starter-web, spring-boot-starter-data-jpa, spring-boot-starter-security, …).

#### Lịch sử phát triển của Spring Boot

Năm 2002, dự án Spring Framework ra đời bởi Rod Johnson, với mục tiêu thay thế mô hình lập trình phức tạp của Java Enterprise (EJB).

Năm 2013, nhóm phát triển Pivotal Software giới thiệu Spring Boot nhằm giải quyết nhược điểm của Spring truyền thống như:

* Quá nhiều cấu hình XML.
* Cần nhiều công đoạn cài đặt môi trường.
* Thiếu khả năng chạy độc lập.

Tháng 4/2014, Spring Boot 1.0 chính thức được phát hành, đánh dấu bước ngoặt lớn của hệ sinh thái Spring.

Từ năm 2018 đến nay, các phiên bản 2.x và 3.x liên tục được cập nhật, bổ sung hỗ trợ cho: Spring WebFlux (Reactive Programming), Kotlin, Groovy, Microservices, Cloud Native Development (Spring Cloud)

Nhờ đó, Spring Boot trở thành một trong những nền tảng phổ biến và mạnh mẽ nhất cho lập trình web bằng Java hiện nay.

#### Kiến trúc của Spring Boot

Cấu trúc một ứng dụng Spring Boot điển hình gồm bốn lớp chính:

|  |  |  |
| --- | --- | --- |
|  | **Chức năng** | **Thành phần chính** |
| **Presentation** | Nhận các yêu cầu (request) từ client (trình duyệt web, ứng dụng mobile hoặc API client).  Xử lý routing và định tuyến URL tới các phương thức xử lý thích hợp.  Trả kết quả về client (thường dưới dạng JSON, HTML hoặc XML). | @Controller, @RestController  @RequestMapping, @GetMapping, @PostMapping |
| **Bussiness** | Xử lý logic nghiệp vụ của ứng dụng.  Quản lý giao dịch (transaction management) và phối hợp các thao tác với dữ liệu. | @Service để đánh dấu các class xử lý nghiệp vụ.  @Transactional để đảm bảo tính toàn vẹn dữ liệu trong các giao dịch. |
| **Persistence** | Tương tác trực tiếp với cơ sở dữ liệu để truy vấn, cập nhật, xóa hoặc thêm dữ liệu.  Ẩn các chi tiết truy vấn SQL khỏi Business Layer, đảm bảo loose coupling. | @Repository để đánh dấu các class truy xuất dữ liệu.  Spring Data JPA (JpaRepository, CrudRepository) hoặc JDBC/Hibernate. |
| **Database** | Lưu trữ dữ liệu lâu dài cho ứng dụng.  Cung cấp khả năng truy vấn và cập nhật dữ liệu hiệu quả. | RDBMS: MySQL, PostgreSQL, Oracle  NoSQL: MongoDB, Redis (nếu ứng dụng sử dụng dữ liệu phi cấu trúc hoặc cache) |

|  |
| --- |
| IMG_256 |
| Hình 1.2: Kiến trúc của một ứng dụng Spring Boot bốn lớp |

Spring Boot còn cung cấp các mô-đun hỗ trợ mở rộng như:

* Spring Boot Starter: Tập hợp phụ thuộc theo nhóm chức năng.
* Spring Boot Auto Configuration: Tự động cấu hình dựa trên thư viện có trong classpath.
* Embedded Server: Cho phép chạy trực tiếp ứng dụng qua Tomcat hoặc Jetty tích hợp sẵn.
* Spring Boot Actuator: Giám sát, kiểm tra sức khỏe hệ thống và thống kê hoạt động.

#### Ưu và nhược điểm của Spring Boot

**Ưu điểm:**

Phát triển nhanh chóng

* Auto-configuration: Tự động cấu hình các thành phần cần thiết.
* Starters: Chỉ cần khai báo spring-boot-starter-data-jpa để tích hợp MySQL.
* Embedded server: Ứng dụng chạy độc lập với Tomcat tích hợp sẵn.

Bảo trì dễ dàng

* Separation of Concerns: Mỗi tầng có nhiệm vụ riêng biệt.
* Dependency Injection: Dễ dàng thay đổi implementation mà không ảnh hưởng code.
* Clean code: Code rõ ràng, dễ đọc và mở rộng.

Production-ready

* Spring Boot Actuator: Giám sát sức khỏe ứng dụng.
* Security: Dễ dàng tích hợp Spring Security.
* Testing: Hỗ trợ mạnh mẽ cho việc viết unit test và integration test.

**Nhược điểm:**

Mặc dù Spring Boot có nhiều ưu điểm như dễ cấu hình, khởi tạo nhanh, hỗ trợ microservices, nhưng nó vẫn tồn tại một số hạn chế:

* Ẩn quá nhiều logic tự động: Khó kiểm soát và debug khi xảy ra lỗi phức tạp.
* Kích thước ứng dụng lớn: File JAR độc lập kèm máy chủ nhúng tốn nhiều dung lượng.
* Tiêu thụ bộ nhớ cao: Do tích hợp nhiều mô-đun và máy chủ nhúng.
* Khó tùy chỉnh sâu: Ghi đè cấu hình mặc định đôi khi phức tạp.
* Học tập ban đầu khó: Annotation, dependency và auto-configuration gây bối rối cho người mới.
* Phụ thuộc vào Spring Ecosystem: Khó di chuyển sang framework khác.

### Giới thiệu về ReactJS + Vite

#### ReactJS là gì?

ReactJS là thư viện JavaScript chuyên biệt cho việc xây dựng giao diện người dùng động, dựa trên kiến trúc component (thành phần giao diện). Điểm nổi bật của ReactJS nằm ở khả năng quản lý trạng thái (state management) hiệu quả, kết hợp cùng Virtual DOM giúp tối ưu quá trình render giao diện.ReactJS giúp lập trình viên tạo ra giao diện phức tạp nhưng hiệu suất cao, thông qua cơ chế Virtual DOM và component-based architecture.

* Virtual DOM: ReactJS sử dụng Virtual DOM – một bản sao nhẹ của DOM thực. Khi có sự thay đổi dữ liệu, ReactJS so sánh Virtual DOM với DOM hiện tại (diffing algorithm) và chỉ cập nhật các phần thay đổi thực sự trên trình duyệt, giúp tăng hiệu suất vượt trội khi so với các phương pháp thao tác DOM truyền thống.
* JSX: ReactJS sử dụng JSX (JavaScript XML) – cú pháp mở rộng của JavaScript cho phép lập trình viên viết các thành phần giao diện theo phong cách khai báo, kết hợp logic và UI trong cùng một khối code, giúp tăng tính trực quan, giảm thiểu lỗi khi xây dựng UI phức tạp.
* One-way Data Flow: ReactJS triển khai luồng dữ liệu một chiều (one-way data binding), giúp dữ liệu luôn di chuyển theo một hướng từ component cha xuống component con, tăng tính predictability (dễ dự đoán), dễ debug và giảm rủi ro lỗi khi ứng dụng phát triển ở quy mô lớn.
* Component Reusability: Các component có thể được đóng gói, tái sử dụng và chia sẻ, phù hợp với mô hình phát triển phần mềm hiện đại, tối ưu hóa thời gian xây dựng và bảo trì hệ thống.

React thường được kết hợp với các thư viện quản lý state như Redux, Recoil, hoặc MobX, và các framework routing như React Router để xây dựng SPA (Single Page Application).

#### Lịch sử phát triển của ReactJS

Năm 2011: ReactJS được phát triển nội bộ tại Facebook để giải quyết các vấn đề về hiệu suất và bảo trì giao diện trên News Feed.

Năm 2013: ReactJS được mã nguồn mở (Open Source), cộng đồng bắt đầu phát triển các công cụ hỗ trợ.

Từ 2015 đến nay: React liên tục cập nhật các phiên bản mới, bổ sung:

* React Hooks (2018) – giúp quản lý state và lifecycle trong functional component.
* Concurrent Mode, Suspense – nâng cao khả năng xử lý đồng thời và tải dữ liệu linh hoạt.
* Hỗ trợ TypeScript và JSX cải tiến trải nghiệm lập trình.

Nhờ đó, ReactJS trở thành thư viện phổ biến nhất cho phát triển giao diện web hiện đại, đặc biệt trong các SPA và PWA (Progressive Web App).

2016 – nay: ReactJS phát triển mạnh mẽ với hàng loạt cải tiến:

* Fiber Reconciliation: Tái cấu trúc toàn bộ lõi xử lý của ReactJS, nâng cấp khả năng xử lý bất đồng bộ, tối ưu hiệu suất render, cho phép rendering ưu tiên (priority rendering).
* Hooks (2019): Cách tiếp cận mới trong quản lý state, side effects và logic chia sẻ giữa các component function, loại bỏ sự phụ thuộc vào class component.
* Concurrent Mode: Hỗ trợ render không đồng bộ, giảm lag, tối ưu trải nghiệm người dùng trên các ứng dụng lớn.
* Server Components, SSR & SSG: Hỗ trợ tối ưu SEO, tăng tốc độ tải trang với server-side rendering (Next.js), static site generation.

ReactJS hiện là nền tảng UI chủ lực của Facebook, Instagram, WhatsApp, cùng hàng nghìn công ty lớn nhỏ trên toàn thế giới.

#### Kiến trúc của ReactJS

Kiến trúc của ReactJS được xây dựng dựa trên triết lý UI là hàm của trạng thái (state), tập trung giải quyết hiệu quả vấn đề cập nhật giao diện động với dữ liệu biến đổi liên tục. React phân tách logic và giao diện bằng mô hình component hóa, tối ưu tốc độ thông qua Virtual DOM, sử dụng luồng dữ liệu một chiều giúp dễ kiểm soát, đồng thời hỗ trợ quản lý vòng đời của từng thành phần giao diện.

|  |
| --- |
| IMG_256 |
| Hình 1.3: Sơ đồ kiến trúc hoạt động của reactJS |

##### Virtual DOM

Virtual DOM (Document Object Model ảo) là một lớp trừu tượng hóa DOM thực trên trình duyệt, giúp giảm tải các thao tác cập nhật tốn kém tài nguyên.

Cơ chế hoạt động:

Khi dữ liệu thay đổi, React tạo một Virtual DOM mới để mô tả lại trạng thái UI.

Hệ thống so sánh (reconciliation) xác định sự khác biệt giữa Virtual DOM mới và cũ bằng thuật toán diffing.

Chỉ những node thực sự thay đổi mới được cập nhật lên DOM thật thông qua batch update (cập nhật theo lô), hạn chế render lại toàn bộ cây DOM.

Virtual DOM mang lại:

* Hiệu năng cao hơn đáng kể so với thao tác trực tiếp trên DOM.
* Tăng tốc độ phản hồi giao diện ngay cả với cấu trúc UI lớn, phức tạp.
* Giảm thiểu lỗi do thao tác DOM không đồng bộ hoặc không kiểm soát.

##### Component-Based Architecture

React tổ chức ứng dụng thành các component nhỏ độc lập. Theo nghiên cứu của Kitchenham et al. (2007) về "Guidelines for performing Systematic Literature Reviews in Software Engineering", kiến trúc dựa trên component mang lại lợi ích về tính tái sử dụng và khả năng bảo trì. Báo cáo từ State of JS Survey 2023 cho thấy React components có tỷ lệ được sử dụng cao nhờ khả năng đóng gói và tái sử dụng. Theo tài liệu của Facebook Engineering, sự cô lập giữa các component giúp giảm độ phụ thuộc và tăng sự gắn kết trong các ứng dụng quy mô lớn.

Mỗi component bao gồm:

* Logic hiển thị (render UI)
* Logic xử lý dữ liệu, sự kiện
* State và props riêng biệt

Đặc điểm chuyên sâu:

* Isolation: Component độc lập, tránh xung đột dữ liệu và logic.
* Reusability: Có thể tái sử dụng component cho nhiều vị trí, nhiều dự án khác nhau.
* Composability: Dễ dàng lồng ghép, tổ hợp các component để xây dựng UI phức tạp.
* Testability: Component dễ kiểm thử đơn vị (unit test) nhờ tính độc lập và xác định rõ input/output.

Phân loại component:

* Functional component: Sử dụng function, thường kết hợp hook cho xử lý logic.
* Class component: Sử dụng class, hỗ trợ đầy đủ lifecycle methods truyền thống.

#### Ưu và nhược điểm của ReactJS

React JS là một thư viện JavaScript được ưa chuộng rộng rãi hiện nay nhờ khả năng xây dựng các giao diện người dùng (UI) hiệu quả và linh hoạt. Tuy nhiên, bên cạnh những ưu điểm nổi bật, React JS cũng tồn tại một số hạn chế nhất định. Tiếp theo Vietnix sẽ đi sâu phân tích các ưu nhược điểm của React JS để giúp bạn đưa ra lựa chọn phù hợp nhất cho dự án của mình.

**Ưu điểm**

ReactJS phù hợp với đa thể loại website: ReactJS hỗ trợ đáng kể trong việc khởi tạo website đơn giản hơn vì bạn chỉ cần sử dụng JavaScript và HTML, các công cụ và tính năng khác đã được ReactJS cung cấp để áp dụng cho nhiều trường hợp khác nhau.

Dễ dàng tái sử dụng các Component: Bạn chỉ cần tốn thời gian vào việc xây dựng ban đầu và có thể sử dụng lại trong các dự án sau này nếu bạn xây dựng các component đủ linh hoạt và tốt, có thể đáp ứng các yêu cầu của nhiều dự án khác nhau. Ngoài ra, không chỉ có ReactJS mà các framework hiện nay như Flutter cũng cho phép chúng ta thực hiện điều này.

Sử dụng được cho cả ứng dụng di động: ReactJS thường được biết đến phổ biến trong việc sử dụng để phát triển các website. Nhưng nếu cần xây dựng và phát triển ứng dụng di động, React Native cũng là một lựa chọn tuyệt vời, đây là một framework khác được nghiên cứu và phát triển bởi Facebook. Với React Native, các component có thể được chia sẻ dễ dàng hoặc được tái sử dụng trong các Business Logic trong ứng dụng.

Tính hiệu quả cao của React JS: Nhờ việc tạo ra các DOM ảo giúp các component tồn tại nên cải thiện hiệu suất. Không những vậy, React JS cũng giúp DOM biết những thay đổi nào phải cập nhật và tiến hành thực hiện, giúp tránh xảy ra những thao tác lãng phí trên DOM.

Có lợi cho SEO: ReactJS, bản chất là một thư viện JavaScript, đã được crawl và index bởi Google Search Engine. Tuy nhiên, bạn cũng cần sử dụng một số thư viện bổ sung để có thể tối ưu hóa SEO hiệu quả nhất có thể.

Hỗ trợ viết code JS đơn giản hơn đáng kể: Nội dung cú pháp Javascript mở rộng cho phép kết hợp Javascript với code HTML. Do đó, bạn hoàn toàn có thể thêm vào các đoạn HTML trong render mà không phải nối chuỗi, đây cũng đồng thời là đặc tính mới của React JS khi có thể sử dụng bộ biến đổi JXS để khởi tạo các đối tượng HTML từ các đoạn HTML.

Nhiều công cụ mở rộng giúp debug dễ dàng: Sử dụng các công cụ mở rộng dành riêng cho React JS sẽ giúp debug code nhanh chóng và dễ dàng hơn, giúp tăng tốc độ release sản phẩm và tốc độ coding.

Test giao diện dễ dàng: Do virtual DOM được cài hoàn toàn bằng JS nên việc sử dụng React JS rất đơn giản trong việc viết các test case giao diện.

Công cụ phát triển website phổ biến nhất: Theo số liệu từ Google Trends ở Việt Nam, cụ thể là các trang web tuyển dụng hàng đầu như Topdev, Itviec,.. bạn sẽ thấy nhu cầu công việc cho vị trí React Developer là rất lớn, đi kèm mức lương hấp dẫn. Điều này cho thấy “độ phủ sóng” đáng kể của ReactJS trên thị trường Việt Nam hiện nay.

**Nhược điểm**

Do React chỉ là View Library chứ không phải một MVC framework nên chỉ có thể đáp ứng nhu cầu của tầng View. Hơn nữa, react cũng không có model và controller nên phải sử dụng các thư viện khác.

Bạn sẽ phải cấu hình lại từ đầu nếu tích hợp react JS vào các framework MVC truyền thống

So với các kích thước khác, React khá lớn, khoảng 35KB, tương đương với Angular khoảng 39KB. Thế nhưng, Angular là một framework hoàn chỉnh

Người mới bắt đầu sử dụng sẽ mất nhiều thời gian để học do khá phức tạp

**Ứng dụng thực tế của ReactJS**

Công dụng chủ yếu của ReactJS là để xây dựng giao diện người dùng (UI), giúp nâng cao hiệu suất hoạt động của chương trình. Thư viện này tận dụng DOM ảo (một đối tượng JavaScript) để tối ưu hóa hiệu quả của ứng dụng.

### Cơ sở lý thuyết về mô hình AI Gemini 2.0

#### Nền tảng hình thành.

#### Gemini 2.0 là thế hệ tiếp theo của dòng mô hình lớn đa phương thức (multimodal large model) của Google DeepMind / Google AI, nhằm mở rộng khả năng từ chỉ xử lý văn bản sang nhập/xuất đa phương thức: văn bản, hình ảnh, âm thanh, video.

#### Mô hình được thiết kế cho kỷ nguyên “agentic” - tức là không chỉ trả lời mà còn thực hiện hành động, gọi công cụ (tool use), tích hợp sâu với môi trường/thông tin bên ngoài (search engine, mã code, video…) để trở thành trợ lý thông minh hơn.

#### Google cho biết Gemini 2.0 được huấn luyện và chạy inference hoàn toàn trên phần cứng TPU thế hệ mới - ví dụ sixth-generation TPUs (Trillium) - để tối ưu hiệu suất.

Khung thời gian: thông báo chính thức vào tháng 12 2024.

#### Kiến trúc và công nghệ chính

Dưới đây là các thành phần kiến trúc và công nghệ nổi bật mà có thể thu thập được từ tài liệu công khai.

* 2.1. Đầu vào – đầu ra đa phương thức (Multimodal I/O)

Gemini 2.0 hỗ trợ đầu vào: văn bản (text), hình ảnh (image), âm thanh (audio), video.

Đầu ra: tối thiểu là văn bản; trong một số phiên bản preview đã hoặc sẽ hỗ trợ hình ảnh, âm thanh (text-to-speech), và hình ảnh tạo mới (text-to-image) và/hoặc video.

Ví dụ: phiên bản “Flash Image” preview hỗ trợ đầu ra hình ảnh.

Cửa sổ ngữ cảnh (Context window) lớn

Gemini 2.0 Flash có cửa sổ ngữ cảnh tới 1 triệu token (input) trong một số mô hình.

Điều này cho phép xử lý “ngữ cảnh dài” – ví dụ đoạn hội thoại dài, tài liệu lớn, hoặc nhiều file cùng lúc.

* Khả năng sử dụng công cụ & gọi hàm (Tool-use / Function calling)

Mô hình được trang bị khả năng gọi tới các công cụ bên ngoài như tìm kiếm web (Google Search), thực thi mã code, hoặc gọi tới hàm (function-calling) do người dùng định nghĩa.

Điều này giúp tăng tính “tác nhân” (agentic) – mô hình không chỉ sinh câu trả lời mà có thể hành động (ví dụ chạy code, tra dữ liệu, kết nối dịch vụ).

* Hiệu năng & tối ưu (Performance & Efficiency)

Gemini 2.0 Flash được thiết kế để có độ trễ thấp, tốc độ nhanh hơn so với phiên bản trước (ví dụ Gemini 1.5) dù vẫn giữ hoặc nâng cao chất lượng.

Việc huấn luyện và inference trên phần cứng chuyên dụng (TPU thế hệ mới) giúp quản lý chi phí, tăng tốc và tối ưu hóa khả năng triển khai ở quy mô lớn.

* Tư duy/giải thích (Reasoning / Explainability)

Có phiên bản “Flash Thinking” (thử nghiệm) của Gemini 2.0, trong đó mô hình có khả năng “nghĩ trước khi trả lời” và giải thích quy trình tư duy (chain‐of‐thought) của mình.

Điều này được xem như bước hướng tới các trợ lý có khả năng giải quyết các bài toán phức tạp, nhiều bước, đa lĩnh vực.

* Hệ thống huấn luyện (Training stack)

Mặc dù Google không công bố chi tiết số lượng tham số (parameters) và toàn bộ kiến trúc bên trong, có thông tin rằng mô hình được huấn luyện trên TPUs, sử dụng công nghệ phần mềm như JAX/XLA (được nhắc trong nghiên cứu phụ).

Đối với dữ liệu: có thể bao gồm dữ liệu đa phương thức lớn, nhiều ngôn ngữ, nhiều nhiệm vụ (pre-training + fine-tuning) – nhưng cụ thể chưa rõ công khai hoàn toàn.

#### Đặc trưng nổi bật

Dưới đây là các điểm nổi bật giúp bạn hiểu “what makes Gemini 2.0 khác biệt”.

Multimodal sâu: không chỉ hiểu văn bản mà còn hình ảnh, âm thanh, video – mở ra các ứng dụng mới.

Agentic & tool-use: có khả năng sử dụng các công cụ bên ngoài, truy cập thông tin trực tuyến, thực thi mã, là bước gần hơn đến trợ lý thông minh thực thụ.

Cửa sổ ngữ cảnh rất lớn: giúp làm việc với các đoạn văn bản/tài liệu rất dài, hoặc nhiều luồng thông tin trong cùng một lần tương tác.

Tối ưu hiệu suất: Google nhấn mạnh rằng Gemini 2.0 nhanh hơn phiên bản trước, với chi phí và độ trễ được cải thiện.

Khả năng tư duy/giải thích: với phiên bản “Thinking”, mô hình có khả năng phân tách bước giải quyết, cung cấp “suy nghĩ” ý nghĩa hơn, không chỉ “ra kết quả”.

Khả năng mở rộng & tích hợp: Bằng cách hỗ trợ tool-calling, multichannel I/O, mô hình dễ tích hợp vào nhiều ứng dụng (search engine, trợ lý, nghiên cứu, sản xuất nội dung…).

Ứng dụng thực tiễn rộng: Từ hỗ trợ tìm kiếm thông minh hơn, trợ lý nghiên cứu (‘Deep Research’), đến các hệ thống robot/lập trình (ăn sâu hơn nhờ Gemini 2.0 làm “não” – xem phần ứng dụng).

#### Nhược điểm

Một mô hình lớn đa modal sẽ tiêu tốn rất nhiều tài nguyên (data, phần cứng, điện năng).

Xử lý “ngữ cảnh dài” hiệu quả cần giải pháp kỹ thuật như chunking, attention scale-up, caching.

Khi gọi tool/công cụ, cần đảm bảo an toàn (mô hình không bị lợi dụng để gọi hành động nguy hiểm) và tin cậy kết quả.

Multi-modal I/O và reasoning phức tạp: dễ mắc lỗi (hallucination, sai hiểu hình ảnh, âm thanh).

Việc triển khai sản phẩm cần cân nhắc hiệu suất (người dùng chờ lâu sẽ không tốt), chi phí, khả năng mở rộng.

# Phân tích và THIẾT KẾ Dự Án

## PHÂN TÍCH

### Mục tiêu hệ thống

Hệ thống hỗ trợ người dùng (bệnh nhân / công chúng) và bác sĩ sơ bộ trong việc nhận diện, phân loại và gợi ý bước xử trí ban đầu cho các bệnh da liễu dựa trên:

thông tin triệu chứng (mô tả bằng text),

lịch sử bệnh/các yếu tố nguy cơ (tuổi, tiền sử dị ứng, thuốc…).

Hệ thống gồm:

Website nội dung y khoa (bài viết, hình ảnh minh họa, hướng dẫn tự chăm sóc, FAQ, tài liệu tham khảo).

Chatbot y khoa: hội thoại dẫn dắt để thu thập triệu chứng, phân loại rủi ro, gợi ý khả năng chẩn đoán sơ bộ, đề xuất hành động tiếp theo (tư vấn tự chăm sóc / khám bác sĩ / cấp cứu).

Bảng quản trị (Admin): quản lý nội dung, người dùng, cấu hình web cơ bản)

### Yêu cầu chức năng

#### Website tra cứu thông tin y khoa.

FR-WEB-01: Quản lý bài viết y khoa

* Admin tạo/sửa/xóa bài viết, tag, danh mục chuyên đề (ví dụ: eczema, viêm nang lông, nấm da).
* Acceptance: Admin có thể đăng bài có tiêu đề, nội dung rich text, ảnh, nguồn tham khảo; bài hiển thị đúng trên trang công khai.

FR-WEB-02: Tìm kiếm & duyệt nội dung

* Người dùng tìm kiếm theo từ khóa, bộ lọc theo chuyên đề.
* Acceptance: Kết quả trả về < 2s cho truy vấn phổ biến; hiển thị phân trang/đề xuất liên quan.

FR-WEB-03: Trang tra cứu thông tin trên chatbot

* Người dùng nhập yêu cầu trực tiếp lên khung chat để nhận về phản hồi từ phía chatbot service.

#### Chatbot hỏi đáp y khoa

FR-CHAT-01 - Nhận diện & phân tích truy vấn người dùng

* Chatbot phân tích nội dung truy vấn đầu vào của người dùng nhằm phân loại đúng ý định. Nếu quá mơ hồ sẽ hỏi lại để làm rõ.

FR-CHAT-02 - Tìm kiếm dữ liệu trong cơ sở tri thức nội bộ

* Khi người dùng hỏi mộtchủ đề, chatbot sẽ gọi công cụ DatabaseSearchTool để tìm kiếm thông tin trong cơ sở dữ liệu nội bộ đã được thu thập.
* Trả về thông tin liên quan nhất với yêu cầu.
* Nếu chưa có thông tin hoặc thông tin không hợp lệ chatbot sẽ thông báo hoặc hỏi lại để tìm đúng yêu cầu của người dùng.

FR-CHAT-03 - Tra cứu mở rộng qua Internet

* Nếu cơ sở tri thức nội bộ không đủ hoặc thông tin cũ, chatbot gọi Google Search Tool để tra cứu trực tuyến thông tin y tế mới nhất.
* Bot lấy kết quả từ các nguồn đáng tin cậy (WHO, CDC, PubMed, Bộ Y tế, các bệnh viện lớn).
* Không hiển thị các nguồn không xác minh hoặc blog cá nhân.

FR-CHAT-04 - Trích xuất và tổng hợp thông tin y khoa

* Khi có dữ liệu từ search\_database và google\_search, chatbot gọi Extracted\_Info\_Agent để trích xuất thông tin quan trọng, rồi tổng hợp lại thành kết quả dễ hiểu.

FR-CHAT-05 - Quản lý ngữ cảnh hội thoại

* Chatbot lưu ngữ cảnh hội thoại ngắn hạn để hiểu mạch trao đổi (ví dụ: nhớ triệu chứng đã nêu, không hỏi lại liên tục).

FR-CHAT-06 - Giới hạn an toàn

* Chatbot phải phát hiện và từ chối các yêu cầu không phù hợp (ví dụ: hỏi cách tự dùng thuốc, điều trị bằng chất độc hại...).
* Không tạo hoặc suy luận thông tin ngoài phạm vi y tế chính thống.

### Yêu cầu về dữ liệu & mô hình AI

#### *Yêu cầu dữ liệu*

1. Dữ liệu từ người dùng (User Input)

Nguồn: Người dùng cuối (web hoặc app).

Định dạng: câu hỏi, triệu chứng, yêu cầu tra cứu y khoa.

Yêu cầu chất lượng: tiếng Việt, chính tả chuẩn, mô tả rõ ràng.

1. FastAPI Server

Chức năng dữ liệu:

Nhận HTTP Request từ client.

Streaming SSE: gửi dữ liệu trả về theo từng chunk (text, partial response).

Chuẩn hóa input trước khi chuyển cho Agent System.

Yêu cầu về dữ liệu:

* Validate input (text dài < 1000 ký tự, file ≤ 5MB).
* Thêm metadata: timestamp, user\_id/session\_id

1. Agent System

Gồm: Root Agent → Search Agent → Tool Executor

Input: query chuẩn hóa từ FastAPI.

Chức năng xử lý dữ liệu:

* Root Agent: quản lý luồng hội thoại, quyết định agent con nào xử lý.
* Search Agent: gọi công cụ tìm kiếm (cơ sở dữ liệu FAISS hoặc Google AI).
* Tool Executor: thực thi các tool (DatabaseSearchTool, Google Search, Extracted Info Agent).

Yêu cầu dữ liệu:

* Query phải được format chuẩn JSON, kèm metadata: ngôn ngữ, model version, context.
* Các agent trả về output JSON: text, confidence score, source (DB/Google AI).
* Lưu log output cho audit, debugging và huấn luyện cải thiện.

1. FAISS Database

Chức năng: lưu trữ vector embedding của các văn bản y khoa nội bộ.

Yêu cầu dữ liệu:

* Input: embedding vector (từ text đã preprocess).
* Output: danh sách các document tương đồng (top-k).

1. Google AI (Gemini)

Chức năng: tạo/gợi ý câu trả lời, trích xuất thông tin, tóm tắt.

Yêu cầu dữ liệu:

Input: prompt dạng text (có thể kèm context JSON từ database).

Output: JSON text đã tóm tắt, confidence score.

Data privacy: không lưu trữ dữ liệu người dùng lâu dài trên server của bên thứ ba.

### Phân tích yêu cầu

#### Đăc tả phần mềm

##### Use Case

|  |
| --- |
|  |
| Hình 2.1: Sơ đồ Use Case tổng quát chức năng Người dùng. |

* Xác định các Actor của ứng dụng

|  |  |
| --- | --- |
| Khách | Người dùng chưa đăng nhập có thể vào tìm kiếm thông tin từ các bài viết của trang web đồng thời tạo tài khoản để tham gia vào website. |
| Người dùng | Người dùng là người đã đăng nhập vào hệ thống. Mở rộng hơn khách chưa đăng nhập ở việc có thể tương tác với chatbot y khoa. |
| Quản trị viên | Là người có quyền truy cập vào tất cả các chức năng của hệ thống. Có thể mở dashboard để quản lý việc thêm, tạo, xóa bài viết và danh mục cũng như cấu hình cơ bản website |

* Xác định các Use case của hệ thống

1. Đăng nhập
   1. Tác nhân: Quản trị viên và người dùng.
   2. Điều kiện:

* Quản trị viên: Đã có tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
* Người dùng: Đã đã đăng ký tài khoản trên hệ thống.
  1. Mô tả:
* Quản trị viên sẽ sử dụng tài khoản đã được cung cấp để đăng nhập nhập vào hệ thống.
* Khách hàng dùng tài khoản đã đăng ký để đăng nhập vào hệ thống

1. Đăng ký
   1. Tác nhân: Khách.
   2. Điều kiện: Chưa có tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp hoặc chưa đăng ký tài khoản.
   3. Mô tả: Khách hàng có thể đăng ký tài khoản trên ứng dụng.
2. Tìm kiếm bài viết y khoa
   1. Tác nhân: Quản trị viên, người dùng và khách.
   2. Mô tả: Quản trị viên, người dùng và khách tìm kiếm bài viết thông qua việc nhập từ khóa trên thanh tìm kiếm.
3. Xem bài viết y khoa
   1. Tác nhân: Quản trị viên, người dùng và khách.
   2. Mô tả: Quản trị viên, người dùng và khách xem bài viết đã đăng được public trên hệ thống .
4. Xem danh mục bài viết
   1. Tác nhân: Quản trị viên, người dùng và khách.
   2. Mô tả: Quản trị viên, người dùng và khách có thể xem các danh mục được public trên hệ thống.
5. Tương tác với chatbot y khoa
   1. Tác nhân: Quản trị viên và người dùng.
   2. Điều kiện: Đã đăng nhập tài khoản trên hệ thống.
   3. Mô tả: Khi gửi một yêu cầu tới phía chatbot, chatbot sẽ xử lý thông tin và tìm kiếm kết quả có độ chính xác cao từ data có sẵn hoặc trên Internet để phản hồi về phía người dùng yêu cầu.
6. Tạo bài viết
   1. Tác nhân: Quản trị viên.
   2. Điều kiện: Quản trị viên đăng nhập tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
   3. Mô tả: Quản trị viên tạo bài viết mới và thêm vào danh mục có sẵn.
7. Chỉnh sửa bài viết
   1. Tác nhân: Quản trị viên.
   2. Điều kiện: Quản trị viên đăng nhập tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
   3. Mô tả: Quản trị viên có thể chỉnh sửa bài viết đã tạo hoặc ẩn bài viết đi.
8. Xóa bài viết
   1. Tác nhân: Quản trị viên.
   2. Điều kiện: Quản trị viên đăng nhập tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
   3. Mô tả: Quản trị viên có thể xóa bài viết đã tạo hoặc ẩn bài viết đã tồn tại.
9. Quản lý danh mục
   1. Tác nhân: Quản trị viên.
   2. Điều kiện: Quản trị viên đăng nhập tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
   3. Mô tả: Quản trị viên tạo mới, chỉnh sửa hoặc xóa danh mục bài viết.
10. Quản lý thông tin trang web.
    1. Tác nhân: Quản trị viên.
    2. Điều kiện: Quản trị viên đăng nhập tài khoản trên ứng dụng do đội ngũ phát triển hệ thống cung cấp.
    3. Mô tả: Quản trị viên tạo mới, chỉnh sửa hoặc xóa các thông tin của trang web như banner quảng cáo, logo, thông tin chân trang.

* Đăng nhập
  + Use Case

|  |
| --- |
|  |
| Hình 2.2: Use Case đăng nhập |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Đăng nhập |
| Mã Use Case | UC-001 |
| Actor(s) | quản trị viên, người dùng |
| Mô tả | Use case này cho phép người dùng và quản trị viên đăng nhập vào hệ thống bằng tên tài khoản và mật khẩu |
| Mục đích | Xác thực danh tính người dùng trước khi cho phép truy cập vào các chức năng của hệ thống |
| Điều kiện tiên quyết | 1. Người dùng/Quản trị viên đã có tài khoản trong hệ thống.  2. Hệ thống website và dịch vụ chatbot đang hoạt động bình thường.  3. Người dùng/Quản trị viên chưa đăng nhập vào hệ thống.  4. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet. |
| Điều kiện kết thúc | Thành công:  - Người dùng được xác thực thành công.  - Phiên làm việc (session) được tạo.  - Người dùng được chuyển hướng đến trang chủ (hoặc trang đích) tương ứng với vai trò (Quản trị viên hoặc Người dùng thông thường).  Thất bại:  - Hệ thống hiển thị thông báo lỗi.  - Người dùng không được xác thực và không thể truy cập hệ thống.  - Phiên làm việc không được tạo. |
| Luồng sự kiện chính | 1. Người dùng truy cập trang đăng nhập của hệ thống.  2. Hệ thống hiển thị form đăng nhập với các trường "Tên tài khoản" và "Mật khẩu".  3. Người dùng nhập tên tài khoản và mật khẩu, sau đó gửi form.  4. Hệ thống kiểm tra tính hợp lệ của dữ liệu nhập (không để trống, đúng định dạng).  5. Hệ thống xác thực thông tin đăng nhập bằng cách so sánh với cơ sở dữ liệu người dùng.  6. Nếu thông tin chính xác, hệ thống tạo một phiên làm việc (session) và ghi nhận trạng thái đã đăng nhập.  7. Hệ thống chuyển hướng người dùng đến trang chủ mặc định. |
| Luồng sự kiện thay thế | Đăng nhập thất bại (thông tin không chính xác)  5a. Tên tài khoản hoặc mật khẩu không khớp với dữ liệu trong hệ thống.  5b. Hệ thống hiển thị thông báo lỗi: "Tên đăng nhập hoặc mật khẩu không chính xác. Vui lòng thử lại."  5c. Use case tiếp tục từ Bước 2, người dùng có thể thử lại. |

* + Sequence Diagram

|  |
| --- |
|  |

* Đăng ký
  + Use Case

|  |
| --- |
|  |
| Hình 2.3: Use Case đăng ký |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Đăng ký |
| Mã Use Case | UC-002 |
| Actor(s) | Khách (người dùng chưa có tài khoản) |
| Mô tả | Use case này cho phép khách hàng đăng ký tài khoản mới trong hệ thống web tích hợp chatbot bằng cách cung cấp thông tin cá nhân |
| Mục đích | Tạo tài khoản mới cho người dùng để có thể truy cập và sử dụng các chức năng chatbot của hệ thống |
| Điều kiện tiên quyết | 1. Người dùng chưa có tài khoản trong hệ thống.  2. Hệ thống đang hoạt động bình thường.  3. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet. |
| Điều kiện kết thúc | Thành công:  - Tài khoản mới được tạo thành công trong hệ thống.  - Thông tin người dùng được lưu vào cơ sở dữ liệu.  - Hệ thống hiển thị thông báo đăng ký thành công.  - Người dùng được chuyển đến trang đăng nhập.  Thất bại:  - Tài khoản không được tạo.  - Hệ thống hiển thị thông báo lỗi.  - Người dùng vẫn ở trên trang đăng ký và có thể sửa thông tin. |
| Luồng sự kiện chính | 1. Khách truy cập vào trang đăng ký tài khoản.  2. Hệ thống hiển thị form đăng ký với các trường: tên tài khoản, email, số điện thoại, mật khẩu, nhập lại mật khẩu.  3. Khách nhập đầy đủ thông tin vào form đăng ký.  4. Khách nhấn nút "Đăng ký".  5. Hệ thống kiểm tra tính hợp lệ của thông tin đã nhập (định dạng, độ dài, v.v.).  6. Hệ thống kiểm tra tên tài khoản, email, số điện thoại chưa tồn tại trong cơ sở dữ liệu.  7. Hệ thống mã hóa mật khẩu và lưu thông tin tài khoản vào cơ sở dữ liệu.  8. Hệ thống hiển thị thông báo "Đăng ký thành công".  9. Hệ thống tự động chuyển hướng người dùng đến trang đăng nhập. |
| Luồng sự kiện thay thế | Luồng thay thế A: Thông tin để trống  5a. Hệ thống phát hiện một hoặc nhiều trường bắt buộc bị để trống.  5b. Hệ thống hiển thị thông báo: "Vui lòng nhập đầy đủ thông tin bắt buộc".  5c. Use case tiếp tục từ Bước 2.  Luồng thay thế B: Tên tài khoản đã tồn tại  6a. Hệ thống phát hiện tên tài khoản đã được sử dụng.  6b. Hệ thống hiển thị thông báo: "Tên tài khoản đã tồn tại. Vui lòng chọn tên khác".  6c. Use case tiếp tục từ Bước 2.  Luồng thay thế C: Email đã tồn tại  6a. Hệ thống phát hiện email đã được đăng ký.  6b. Hệ thống hiển thị thông báo: "Email đã được sử dụng. Vui lòng sử dụng email khác hoặc đăng nhập".  6c. Use case tiếp tục từ Bước 2.  Luồng thay thế D: Mật khẩu không khớp  5a. Hệ thống phát hiện mật khẩu và nhập lại mật khẩu không giống nhau.  5b. Hệ thống hiển thị thông báo: "Mật khẩu nhập lại không khớp".  5c. Use case tiếp tục từ Bước 2.  Luồng thay thế E: Thông tin không hợp lệ  5a. Hệ thống phát hiện thông tin không đúng định dạng (ví dụ: email sai format, số điện thoại không hợp lệ, mật khẩu quá ngắn).  5b. Hệ thống hiển thị thông báo lỗi cụ thể cho từng trường (ví dụ: "Email không hợp lệ", "Mật khẩu phải có ít nhất 8 ký tự").  5c. Use case tiếp tục từ Bước 2. |

* + Sequence Diagram

|  |
| --- |
|  |
|  |

* Tìm kiếm bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.4: Use Case tìm bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Tìm kiếm bài viết y khoa |
| Mã Use Case | UC-003 |
| Actor(s) | quản trị viên, người dùng, khách |
| Mô tả | Use case này cho phép Khách, Người dùng và Quản trị viên tìm kiếm các bài viết y khoa trong hệ thống bằng cách nhập từ khóa. |
| Mục đích | Cung cấp khả năng truy xuất nhanh chóng và chính xác đến kho tàng bài viết y khoa dựa trên nhu cầu thông tin của người dùng. |
| Điều kiện tiên quyết | 1. Hệ thống đang hoạt động bình thường.  2. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet.  3. Hệ thống có sẵn ít nhất một số bài viết y khoa trong cơ sở dữ liệu. |
| Điều kiện kết thúc | Thành công:  - Hệ thống trả về danh sách các bài viết y khoa phù hợp với từ khóa tìm kiếm.  - Người dùng có thể xem trước hoặc đọc toàn văn các bài viết trong kết quả tìm kiếm.  Thất bại:  - Hệ thống không tìm thấy bài viết nào phù hợp.  - Hệ thống hiển thị thông báo "Không tìm thấy kết quả phù hợp". |
| Luồng sự kiện chính | 1. Actor truy cập vào trang/chức năng tìm kiếm bài viết y khoa của hệ thống.  2. Hệ thống hiển thị ô nhập liệu tìm kiếm.  3. Actor nhập từ khóa liên quan đến chủ đề y khoa cần tìm vào ô tìm kiếm.  4. Actor gửi yêu cầu tìm kiếm (bằng cách nhấn nút "Tìm kiếm" hoặc phím Enter).  5. Hệ thống thực hiện tìm kiếm trong cơ sở dữ liệu các bài viết y khoa dựa trên từ khóa.  6. Hệ thống hiển thị danh sách kết quả các bài viết phù hợp (bao gồm tiêu đề, mô tả ngắn, hình ảnh đại diện...).  7. Actor có thể chọn một bài viết từ danh sách kết quả để xem chi tiết. |
| Luồng sự kiện thay thế | Luồng thay thế A: Không có kết quả phù hợp  5a. Hệ thống không tìm thấy bài viết nào khớp với từ khóa.  5b. Hệ thống hiển thị thông báo: "Không tìm thấy bài viết nào phù hợp với từ khóa '[từ khóa]'. Vui lòng thử lại với từ khóa khác."  5c. Use case tiếp tục từ Bước 2. |

n Sequence Diagram

|  |
| --- |
|  |

* Xem bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.5: Use Case tìm bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Xem bài viết y khoa |
| Mã Use Case | UC-004 |
| Actor(s) | quản trị viên, người dùng, khách |
| Mô tả | Use case này cho phép Khách, Người dùng và Quản trị viên xem nội dung chi tiết của một bài viết y khoa trong hệ thống. |
| Mục đích | Cung cấp thông tin y khoa chi tiết, đầy đủ cho người đọc. |
| Điều kiện tiên quyết | 1. Hệ thống đang hoạt động bình thường.  2. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet.  3. Bài viết tồn tại và có thể truy cập được trong hệ thống. |
| Điều kiện kết thúc | Thành công:  - Người dùng xem được toàn bộ nội dung bài viết.  Thất bại:  - Người dùng không thể xem được nội dung bài viết.  - Hệ thống hiển thị thông báo lỗi do không tìm thấy bài viết nào, hiện thị trang 404. |
| Luồng sự kiện chính | 1. Actor tìm thấy một bài viết thông qua danh sách bài viết, kết quả tìm kiếm hoặc đường link được chia sẻ.  2. Actor nhấp chọn vào bài viết muốn xem.  3. Hệ thống ghi nhận yêu cầu và truy xuất nội dung bài viết từ cơ sở dữ liệu.  4. Hệ thống hiển thị trang chi tiết bài viết, bao gồm: Tiêu đề, ảnh đại diện, tóm tắt, nội dung đầy đủ, ngày đăng, tác giả và các thông tin meta khác.  5. Actor cuộn trang để đọc toàn bộ nội dung bài viết. |
| Luồng sự kiện thay thế | Luồng thay thế A: Bài viết không tồn tại  3a. Hệ thống không tìm thấy bài viết theo ID hoặc slug được yêu cầu.  3b. Hệ thống hiển thị trang thông báo lỗi "404 - Bài viết không tìm thấy".  3c. Use case kết thúc. |

* Sequence Diagram

|  |
| --- |
|  |
|  |

* Xem danh mục bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.6: Use Case xem danh mục bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Xem danh mục bài viết |
| Mã Use Case | UC-005 |
| Actor(s) | quản trị viên, người dùng, khách |
| Mô tả | Use case này cho phép Khách, Người dùng và Quản trị viên xem danh sách các bài viết thuộc một danh mục cụ thể được hiển thị trên thanh header của website. |
| Mục đích | Giúp người dùng dễ dàng khám phá và lọc các bài viết y khoa theo từng chủ đề/chuyên mục đã được phân loại. |
| Điều kiện tiên quyết | 1. Hệ thống đang hoạt động bình thường.  2. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet.  3. Trên thanh header của hệ thống có hiển thị ít nhất một danh mục bài viết. |
| Điều kiện kết thúc | Thành công:  - Hệ thống hiển thị danh sách các bài viết thuộc danh mục mà người dùng đã chọn.  Thất bại:  - Hệ thống không thể tải được trang danh mục. |
| Luồng sự kiện chính | 1. Actor truy cập vào trang chủ hoặc bất kỳ trang nào của hệ thống.  2. Actor nhấp chuột vào một tên danh mục bài viết cụ thể trên thanh header (menu điều hướng chính).  3. Hệ thống ghi nhận yêu cầu và truy xuất danh sách tất cả các bài viết thuộc danh mục đó từ cơ sở dữ liệu.  4. Hệ thống hiển thị trang danh mục, bao gồm: Tên danh mục và danh sách các bài viết thuộc danh mục.  5. Actor có thể cuộn và duyệt qua danh sách bài viết. Actor có thể nhấp vào một bài viết bất kỳ để chuyển sang Use Case Xem bài viết y khoa (UC-004). |
| Luồng sự kiện thay thế |  |

* Sequence Diagram

|  |
| --- |
|  |
|  |

* Tương tác với Chatbot
  + Use Case

|  |
| --- |
|  |
| Hình 2.7: Use Case tương tác với chatbot |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Tương tác với Chatbot |
| Mã Use Case | UC-006 |
| Actor(s) | quản trị viên, người dùng |
| Mô tả | Use case này cho phép Người dùng và Quản trị viên tương tác với chatbot y khoa bằng cách gửi các yêu cầu (câu hỏi, thắc mắc). Một agent AI sẽ xử lý yêu cầu và gửi phản hồi phù hợp về cho người dùng. |
| Mục đích | Cung cấp phản hồi tự động, nhanh chóng và chính xác cho các câu hỏi về y khoa, giúp người dùng tra cứu thông tin một cách thuận tiện. |
| Điều kiện tiên quyết | 1. Người dùng/Quản trị viên đã đăng nhập vào hệ thống (UC-001).  2. Hệ thống website và dịch vụ chatbot/agent AI đang hoạt động bình thường.  3. Trình duyệt web của người dùng hoạt động tốt và có kết nối Internet. |
| Điều kiện kết thúc | Thành công:  - Người dùng gửi được yêu cầu tới chatbot.  - Hệ thống chatbot xử lý và trả về phản hồi phù hợp trong giao diện trò chuyện.  Thất bại:  - Người dùng không thể gửi được tin nhắn.  - Chatbot không phản hồi hoặc hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Actor (đã đăng nhập) truy cập vào giao diện trò chuyện với chatbot trên website.  2. Actor nhập câu hỏi hoặc yêu cầu liên quan đến y khoa vào ô nhập tin nhắn.  3. Actor gửi yêu cầu (bằng cách nhấn nút "Gửi" hoặc phím Enter).  4. Hệ thống hiển thị tin nhắn của actor trong khung chat.  5. Hệ thống chuyển yêu cầu của actor đến agent AI (chatbot) để xử lý.  6. Agent AI phân tích yêu cầu, xử lý và tạo ra câu trả lời phù hợp.  7. Hệ thống nhận phản hồi từ agent AI và hiển thị câu trả lời trong khung chat, dưới dạng tin nhắn đến từ chatbot.  8. Actor đọc phản hồi từ chatbot. |
| Luồng sự kiện thay thế | Luồng thay thế A: Chatbot không hiểu yêu cầu  6a. Agent AI không thể phân tích hoặc tìm thấy thông tin phù hợp với yêu cầu.  6b. Hệ thống hiển thị phản hồi mặc định từ chatbot: "Hệ thống yêu cầu bổ sung thông tin để trả về phản hồi tốt hơn."  6c. Use case tiếp tục từ Bước 2.  Luồng thay thế B: Yêu cầu chứa nội dung không phù hợp  6a. Agent AI phát hiện yêu cầu có nội dung độc hại, spam, tự ý kê đơn thuốc để tự điều trị hoặc không liên quan đến y khoa.  6b. Hệ thống hiển thị cảnh báo.  6c. Use case tiếp tục từ Bước 2. |

* Sequence Diagram

|  |
| --- |
|  |

* Tạo bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.8: Use Case tạo bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Tạo bài viết |
| Mã Use Case | UC-007 |
| Actor(s) | quản trị viên |
| Mô tả | Use case này cho phép Quản trị viên tạo và đăng một bài viết y khoa mới thông qua trang quản trị (Dashboard). |
| Mục đích | Bổ sung nội dung, thông tin y khoa mới vào hệ thống, làm phong phú kho tài nguyên cho người dùng. |
| Điều kiện tiên quyết | 1. Quản trị viên đã đăng nhập thành công vào hệ thống (UC-001).  2. Quản trị viên có quyền hạn phù hợp để tạo bài viết.  3. Hệ thống đang hoạt động bình thường.  4. Quản trị viên đã truy cập vào khu vực Dashboard. |
| Điều kiện kết thúc | Thành công:  - Bài viết mới được tạo, lưu trữ trong cơ sở dữ liệu và có thể được hiển thị trên website.  - Hệ thống hiển thị thông báo xác nhận đăng bài thành công.  Thất bại:  - Bài viết không được tạo.  - Hệ thống hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Quản trị viên điều hướng đến mục "Quản lý bài viết" hoặc tương đương trong Dashboard.  2. Quản trị viên nhấn vào nút "Tạo bài viết mới".  3. Hệ thống hiển thị form soạn thảo bài viết với các trường: Tiêu đề, Ảnh đại diện, Mô tả ngắn, Nội dung (có trình soạn thảo văn bản), Danh mục, Tags, v.v.  4. Quản trị viên nhập hoặc chọn tất cả thông tin cần thiết cho bài viết.  5. Quản trị viên nhấn nút "Lưu”.  6. Hệ thống kiểm tra tính hợp lệ của dữ liệu nhập.  7. Hệ thống lưu bài viết vào cơ sở dữ liệu.  8. Hệ thống hiển thị thông báo "Tạo bài viết thành công" và chuyển hướng Quản trị viên về trang quản lý bài viết hoặc trang chi tiết của bài viết vừa tạo. |
| Luồng sự kiện thay thế | Luồng thay thế A: Dữ liệu nhập không hợp lệ  6a. Hệ thống phát hiện dữ liệu nhập bị lỗi.  6b. Hệ thống hiển thị thông báo lỗi cụ thể tại các trường có vấn đề.  6c. Use case tiếp tục từ Bước 3, Quản trị viên có thể sửa các lỗi. |

* Sequence Diagram

|  |
| --- |
|  |

* Chỉnh sửa bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.9: Use Case chỉnh sửa bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Chỉnh sửa bài viết |
| Mã Use Case | UC-008 |
| Actor(s) | quản trị viên |
| Mô tả | Use case này cho phép Quản trị viên cập nhật, sửa đổi nội dung của một bài viết y khoa đã có trong hệ thống thông qua trang quản trị (Dashboard). |
| Mục đích | Duy trì tính chính xác, cập nhật và chất lượng của các bài viết y khoa theo thời gian. |
| Điều kiện tiên quyết | 1. Quản trị viên đã đăng nhập thành công vào hệ thống (UC-001).  2. Quản trị viên có quyền hạn phù hợp để chỉnh sửa bài viết.  3. Hệ thống đang hoạt động bình thường.  4. Bài viết cần chỉnh sửa đã tồn tại trong hệ thống. |
| Điều kiện kết thúc | Thành công:  - Thông tin cập nhật của bài viết được lưu trữ vào cơ sở dữ liệu.  - Hệ thống hiển thị thông báo xác nhận chỉnh sửa thành công.  Thất bại:  - Bài viết không được cập nhật.  - Hệ thống hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Quản trị viên điều hướng đến mục "Quản lý bài viết" trong Dashboard.  2. Hệ thống hiển thị danh sách các bài viết.  3. Quản trị viên tìm và chọn bài viết cần chỉnh sửa bằng cách click chuột vào biểu tượng sửa hình cây bút.  4. Hệ thống tải và hiển thị form chỉnh sửa bài viết với đầy đủ thông tin hiện tại của bài viết.  5. Quản trị viên thực hiện các thay đổi cần thiết lên các trường (tiêu đề, nội dung, ảnh đại diện, danh mục...).  6. Quản trị viên nhấn nút "Lưu thay đổi".  7. Hệ thống kiểm tra tính hợp lệ của dữ liệu nhập.  8. Hệ thống lưu các thay đổi vào cơ sở dữ liệu.  9. Hệ thống hiển thị thông báo "Cập nhật bài viết thành công". |
| Luồng sự kiện thay thế | Luồng thay thế A: Dữ liệu nhập không hợp lệ  7a. Hệ thống phát hiện dữ liệu nhập bị lỗi (ví dụ: thiếu tiêu đề sau khi sửa).  7b. Hệ thống hiển thị thông báo lỗi cụ thể tại các trường có vấn đề.  7c. Use case tiếp tục từ Bước 4, Quản trị viên có thể sửa các lỗi.  Luồng thay thế B: Hủy bỏ thao tác  5a. Quản trị viên nhấn nút "X" trước khi lưu.  5b. Hệ thống chuyển hướng Quản trị viên trở lại trang danh sách bài viết mà không lưu bất kỳ thay đổi nào.  5c. Use case kết thúc. |

* Sequence Diagram

|  |
| --- |
|  |

* Xóa bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.10: Use Case xóa bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Xóa bài viết |
| Mã Use Case | UC-009 |
| Actor(s) | quản trị viên. |
| Mô tả | Use case này cho phép Quản trị viên xóa vĩnh viễn một bài viết y khoa khỏi hệ thống thông qua trang quản trị (Dashboard). |
| Mục đích | Loại bỏ các bài viết không còn phù hợp, không chính xác hoặc vi phạm chính sách của hệ thống. |
| Điều kiện tiên quyết | 1. Quản trị viên đã đăng nhập thành công vào hệ thống (UC-001).  2. Quản trị viên có quyền hạn phù hợp để xóa bài viết.  3. Hệ thống đang hoạt động bình thường.  4. Bài viết cần xóa đã tồn tại trong hệ thống. |
| Điều kiện kết thúc | Thành công:  - Bài viết được xóa vĩnh viễn khỏi cơ sở dữ liệu.  - Hệ thống hiển thị thông báo xác nhận xóa thành công.  Thất bại:  - Bài viết không bị xóa.  - Hệ thống hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Quản trị viên điều hướng đến mục "Quản lý bài viết" trong Dashboard.  2. Hệ thống hiển thị danh sách các bài viết.  3. Quản trị viên tìm và chọn bài viết cần xóa.  4. Quản trị viên nhấn vào nút "Xóa" liên quan đến bài viết đó.  5. Hệ thống hiển thị hộp thoại xác nhận xóa với thông tin chi tiết về bài viết.  6. Quản trị viên xác nhận muốn xóa bài viết bằng cách nhấn nút "Xác nhận" trong hộp thoại.  7. Hệ thống thực hiện lệnh xóa bài viết khỏi cơ sở dữ liệu.  8. Hệ thống hiển thị thông báo "Xóa bài viết thành công".  9. Hệ thống tự động làm mới danh sách bài viết. |
| Luồng sự kiện thay thế | Luồng thay thế A: Hủy bỏ thao tác xóa  6a. Quản trị viên nhấn nút "Hủy" trong hộp thoại xác nhận xóa.  6b. Hệ thống đóng hộp thoại xác nhận.  6c. Bài viết không bị xóa, use case kết thúc. |

* Sequence Diagram

|  |
| --- |
|  |

* Quản lý danh mục bài viết
  + Use Case

|  |
| --- |
|  |
| Hình 2.11: Use Case quản lý danh mục bài viết |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Quản lý danh mục bài viết |
| Mã Use Case | UC-010 |
| Actor(s) | Quản trị viên |
| Mô tả | Use case này cho phép Quản trị viên thực hiện các thao tác CRUD (Tạo, Xem, Sửa, Xóa) để quản lý các danh mục bài viết y khoa trong hệ thống thông qua trang quản trị (Dashboard). |
| Mục đích | Tổ chức và duy trì hệ thống danh mục bài viết một cách khoa học, giúp việc phân loại và tìm kiếm bài viết của người dùng được hiệu quả. |
| Điều kiện tiên quyết | 1. Quản trị viên đã đăng nhập thành công vào hệ thống (UC-001).  2. Quản trị viên có quyền hạn phù hợp để quản lý danh mục.  3. Hệ thống đang hoạt động bình thường.  4. Quản trị viên đã truy cập vào khu vực Dashboard. |
| Điều kiện kết thúc | Thành công:  - Quản trị viên hoàn thành một trong các thao tác con (Tạo, Sửa, Xóa) và hệ thống lưu lại thay đổi.  - Hệ thống hiển thị danh sách danh mục cập nhật.  Thất bại:  - Thao tác không được thực hiện (ví dụ: xóa thất bại do danh mục đang chứa bài viết).  - Hệ thống hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Quản trị viên điều hướng đến mục "Quản lý Danh mục" trong Dashboard.  2. Hệ thống hiển thị giao diện quản lý danh mục, bao gồm danh sách các danh mục hiện có và các nút chức năng (Thêm mới, Sửa, Xóa).  3. Quản trị viên thực hiện một trong các tác vụ mở rộng (Tạo mới, Chỉnh sửa, Xóa).  4. Sau khi hoàn thành hoặc hủy bỏ một tác vụ mở rộng, hệ thống quay lại và hiển thị danh sách danh mục. |
| Luồng sự kiện thay thế | Luồng mở rộng A: Tạo mới danh mục (Extends Bước 3)  A1. Quản trị viên nhấn nút "Thêm mới".  A2. Hệ thống hiển thị form tạo danh mục với các trường (Tên danh mục, Mô tả...).  A3. Quản trị viên nhập thông tin và nhấn "Lưu".  A4. Hệ thống kiểm tra tính hợp lệ (ví dụ: tên không trống, không trùng lặp).  A5. Hệ thống lưu danh mục mới và hiển thị thông báo thành công.  A6. Luồng trở về Bước 4 của luồng chính.  Luồng mở rộng B: Chỉnh sửa danh mục (Extends Bước 3)  B1. Quản trị viên chọn một danh mục từ danh sách và nhấn nút "Sửa".  B2. Hệ thống hiển thị form chỉnh sửa với thông tin hiện tại của danh mục.  B3. Quản trị viên chỉnh sửa thông tin và nhấn "Cập nhật".  B4. Hệ thống kiểm tra tính hợp lệ và lưu thay đổi.  B5. Hệ thống hiển thị thông báo cập nhật thành công.  B6. Luồng trở về Bước 4 của luồng chính.  Luồng mở rộng C: Xóa danh mục (Extends Bước 3)  C1. Quản trị viên chọn một danh mục từ danh sách và nhấn nút "Xóa".  C2. Hệ thống hiển thị hộp thoại xác nhận xóa.  C3. Quản trị viên xác nhận xóa.  C4. Hệ thống kiểm tra điều kiện xóa (ví dụ: danh mục không chứa bài viết nào).  C5. Hệ thống xóa danh mục và hiển thị thông báo thành công.  C6. Luồng trở về Bước 4 của luồng chính. |

* Sequence Diagram

|  |
| --- |
|  |

* Quản lý thông tin website
  + Use Case

|  |
| --- |
|  |
| Hình 2.12: Use Case quản lý thông tin website |

* + Đặc tả Use Case

|  |  |
| --- | --- |
| Thuộc tính | Mô tả |
| Tên Use Case | Quản lý thông tin website |
| Mã Use Case | UC-011 |
| Actor(s) | Quản trị viên |
| Mô tả | Use case này cho phép Quản trị viên cập nhật và quản lý các thông tin, giao diện cố định của trang web (như Banner, Logo, Footer) thông qua trang quản trị (Dashboard). |
| Mục đích | Duy trì và tùy chỉnh giao diện, thông tin chung của website một cách linh hoạt mà không cần can thiệp mã nguồn. |
| Điều kiện tiên quyết | 1. Quản trị viên đã đăng nhập thành công vào hệ thống (UC-001).  2. Quản trị viên có quyền hạn chỉnh sửa.  3. Hệ thống đang hoạt động bình thường.  4. Quản trị viên đã truy cập vào khu vực Dashboard. |
| Điều kiện kết thúc | Thành công:  - Quản trị viên hoàn thành việc chỉnh sửa một hoặc nhiều thành phần (Banner, Logo, Footer).  - Các thay đổi được lưu lại và hiển thị ngay lập tức trên giao diện người dùng.  Thất bại:  - Thay đổi không được lưu (ví dụ: file ảnh không hợp lệ, lỗi kết nối).  - Hệ thống hiển thị thông báo lỗi. |
| Luồng sự kiện chính | 1. Quản trị viên điều hướng đến mục các mục cần chỉnh sửa của website trong Dashboard.  2. Hệ thống hiển thị giao diện tổng quan cho phép quản lý các thiết lập chung của website.  3. Quản trị viên thực hiện một trong các tác vụ mở rộng để chỉnh sửa các thành phần cụ thể (Banner, Logo, Footer).  4. Sau khi hoàn thành hoặc hủy bỏ một tác vụ mở rộng, hệ thống quay lại giao diện quản lý tổng quan. |
| Luồng sự kiện thay thế | Luồng mở rộng A: Chỉnh sửa Banner (Extends Bước 3)  A1. Quản trị viên nhấn vào mục "Quản lý Banner" hoặc tương đương.  A2. Hệ thống hiển thị giao diện cho phép tải lên ảnh banner mới, chỉnh sửa liên kết, hoặc văn bản trên banner.  A3. Quản trị viên thực hiện các thay đổi và nhấn "Lưu".  A4. Hệ thống xử lý và lưu thông tin banner mới.  A5. Hệ thống hiển thị thông báo "Cập nhật Banner thành công".  A6. Luồng trở về Bước 4 của luồng chính.  Luồng mở rộng B: Chỉnh sửa Logo (Extends Bước 3)  B1. Quản trị viên nhấn vào mục "Đổi Logo".  B2. Hệ thống hiển thị giao diện cho phép tải lên một tệp ảnh logo mới.  B3. Quản trị viên chọn tệp ảnh từ máy tính và nhấn "Tải lên".  B4. Hệ thống kiểm tra tính hợp lệ của tệp (định dạng, kích thước).  B5. Hệ thống lưu logo mới và thay thế logo cũ.  B6. Hệ thống hiển thị thông báo "Đổi Logo thành công".  B7. Luồng trở về Bước 4 của luồng chính.  Luồng mở rộng C: Chỉnh sửa Footer (Extends Bước 3)  C1. Quản trị viên nhấn vào mục "Chỉnh sửa Footer".  C2. Hệ thống hiển thị trình soạn thảo văn bản với nội dung footer hiện tại.  C3. Quản trị viên chỉnh sửa thông tin (địa chỉ, số điện thoại, bản quyền, liên kết mạng xã hội...).  C4. Quản trị viên nhấn "Lưu thay đổi".  C5. Hệ thống lưu nội dung footer mới.  C6. Hệ thống hiển thị thông báo "Cập nhật Footer thành công".  C7. Luồng trở về Bước 4 của luồng chính. |

* Sequence Diagram

|  |
| --- |
|  |

##### Sơ đồ hoạt động

##### Sơ đồ lớp của website

|  |
| --- |
|  |
| Hình 2.13: ClassDiagram của hệ thống |

## THIẾT KẾ HỆ THỐNG

### Thiết kế CSDL

#### Xác định các thực thể

1. Thực Thể User (Người dùng)

users (PK Id, username, password, phone, is\_active, created\_at, updated\_at, FK avatar\_id)

Mô tả: Quản lý thông tin tài khoản người dùng và quản trị viên.

1. Thực Thể Post (Bài viết)

posts (PK Id, title, slug, content, status, FK category\_id, author\_id, created\_at, updated\_at, name, thumbnail\_id, footer\_id)

Mô tả: Lưu trữ thông tin và nội dung bài viết y khoa.

1. Thực Thể Category (Danh mục)

categories (PK Id, name, slug, description, level, parent\_id, is\_active, created\_at, updated\_at)

Mô tả: Quản lý danh mục bài viết theo cấu trúc phân cấp.

1. Thực thể Image (Hình ảnh)

images (PK Id, url, type)

Mô tả: Quản lý tập tin hình ảnh cho avatar, Logo, navicon và thumbnail.

1. Thực thể Footer (Chân trang)

footers (PK Id, company\_name, company\_full\_name, company\_description, office\_address, hotline, email, business\_license, copyright\_text, working\_hours, facebook\_url, youtube\_url, twitter\_url, is\_active)

Mô tả: Lưu trữ thông tin công ty và cấu hình chân trang

1. Thực Thể Role (Vai trò)

roles (PK Id, name, description, role\_slug, is\_active)

Mô tả: Định nghĩa các vai trò trong hệ thống

1. Thực Thể Permission (Quyền)

permission (PK Id, name, slug, description)

Mô tả: Quản lý các quyền thao tác trong hệ thống

1. Thực Thể RefreshToken (Token làm mới)

refresh\_token (PK Id, expiry\_date, revoked, FK user\_id)

Mô tả: Quản lý token xác thực cho phiên đăng nhập

1. Thực Thể UserRoles (Phân quyền người dùng)

user\_roles (PK FK user\_id, PK FK role\_id)

Mô tả: Bảng kết nối quan hệ nhiều-nhiều giữa User và Role

1. Thực Thể RolePermissions (Phân quyền vai trò)

role\_permissions (PK FK role\_id, PK FK permission\_id)

Mô tả: Bảng kết nối quan hệ nhiều-nhiều giữa Role và Permission

#### Mối quan hệ giữa các thực thể

1. User – Post

Một User có thể viết nhiều Post.

Một Post chỉ thuộc về một User (author).

Quan hệ: 1 – N

1. Post – Category

Một Category có thể có nhiều Post.

Một Post chỉ thuộc về một Category.

Quan hệ: 1 – N

1. User – RefreshToken

Một User có thể có nhiều RefreshToken.

Một RefreshToken chỉ thuộc về một User.

Quan hệ: 1 – N

1. User – Role (qua UserRoles)

Một User có thể có nhiều Role.

Một Role có thể gán cho nhiều User.

Quan hệ: N – N

1. Role – Permission (qua RolePermissions)

Một Role có nhiều Permission.

Một Permission có thể gán cho nhiều Role.

Quan hệ: N – N

1. Post – Footer / Post – Image

Một Post có thể liên kết đến một Footer và một Thumbnail (Image).

Một Image có thể được dùng cho nhiều Post.

Quan hệ: 1 – N (Image), 1 – 1 (Footer)

1. Category – Category (Parent)

Một Category có thể là cha của nhiều Category con.

Quan hệ: 1 – N (tự liên kết)

#### Mô hình thực thể ERD

|  |
| --- |
|  |
| Hình 2.14: Mô hình ERD của hệ thống |

#### Đặc tả bảng

Bảng users

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã định danh người dùng |
| username | VARCHAR(50) | NOT NULL, UNIQUE | Tên đăng nhập |
| password | VARCHAR(255) | NOT NULL | Mật khẩu |
| phone | VARCHAR(20) | NULL | Số điện thoại |
| is\_active | BOOLEAN | DEFAULT TRUE | Trạng thái hoạt động |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo |
| updated\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật |
| avatar\_id | BIGINT | FK → images(id) | Ảnh đại diện |

Bảng posts

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã định danh bài viết |
| title | VARCHAR(255) | NOT NULL | Tiêu đề bài viết |
| slug | VARCHAR(255) | NOT NULL, UNIQUE | URL thân thiện |
| content | TEXT | NOT NULL | Nội dung bài viết |
| status | VARCHAR(20) | DEFAULT 'draft' | Trạng thái bài viết |
| category\_id | BIGINT | FK → categories(id) | Danh mục bài viết |
| author\_id | BIGINT | FK → users(id) | Tác giả |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo |
| updated\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật |
| name | VARCHAR(255) | NULL | Tên hiển thị |
| thumbnail\_id | BIGINT | FK → images(id) | Ảnh đại diện |
| footer\_id | BIGINT | FK → footers(id) | Chân trang liên kết |

Bảng categories

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã định danh danh mục |
| name | VARCHAR(100) | NOT NULL | Tên danh mục |
| slug | VARCHAR(100) | NOT NULL, UNIQUE | URL thân thiện |
| description | TEXT | NULL | Mô tả danh mục |
| level | INT | DEFAULT 0 | Cấp độ danh mục |
| parent\_id | BIGINT | FK → categories(id) | Danh mục cha |
| is\_active | BOOLEAN | DEFAULT TRUE | Trạng thái hoạt động |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | Thời gian tạo |
| updated\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | Thời gian cập nhật |

Bảng images

|  |  |  |
| --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** |
| id | BIGINT | PK, auto\_increment |
| url | VARCHAR(255) | NOT NULL |
| type | VARCHAR(50) | NULL |

Bảng footers

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã định danh footer |
| company\_name | VARCHAR(255) | NOT NULL | Tên công ty rút gọn |
| company\_full\_name | VARCHAR(255) | NOT NULL | Tên công ty đầy đủ |
| company\_description | TEXT | NULL | Mô tả công ty |
| office\_address | VARCHAR(255) | NOT NULL | Địa chỉ văn phòng |
| hotline | VARCHAR(50) | NOT NULL | Số hotline |
| email | VARCHAR(100) | NOT NULL | Email hỗ trợ |
| business\_license | VARCHAR(255) | NULL | Giấy phép kinh doanh |
| copyright\_text | VARCHAR(255) | NULL | Nội dung bản quyền |
| working\_hours | VARCHAR(100) | NULL | Giờ làm việc |
| facebook\_url | VARCHAR(255) | NULL | Link Facebook |
| youtube\_url | VARCHAR(255) | NULL | Link Youtube |
| twitter\_url | VARCHAR(255) | NULL | Link Twitter |
| is\_active | BOOLEAN | DEFAULT TRUE | Trạng thái hoạt động |

Bảng roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã vai trò |
| name | VARCHAR(100) | NOT NULL | Tên vai trò |
| description | TEXT | NULL | Mô tả vai trò |
| role\_slug | VARCHAR(100) | NOT NULL, UNIQUE | Định danh vai trò |
| is\_active | BOOLEAN | DEFAULT TRUE | Trạng thái hoạt động |

Bảng permissions

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| id | BIGINT | PK, auto\_increment | Mã quyền |
| name | VARCHAR(100) | NOT NULL | Tên quyền |
| slug | VARCHAR(100) | NOT NULL, UNIQUE | Định danh quyền |
| description | TEXT | NULL | Mô tả quyền |

Bảng user\_roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| user\_id | BIGINT | PK, FK → users(id) | Người dùng |
| role\_id | BIGINT | PK, FK → roles(id) | Vai trò |

Bảng role\_permissions

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| role\_id | BIGINT | PK, FK → roles(id) | Vai trò |
| permission\_id | BIGINT | PK, FK → permissions(id) | Quyền |

#### Mô hình CSDL

|  |
| --- |
|  |
| Hình 2.15: Mô hình cơ sở dữ liệu |

### Thiết kế Agent System

#### Ý tưởng thiết kế

Mục Tiêu: Xây dựng chatbot thông minh có khả năng:

* Phân tích triệu chứng bệnh da liễu từ mô tả của người dùng
* Truy xuất thông tin từ cơ sở kiến thức y khoa chuyên ngành
* Đề xuất hướng xử lý ban đầu và khuyến nghị thăm khám
* Cảnh báo các trường hợp cần can thiệp y tế khẩn cấp

Đối Tượng Sử Dụng:

* Bệnh nhân cần tư vấn sơ bộ về các vấn đề da liễu
* Người dùng tìm kiếm thông tin về bệnh ngoài da
* Cộng đồng quan tâm đến chăm sóc sức khỏe da

#### Kiến trúc tổng quan

Hệ thống được thiết kế theo mô hình kiến trúc hướng tác tử (Agent-oriented Architecture) với luồng xử lý tuần tự qua các thành phần chính:

Luồng xử lý:

* Lớp giao diện người dùng: Người dùng tương tác thông qua giao diện web để gửi các truy vấn y khoa
* Lớp xử lý nghiệp vụ: Tác tử trung tâm (Central Agent) tiếp nhận và phân tích yêu cầu
* Lớp truy xuất tri thức: Hệ thống thực hiện truy vấn đa nguồn:
* Cơ sở dữ liệu vector FAISS: Truy xuất thông tin từ tri thức nội bộ đã được vector hóa
* Tìm kiếm thông tin ngoài: Thu thập dữ liệu từ internet thông qua Gemini 2.0 Flash
* Lớp tổng hợp và phản hồi: Kết quả được xử lý và truyền trả liên tục (streaming) thông qua FastAPI về giao diện người dùng

Đặc điểm nổi bật:

* Xử lý đa tác tử: Phối hợp nhiều tác tử chuyên biệt
* Truy xuất đa nguồn: Kết hợp tri thức nội bộ và thông tin cập nhật
* Phản hồi thời gian thực: Sử dụng kỹ thuật streaming để cung cấp kết quả liên tục

|  |
| --- |
|  |
| Hình 2.16: Kiến trúc cơ bản của hệ thống chatbot |

#### Quy trình xử lý yêu cầu

|  |
| --- |
|  |
| Hình 2.17: Sơ dồ tuần tự quy trình xử lý yêu cầu của người dùng |

#### Xây dựng từng module

##### Tạo hệ thống Embedding với Rotation

Module RotatingEmbeddings được thiết kế để giải quyết vấn đề giới hạn hạn ngạch API khi thực hiện embedding số lượng lớn văn bản y khoa. Hệ thống tích hợp cơ chế luân phiên tự động giữa nhiều API key, đảm bảo tính liên tục của dịch vụ vector hóa dữ liệu.

Kiến Trúc Lớp RotatingEmbeddings

Lớp RotatingEmbeddings kế thừa nguyên lý embedding từ thư viện LangChain và bổ sung cơ chế quản lý đa API key:

Thành phần dữ liệu chính:

* api\_keys: Danh sách các khoá API dự phòng
* model: Mô hình embedding sử dụng (models/gemini-embedding-001)
* index: Chỉ số API key hiện tại đang hoạt động
* emb: Thể hiện embedding hiện tại

**Phương Thức Khởi Tạo Embedding**

Phương thức \_make\_embedding đóng vai trò trung tâm trong việc khởi tạo kết nối đến Google Generative AI Embedding service:

def \_make\_embedding(self):

return GoogleGenerativeAIEmbeddings(

google\_api\_key=self.api\_keys[self.index],

model=self.model

)

Phương thức này tạo và trả về một instance của googleGenerativeAIEmbeddings với API key hiện tại được chọn từ danh sách api\_keys. Đây là cơ chế cốt lõi cho phép hệ thống động chuyển đổi giữa các API key khác nhau.

**Cơ Chế Luân Phiên API Key**

Hệ thống triển khai thuật toán Round-Robin để phân phối tải:

def \_rotate\_key(self):

self.index = (self.index + 1) % len(self.api\_keys)

Khi phát hiện lỗi hạn ngạch (HTTP 429), hệ thống tự động chuyển sang API key tiếp theo trong danh sách, đảm bảo không gián đoạn quá trình xử lý.

**Phương Thức Embedding**

embed\_query và embed\_documents là hai phương thức cốt lõi của thư viện LangChain để tương tác với Embedding Model. Cả hai phương thức đều thực hiện chuyển đổi văn bản thành vector số học (embedding), nhưng được thiết kế cho các mục đích và ngữ cảnh sử dụng khác biệt trong kiến trúc Truy xuất Tăng cường (Retrieval-Augmented Generation - RAG) và hệ thống tìm kiếm ngữ nghĩa.Hàm embed\_query được sử dụng để nhúng câu hỏi, truy vấn hoặc một đoạn văn bản ngắn do người dùng nhập vào.

Phương Thức embed\_query dùng để nhúng câu hỏi, truy vấn hoặc đoạn văn bản ngắn từ người dùng.

def embed\_query(self, text):

return self.emb.embed\_query(text)

Vector kết quả được sử dụng để tính toán độ tương đồng ngữ nghĩa với các vector tài liệu đã được lưu trữ trước trong cơ sở dữ liệu vector (FAISS, ChromaDB, Pinecone). Trong hệ thống chatbot y khoa, phương thức này chuyển đổi triệu chứng hoặc câu hỏi của bệnh nhân thành vector để tìm kiếm thông tin liên quan.

Phương Thức embed\_documents dùng để nhúng hàng loạt các đoạn tài liệu lớn hoặc văn bản đã được chia nhỏ (chunks).

def embed\_documents(self, texts):

return self.emb.embed\_documents(texts)

Các vector kết quả được lưu trữ trong cơ sở dữ liệu vector, tạo thành kho tri thức (knowledge base) để phục vụ cho quá trình tìm kiếm ngữ nghĩa sau này. Trong bối cảnh y khoa, phương thức này được sử dụng để vector hóa các tài liệu chuyên môn, hướng dẫn điều trị, và nghiên cứu lâm sàng.

##### Xây dựng Vector Database với FAISS

**Vector Embedding**

Vector embedding là biểu diễn số hóa của dữ liệu (văn bản, hình ảnh, âm thanh,…) trong không gian nhiều chiều. Các mô hình như Word2Vec, BERT, Sentence-BERT, CLIP,… thường được sử dụng để tạo embedding.

|  |
| --- |
| IMG_256 |
| Hình 2.18: Dữ liệu sau khi được chuyển thành các vector embedding. |

Dữ liệu đầu vào được mã hóa thành các vectơ có độ dài cố định trong một không gian chiều cao, nơi các mục tương tự được đặt gần nhau và các mục khác nhau cách xa nhau hơn.

Sự sắp xếp không gian này cho phép đo lường sự tương đồng về mặt toán học cho phép các ứng dụng như tìm kiếm, đề xuất và phân loại hoạt động dựa trên ý nghĩa thay vì kết quả khớp chính xác.

Tính tương đồng giữa các vector thường được đánh giá bằng:

* Khoảng cách Euclidean (L2 distance)
* Tích vô hướng (dot product)
* Cosine similarity

**FAISS và Bài toán Similarity Search**

FAISS (Facebook AI Similarity Search) là một thư viện mã nguồn mở được phát triển bởi Meta AI, thiết kế chuyên biệt cho các tác vụ tìm kiếm tương đồng (similarity search) và phân cụm (clustering) trên các vector đặc (dense vectors). Thư viện được xây dựng bằng C++ nhằm tối ưu hiệu năng, đồng thời cung cấp các bộ wrapper đầy đủ cho Python, giúp quá trình tích hợp với các ứng dụng máy học trở nên thuận tiện hơn.

Đặc điểm nổi bật của FAISS là khả năng xử lý các tập dữ liệu có kích thước rất lớn, lên tới hàng triệu hoặc hàng tỷ vector, thậm chí cả những bộ dữ liệu không thể chứa hoàn toàn trong RAM. FAISS cũng hỗ trợ triển khai nhiều thuật toán trên GPU, nhờ đó tăng tốc đáng kể tốc độ tìm kiếm so với các giải pháp chạy hoàn toàn trên CPU.

Bài toán tìm kiếm sự tương đồng giữa một vector truy vấn và một tập hợp vector trong cơ sở dữ liệu.

Cho một tập dữ liệu gồm các vector:

|  |
| --- |
|  |

và một vector truy vấn mới:

|  |
| --- |
|  |

Mục tiêu của bài toán similarity search là tìm vector trong tập dữ liệu có độ tương đồng cao nhất hoặc khoảng cách nhỏ nhất tới vector truy vấn. Dạng bài toán phổ biến nhất là Nearest Neighbor Search, được mô tả bằng công thức:

|  |
| --- |
|  |

trong đó || **.** || thường là khoảng cách Euclidean (L2).  
 Có hai dạng chính:

FAISS sử dụng ANN để đạt tốc độ rất nhanh mà vẫn đảm bảo độ chính xác cao. ANN không tìm kiếm chính xác tuyệt đối nhưng tối ưu đáng kể thời gian truy vấn.

|  |
| --- |
|  |
| Hình 2.19: Workflow của tạo Vector Database với FAISS |

Chuẩn bị dữ liệu:  
 Dữ liệu ban đầu được crawl từ trang tin tức y khoa Youmed dùng để làm docs hỗ trợ cho việc tạo database.

|  |
| --- |
|  |
| |  | | --- | | Hình 2.20: Dữ liệu được sử dụng để tạo Vector Database | |

Dữ liệu sau đó sẽ được sử dụng để tạo thành các vector database thông qua FAISS và gemini-embedding-001 đã được giới thiệu trước đó để tạo thành các index

|  |
| --- |
|  |
| Hình 2.21: Mã nguồn cập nhật động (incremental update) cho một Vector Database sử dụng FAISS. |

Vector Database sau khi được tạo ra sẽ được tăng cường liên tục để mở rộng phạm vi và cải thiện độ chính xác của phản hồi.

##### Xây dựng Agent System với LangChain

Hệ thống được xây dựng theo mô hình Hierarchical Multi-Agent với cấu trúc phân cấp rõ ràng:

|  |
| --- |
|  |
|  |

**Agent Service - Quản lý vòng đời Agent:**

class AgentService:

    """Service để quản lý và tạo các agent instances"""

    def \_\_init\_\_(self, config: dict):

        self.config = config

        self.\_root\_agent = None

        self.\_extracted\_info\_agent = None

        self.\_search\_agent = None

Đặc điểm thiết kế:

* Lazy Loading Pattern: Các agent chỉ được khởi tạo khi thực sự cần
* Centralized Management: Quản lý tập trung tất cả agent instances
* Reset Capability: Cho phép reset toàn bộ agent khi có thay đổi cấu hình

**Root Agent**

def create\_root\_agent(config: dict) -> Agent:

    extracted\_info\_agent = create\_extracted\_info\_agent(config)

    return Agent(

        name="TraCuuYKhoaAgent",

        model=config["genmini"]["model"],

        description=(

            "Trợ lý y khoa thông minh và thân thiện, hỗ trợ người dùng tra cứu, tìm kiếm "

            "thông tin y tế và sức khỏe chính xác, dễ hiểu."

        ),

        sub\_agents=[extracted\_info\_agent],

        instruction="""

        Bạn là \*\*Y Khoa Trí Tuệ\*\* – một trợ lý y tế thông minh, tận tâm và đáng tin cậy.

        Tính cách:

        - Giải thích thông tin y khoa một cách \*\*chính xác, dễ hiểu\*\* cho mọi đối tượng.

        - Luôn giữ giọng điệu \*\*thân thiện, bình tĩnh, hỗ trợ\*\*.

        - Không dùng thuật ngữ chuyên môn quá phức tạp nếu không cần thiết; nếu bắt buộc dùng, hãy kèm giải thích ngắn gọn.

        - Không đưa ra lời khuyên y khoa thay cho bác sĩ; chỉ cung cấp thông tin tham khảo và khuyến khích người dùng hỏi ý kiến chuyên gia.

        Ngôn ngữ:

        - Chỉ sử dụng tiếng Việt.

        - Văn phong tự nhiên, dễ đọc, tránh từ ngữ gây hoang mang hoặc khó hiểu.

        <INSTRUCTION>

        ## Nhiệm vụ chính

        - Hỗ trợ người dùng tra cứu thông tin y khoa, triệu chứng, bệnh lý, thuốc men, xét nghiệm, phương pháp điều trị liên quan tới bệnh về da liễu.

        - Tìm kiếm và tóm tắt thông tin từ nguồn đáng tin cậy (Tổ chức Y tế Thế giới, Bộ Y tế Việt Nam, CDC, PubMed...).

        - Cung cấp thông tin tham khảo, \*\*không tự ý kê đơn thuốc\*\*.

        ## Kỹ năng hỗ trợ

        1. \*\*Tra cứu trực tuyến\*\* – tìm kiếm thông tin y tế mới nhất trên Internet về bệnh da liễu.

        2. \*\*Phân tích triệu chứng\*\* – giúp người dùng hiểu các nguyên nhân có thể liên quan.

        3. \*\*Giải thích thuật ngữ y khoa\*\* – chuyển đổi thuật ngữ khó sang ngôn ngữ dễ hiểu.

        ## Cách xử lý tình huống

        - Nếu người dùng chỉ nêu triệu chứng chung chung → hỏi thêm thông tin cần thiết (tuổi, giới tính, thời gian diễn ra, mức độ).

        - Nếu thông tin người dùng yêu cầu quá rộng → đề xuất thu hẹp chủ đề.

        - Nếu không tìm thấy thông tin phù hợp → trả lời:

          \*\*"Mình chưa tìm thấy thông tin phù hợp, bạn có thể mô tả chi tiết hơn không?"\*\*

        - Luôn khuyến cáo:

          \*\*"Thông tin này chỉ mang tính tham khảo, bạn nên trao đổi với bác sĩ hoặc chuyên gia y tế để được tư vấn chính xác."\*\*

        ## Các công cụ bạn có thể sử dụng

        - Bạn có thể gọi Extracted\_Info\_Agent: là một Agent để tìm kiếm và trích xuất thông tin y khoa từ các trang thông tin y tế đáng tin cậy.

        <INSTRUCTION>

        """,

    )

Đặc điểm:

* Persona-driven: Có tính cách rõ ràng ("Y Khoa Trí Tuệ")
* Vietnamese-focused: Chỉ sử dụng tiếng Việt
* Medical Ethics: Tuân thủ nguyên tắc y đức - không tự ý kê đơn, chỉ cung cấp thông tin tham khảo

**Extracted Info Agent - Agent Trích xuất Thông tin**

Chức năng: Chuyên xử lý trích xuất thông tin y khoa từ nhiều nguồn.

def create\_extracted\_info\_agent(config: dict) -> Agent:

    search\_agent = create\_search\_agent(config)

    search\_database = DatabaseSearchTool(config)

    return Agent(

        model=config["genmini"]["model"],

        name="Extracted\_Info\_Agent",

        output\_key="extracted\_results",

        instruction="""

# Công việc của bạn là trích xuất thông tin quan trọng từ danh sách các trang thông tin y tế.

# Các bước thực hiện (hãy thông báo cho người dùng bạn đang thực hiện bước nào, nhưng không tiết lộ chi tiết kỹ thuật):

1. Gọi công cụ search\_database(query="<text>").

2. Gọi công cụ google\_search(request=<text>) với truy vấn của người dùng để tìm kiếm thông tin y tế trên internet.

4. Đọc nội dung của các trang y tế  và thông tin y tế trên internet vừa mới tìm kiếm. Sau đó trích xuất thông tin chi tiết liên quan đến truy vấn.

5. Trả về thông tin đã trích xuất bằng tiếng việt dưới định dạng JSON.:

```json

{

  "extracted\_results": ""

}

```

        """,

        tools=[search\_database, AgentTool(agent=search\_agent)],

    )

Luồng xử lý:

* Search Database: Tìm kiếm trong cơ sở dữ liệu nội bộ
* Google Search: Tìm kiếm thông tin mới nhất trên internet
* Extract Information: Trích xuất thông tin chi tiết
* Format Output: Trả về dạng JSON chuẩn hóa

**Search Agent - Agent Tìm kiếm Chuyên dụng**

Chức năng: Chuyên xử lý tìm kiếm thông tin

def create\_search\_agent(config: dict) -> Agent:

    return Agent(

        model=config["genmini"]["model"],

        name="SearchAgent",

        instruction="""

        You're a spealist in Google Search

        """,

        tools=[

            google\_search,

        ],

    )

**Quy trình xử lý truy vấn**

|  |
| --- |
|  |
|  |

### Thiết kế giao diện

# TRIỂN KHAI VÀ KIỂM THỬ HỆ THỐNG

## Môi trường triển khai thử nghiệm

## Dữ liệu sử dụng cho kiểm thử

## Kịch bản kiểm thử

## Công nghệ sử dụng kiểm thử

## Đánh giá kết quả

KẾT LUẬN

Trong bối cảnh công nghệ thông tin và trí tuệ nhân tạo ngày càng phát triển mạnh mẽ, việc ứng dụng AI vào lĩnh vực y khoa không chỉ mang lại hiệu quả quản lý vượt trội mà còn góp phần quan trọng vào việc xây dựng môi trường giao thông thông minh và an toàn hơn. Đề tài "Nghiên cứu và xây dụng website hỗ trợ chẩn đoán bệnh da liễu dựa trên triệu chứng" đã bước đầu khẳng định tính khả thi của việc tích hợp AI và thị giác máy tính trong việc giám sát và xử lý vi phạm giao thông. Hệ thống được thiết kế giúp nhận diện chính xác tín hiệu đèn và hành vi vi phạm, từ đó giảm thiểu sự can thiệp của con người và tối ưu hóa quá trình giám sát.

Dưới sự hỗ trợ tận tình từ giảng viên giảng dạy **Đại uý, TS. Tống Anh Tuấn**, chúng em đã hoàn thành bài báo cáo chuyên đề này. Tuy nhiên, do giới hạn về thời gian, kinh nghiệm và nguồn lực, nghiên cứu vẫn còn một số mặt cần khắc phục, như mở rộng dữ liệu huấn luyện, cải thiện độ chính xác trong các tình huống giao thông phức tạp và tối ưu hóa hiệu suất hệ thống. Trong thời gian tới, nhóm định hướng sẽ tiếp tục hoàn thiện hệ thống, đồng thời mở rộng phạm vi ứng dụng, góp phần tạo ra những giá trị thực tiễn cao hơn.

Cuối cùng, chúng em xin gửi lời cảm ơn sâu sắc đến các thầy cô trong khoa, đặc biệt là giảng viên giảng dạy **Đại uý, TS. Tống Anh Tuấn** đã luôn đồng hành, định hướng và tạo điều kiện thuận lợi để chúng em hoàn thành báo cáo này. Những kiến thức, kinh nghiệm quý báu được truyền đạt không chỉ giúp chúng em hoàn thành báo cáo mà còn là hành trang quan trọng cho những bước đi tiếp theo trên hành trình học tập và nghiên cứu. Chúng em rất mong nhận được sự đóng góp ý kiến từ thầy cô để hoàn thiện hơn nữa chuyên đề của nhóm.

TÀI LIỆU THAM KHẢO

[1] MaryamBoneh. (n.d.). *GitHub - MaryamBoneh/Vehicle-Detection: Vehicle detection using deep learning and YOLO algorithm*. GitHub. <https://github.com/MaryamBoneh/Vehicle-Detection>

[2] *Comparison of YOLO different versions developed by Ultralytics in terms. . .* (n.d.). ResearchGate. <https://www.researchgate.net/figure/Comparison-of-YOLO-different-versions-developed-by-Ultralytics-in-terms-of-mAP_fig2_380729550>

[3] *Algorithm principles and implementation with YOLOv8 — MMYOLO 0.6.0 documentation*. (n.d.). <https://mmyolo.readthedocs.io/en/latest/recommended_topics/algorithm_descriptions/yolov8_description.html>

[4] HumanSignal. (n.d.). *GitHub - HumanSignal/labelImg: LabelImg is now part of the Label Studio community. The popular image annotation tool created by Tzutalin is no longer actively being developed, but you can check out Label Studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data.* GitHub. <https://github.com/HumanSignal/labelImg>

[5] Mì AI. (2024, June 23). *Thư viện Mì AI - sách, dataset miễn phí cho các bạn học AI - Mì AI*. <https://www.miai.vn/thu-vien-mi-ai/>

[6] Lê Quốc Huy (2021), "*Thuật toán học sâu CNN và ứng dụng trong nhận diện khuôn mặt*", Đại học Bách Khoa Hà Nội, tập 58, số 2, trang 12-20

* **CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**
* **2.1. Tổng quan và phân tích hệ thống**
* **2.1.1. Tổng quan về hệ thống**

Hệ thống được xây dựng nhằm phục vụ công tác **hậu kiểm và xử lý vi phạm giao thông** thông qua video giám sát tại các nút đèn tín hiệu. Người dùng là **cán bộ nghiệp vụ** sẽ tải lên video do camera giao thông ghi lại, hệ thống sẽ tự động xử lý, phát hiện hành vi **vượt đèn đỏ**, **trích xuất dữ liệu vi phạm** và hiển thị trên website.  
Hệ thống gồm 3 thành phần chính:

* **Mô-đun trí tuệ nhân tạo (AI)**: xử lý video, phát hiện vi phạm và nhận diện phương tiện.
* **Nền tảng website**: giao diện tương tác với người dùng.
* **Cơ sở dữ liệu**: lưu trữ thông tin vi phạm và dữ liệu liên quan.
* **2.1.2. Phân tích chức năng hệ thống**

| **STT** | **Chức năng chính** | **Mô tả** |
| --- | --- | --- |
| 1 | Tải video | Cán bộ tải video ghi hình từ camera giám sát lên hệ thống |
| 2 | Phân tích vi phạm | AI phân tích trạng thái đèn + hành vi phương tiện |
| 3 | Xác định vi phạm vượt đèn đỏ | Đối chiếu thời điểm và vị trí để phát hiện hành vi vi phạm |
| 4 | Trích xuất dữ liệu vi phạm | Gồm hình ảnh, loại xe, biển số (nếu có), thời gian, đoạn video |
| 5 | Hiển thị và lưu trữ kết quả | Kết quả được hiển thị trên web và lưu vào cơ sở dữ liệu |
| 6 | Tìm kiếm, tra cứu, xuất báo cáo | Hỗ trợ lọc theo thời gian, biển số, loại phương tiện… |

* **2.1.3. Thiết kế cơ sở dữ liệu**

Hệ thống sử dụng cơ sở dữ liệu quan hệ (PostgreSQL hoặc tương đương) để lưu trữ các thông tin chính như sau:

**Các bảng dữ liệu chính:**

* videos: thông tin các video đã tải lên.
* violations: danh sách các vi phạm được phát hiện từ mỗi video.
* vehicles: thông tin phương tiện vi phạm (nếu trích xuất được).
* users: tài khoản cán bộ sử dụng hệ thống (nếu phân quyền nhiều cấp).

**Sơ đồ quan hệ cơ bản:**

pgsql

CopyEdit

videos (video\_id, name, path, upload\_time)

|

| 1-n

v

violations (violation\_id, video\_id, vehicle\_id, timestamp, image\_path, clip\_path, light\_status, location)

|

| n-1

v

vehicles (vehicle\_id, plate\_number, vehicle\_type)

* **2.2. Xây dựng mô-đun AI phát hiện vi phạm**
* **2.2.1. Quy trình xây dựng mô hình AI**

Quá trình xây dựng mô hình trí tuệ nhân tạo để phát hiện hành vi vượt đèn đỏ bao gồm các bước sau:

1. **Thu thập và xử lý dữ liệu**
   * Tập hợp video giao thông từ các camera tại nút đèn tín hiệu.
   * Tách khung hình và gán nhãn cho các đối tượng: phương tiện, đèn giao thông, vạch dừng.
2. **Huấn luyện mô hình phát hiện đối tượng**
   * Sử dụng mô hình **YOLOv5 hoặc YOLOv8** để phát hiện:
     + Đèn giao thông và trạng thái (đỏ, xanh, vàng).
     + Phương tiện: xe máy, ô tô, xe tải…
   * Áp dụng mô hình **DeepSORT** để theo dõi chuyển động của từng phương tiện qua các khung hình.
3. **Xây dựng logic phát hiện hành vi vượt đèn đỏ**
   * Theo dõi trạng thái của đèn tại từng thời điểm.
   * Gắn ID cho mỗi phương tiện và kiểm tra vị trí so với vạch dừng khi đèn đỏ đang bật.
   * Ghi nhận vi phạm nếu phương tiện vượt qua vạch trong lúc đèn đỏ.
4. **Nhận dạng biển số xe (nếu có)**
   * Cắt ảnh vùng biển số xe từ khung hình vi phạm.
   * Sử dụng mô hình OCR như **EasyOCR** hoặc **PaddleOCR** để đọc biển số.
5. **Tối ưu và đóng gói mô hình**
   * Tối ưu tốc độ xử lý và dung lượng mô hình.
   * Đóng gói mô hình dưới dạng dịch vụ API để tích hợp với hệ thống website.

* **2.2.2. Kết quả đầu ra của AI**

Sau khi xử lý video, mô hình AI xuất ra:

* Ảnh chụp tại thời điểm vi phạm.
* Thời điểm vi phạm (timestamp).
* Loại phương tiện, biển số xe (nếu có).
* Đoạn video clip (3–5 giây) làm bằng chứng.
* Tọa độ vi phạm (nếu tích hợp GPS/video metadata).
* **2.3. Thiết kế và triển khai nền tảng website**
* **2.3.1. Vai trò của website trong hệ thống**

Website đóng vai trò là **giao diện nghiệp vụ** giữa người dùng và hệ thống AI. Thông qua đó, cán bộ chức năng có thể thao tác nhanh chóng, hiệu quả mà không cần tiếp xúc với các mô-đun kỹ thuật chuyên sâu.

* **2.3.2. Chức năng chính của website**

| **Chức năng** | **Mô tả** |
| --- | --- |
| Giao diện tải video | Hỗ trợ kéo-thả hoặc chọn tệp video để phân tích |
| Xem kết quả | Hiển thị danh sách vi phạm được phát hiện từ video đã xử lý |
| Tra cứu | Tìm kiếm theo biển số, thời gian, loại xe, vị trí |
| Chi tiết vi phạm | Xem hình ảnh, đoạn video, thông tin chi tiết về từng vi phạm |
| Xuất báo cáo | Tải dữ liệu ra Excel/PDF phục vụ công tác lập biên bản, báo cáo |

* **2.3.3. Kiến trúc triển khai web**

Hệ thống được xây dựng theo mô hình 3 lớp:

1. **Frontend**: Giao diện website người dùng (ReactJS/VueJS).
2. **Backend**: Xử lý yêu cầu người dùng, gọi AI, lưu trữ kết quả (Flask/FastAPI).
3. **Cơ sở dữ liệu và lưu trữ**: PostgreSQL + Hệ thống lưu trữ file ảnh/video.

* **2.3.4. Quy trình hoạt động của website**

plaintext

CopyEdit

Người dùng → Tải video → Website gửi đến mô-đun AI → Nhận kết quả → Hiển thị kết quả trên Web → Lưu vào CSDL

Nếu em cần thêm:  
✅ Sơ đồ Use Case cho Website  
✅ Wireframe giao diện website  
✅ Các đoạn mô tả cho báo cáo chính thức

Anh có thể tiếp tục hỗ trợ chi tiết! Hoặc nếu em muốn trích dẫn các tài liệu học thuật về YOLO, OCR, giao thông thông minh... thì anh cũng có thể bổ sung nguồn luôn nha.