

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
—oo—



**BÁO CÁO HỌC PHẦN**  
CSDL Web và hệ thống thông tin MAT3385

Đề tài: Hệ thống Website quản lý bệnh viện

Nhóm thực hiện  
Nguyễn Việt Phúc - 23001547  
Nguyễn Khắc Huy - 23001525

Giảng viên hướng dẫn  
PGS.TS Vũ Tiến Dũng  
GV Phạm Duy Phương

Học kỳ 1, Năm học 2025-2026

**HÀ NỘI - 2025**

# Thông tin Dự án

**Học phần:** MAT3385 - CSDL Web và hệ thống thông tin

**Học kỳ:** Học kỳ 1, Năm học 2025-2026

**Trường:** VNU-HUS (ĐHQGHN - Trường ĐH Khoa học Tự nhiên)

**Tên dự án:** Hệ thống Website quản lý bệnh viện

**Website:** Bệnh viện Khoa học tự nhiên

**Slide thuyết trình:** Liên kết Slide

**Video demo:** Liên kết Video

**Kho GitHub:** Liên kết Github

## Thành viên nhóm

Họ tên	Mã sinh viên	Tên GitHub	Đóng góp
Nguyễn Việt Phúc	23001547	Dr-Vphuc	Thiết kế Database, Backend
Nguyễn Khắc Huy	23001525	khachuy171	Thiết kế giao diện, Frontend, Devops

# Mục lục

<b>1 Giới thiệu</b>	<b>6</b>
1.1 Lời nói đầu . . . . .	6
1.2 Tóm tắt dự án . . . . .	7
1.3 Bài toán đặt ra . . . . .	8
1.3.1 Hiện trạng và thách thức . . . . .	8
1.3.2 Yêu cầu về tính sẵn sàng trên Cloud . . . . .	9
1.4 Mục tiêu đề tài . . . . .	10
1.5 Kết luận chương 1 . . . . .	10
<b>2 Kiến trúc Hệ thống và Thiết kế CSDL</b>	<b>12</b>
2.1 Kiến trúc tổng quát (System Pipeline) . . . . .	12
2.2 Thiết kế Cơ sở dữ liệu . . . . .	13
2.2.1 Sơ đồ thực thể mối quan hệ (ERD) . . . . .	13
2.2.2 Mô tả các bảng dữ liệu và ràng buộc . . . . .	13
2.3 Cấu trúc ORM (Object-Relational Mapping) . . . . .	14
2.3.1 Ánh xạ Class sang Bảng . . . . .	15
2.3.2 Bảo mật dữ liệu với Werkzeug Security . . . . .	15
2.3.3 Ưu điểm của cấu trúc ORM đã thiết kế . . . . .	16
2.4 Giao diện với Jinja2 . . . . .	16
2.5 Kết luận chương 2 . . . . .	16
<b>3 Triển khai và Cấu hình</b>	<b>18</b>
3.1 Môi trường phát triển . . . . .	18
3.1.1 Cài đặt thư viện . . . . .	18
3.2 Cấu hình kết nối MySQL . . . . .	18
3.2.1 Quản lý biến môi trường . . . . .	19
3.3 Triển khai trên AWS EC2 . . . . .	19
3.3.1 Cấu hình Instance và Security Groups . . . . .	19
3.3.2 Thiết lập Web Server (Gunicorn và Nginx) . . . . .	20
3.3.3 Quản lý và Di chuyển Dữ liệu (Database Migration) . . . . .	20
3.4 Kết luận chương 3 . . . . .	21

<b>4 Kết quả thực nghiệm và đánh giá</b>	<b>22</b>
4.1 Các kết quả đạt được . . . . .	22
4.1.1 Các tính năng . . . . .	22
4.1.2 Demo giao diện . . . . .	23
4.2 Đánh giá hiệu năng trên máy chủ EC2 . . . . .	24
4.3 Các hạn chế . . . . .	24
<b>5 Kết luận và hướng phát triển</b>	<b>31</b>
5.1 Kết luận . . . . .	31
5.2 Hướng phát triển . . . . .	31
5.2.1 Ngắn hạn . . . . .	31
5.2.2 Dài hạn . . . . .	31
5.3 Kết luận cuối cùng . . . . .	32
<b>Tài liệu tham khảo</b>	<b>33</b>
<b>A Hướng dẫn Vận hành và Bảo trì</b>	<b>34</b>
A.1 Tài liệu cài đặt và hướng dẫn sử dụng . . . . .	34
A.1.1 Yêu cầu hệ thống . . . . .	34
A.1.2 Cấu trúc mã nguồn . . . . .	34
A.1.3 Hướng dẫn cài đặt và triển khai bằng Docker . . . . .	35
A.1.4 Hướng dẫn sử dụng hệ thống . . . . .	36
A.1.5 Khả năng mở rộng và tái triển khai . . . . .	36
A.2 Quy trình Backup và khôi phục dữ liệu . . . . .	36
A.2.1 Truy cập Server . . . . .	37
A.2.2 Sao lưu dữ liệu (Backup) . . . . .	37
A.2.3 Khôi phục dữ liệu (Restore/Import) . . . . .	37
A.2.4 Tự động hóa (Automation) . . . . .	38
<b>B Danh sách API (API Endpoints)</b>	<b>39</b>
B.1 Authentication (Dùng chung) . . . . .	39
B.2 Admin API (Quản trị viên) . . . . .	39
B.3 Doctor API (Bác sĩ) . . . . .	40
B.4 Patient API (Bệnh nhân) . . . . .	40

# Danh sách hình vẽ

2.1	Sơ đồ luồng xử lý (System Pipeline) của hệ thống . . . . .	12
2.2	Sơ đồ thực thể mối quan hệ (ERD) . . . . .	13
2.3	Cơ sở dữ liệu quan hệ chuyển từ sơ đồ thực thể - MySQL . . . . .	14
4.1	Giao diện Dashboard phân tích tổng thể của role Admin . . . . .	23
4.2	Giao diện trang bệnh nhân của role bác sĩ . . . . .	23
4.3	Giao diện trang bác sĩ của role Admin . . . . .	24
4.4	Giao diện phân tích hiệu suất làm việc của bác sĩ . . . . .	25
4.5	Giao diện trang đơn thuốc của role bác sĩ . . . . .	26
4.6	Giao diện thêm đơn thuốc mới của role bác sĩ . . . . .	26
4.7	Giao diện trang kho thuốc của role Admin . . . . .	27
4.8	Giao diện tạo lô thuốc mới của role Admin . . . . .	27
4.9	Giao diện trang phòng bệnh của role Admin . . . . .	28
4.10	Giao diện danh sách bệnh nhân ở 1 phòng cụ thể của role Admin	28
4.11	Giao diện hồ sơ cá nhân của role bệnh nhân . . . . .	29
4.12	Giao diện theo dõi các đơn thuốc của bệnh nhân role Admin và bác sĩ . . . . .	29
4.13	Giao diện trang các đơn thuốc được kê cho tôi của role bệnh nhân	30

# Chương 1

## Giới thiệu

### 1.1 Lời nói đầu

Trong bối cảnh khoa học – công nghệ phát triển mạnh mẽ, đặc biệt là sự bùng nổ của công nghệ thông tin, việc ứng dụng các hệ thống phần mềm vào công tác quản lý đã và đang trở thành xu hướng tất yếu trong nhiều lĩnh vực của đời sống xã hội. Ngành y tế, với đặc thù liên quan trực tiếp đến sức khỏe và tính mạng con người, càng đòi hỏi các giải pháp quản lý chính xác, hiệu quả và an toàn hơn bao giờ hết. Trước yêu cầu đó, việc xây dựng một hệ thống **Quản lý Bệnh viện** hiện đại, đồng bộ và có tính thực tiễn cao là một nhiệm vụ vô cùng cần thiết.

Xuất phát từ nhu cầu thực tế trong công tác quản lý bệnh viện, nơi phải xử lý một khối lượng lớn dữ liệu liên quan đến bệnh nhân, bác sĩ, khoa phòng, hồ sơ bệnh án, quá trình khám chữa bệnh và thanh toán viện phí, nhóm chúng em đã lựa chọn đề tài “Xây dựng hệ thống Quản lý Bệnh viện” làm nội dung cho dự án này. Mục tiêu của dự án không chỉ dừng lại ở việc xây dựng một phần mềm đáp ứng các chức năng cơ bản, mà còn hướng tới việc tối ưu hóa quy trình quản lý, giảm thiểu sai sót thủ công, nâng cao hiệu quả làm việc và góp phần cải thiện chất lượng dịch vụ y tế.

Thông qua dự án, chúng em có cơ hội vận dụng và củng cố các kiến thức đã học về lập trình, cơ sở dữ liệu, phân tích và thiết kế hệ thống, cũng như tiếp cận quy trình phát triển một phần mềm quản lý hoàn chỉnh trong thực tế. Bên cạnh đó, dự án còn giúp chúng em rèn luyện tư duy phân tích vấn đề, khả năng làm việc độc lập và làm việc nhóm, cũng như kỹ năng giải quyết các tình huống phát sinh trong quá trình xây dựng hệ thống.

Mặc dù đã có nhiều cố gắng trong quá trình thực hiện, song do thời gian và kinh nghiệm thực tế còn hạn chế, dự án khó tránh khỏi những thiếu sót nhất định. Chúng em rất mong nhận được sự đóng góp ý kiến, nhận xét và hướng dẫn từ giảng viên để có thể hoàn thiện hơn trong các nghiên cứu và dự án tiếp theo.

Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của giảng viên

cùng sự hỗ trợ từ các tài liệu tham khảo đã giúp chúng em hoàn thành báo cáo này.

## 1.2 Tóm tắt dự án

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ trên toàn cầu, ngành y tế ngày càng chú trọng đến việc ứng dụng công nghệ thông tin nhằm nâng cao hiệu quả quản lý, tối ưu hóa quy trình vận hành và cải thiện chất lượng dịch vụ khám chữa bệnh. Việc quản lý dữ liệu y tế theo phương thức truyền thống không chỉ tốn nhiều thời gian, nhân lực mà còn tiềm ẩn nguy cơ sai sót, thất lạc thông tin và khó mở rộng trong tương lai. Do đó, xây dựng các hệ thống quản lý bệnh viện dựa trên nền tảng công nghệ hiện đại là một yêu cầu cấp thiết.

Xuất phát từ nhu cầu thực tiễn đó, dự án này tập trung xây dựng một hệ thống **Cơ sở dữ liệu Web quản lý bệnh viện**, cho phép quản lý tập trung và thống nhất các thông tin quan trọng như hồ sơ bệnh nhân, thông tin bác sĩ, khoa phòng, cũng như lịch sử khám chữa bệnh và điều trị. Hệ thống được thiết kế nhằm hỗ trợ công tác quản lý một cách khoa học, giúp giảm thiểu thao tác thủ công, tăng độ chính xác của dữ liệu và nâng cao hiệu suất làm việc của đội ngũ y tế.

Về mặt chức năng, hệ thống cho phép người dùng thực hiện các nghiệp vụ cơ bản như thêm, sửa, xóa và tra cứu thông tin bệnh nhân và bác sĩ; theo dõi quá trình khám chữa bệnh; lưu trữ lịch sử điều trị và hỗ trợ truy xuất dữ liệu nhanh chóng khi cần thiết. Bên cạnh đó, hệ thống được xây dựng theo mô hình ứng dụng Web, giúp người dùng có thể truy cập và sử dụng mọi lúc, mọi nơi thông qua trình duyệt, mà không phụ thuộc vào thiết bị hay vị trí địa lý.

Dự án được phát triển dựa trên các công nghệ mã nguồn mở phổ biến và các nền tảng hiện đại, đảm bảo tính linh hoạt, khả năng mở rộng và dễ dàng bảo trì trong tương lai, cụ thể:

- **Backend:** Sử dụng Framework **Flask** (Python) để xây dựng logic xử lý nghiệp vụ và các API phục vụ trao đổi dữ liệu giữa client và server. Flask với kiến trúc nhẹ, linh hoạt giúp hệ thống dễ dàng mở rộng và tùy biến theo nhu cầu thực tế.
- **Database:** Hệ quản trị cơ sở dữ liệu **MySQL** được sử dụng để lưu trữ dữ liệu có cấu trúc, đảm bảo tính toàn vẹn, nhất quán và độ tin cậy cao cho các thông tin y tế quan trọng.
- **ORM:** Thư viện **SQLAlchemy** đóng vai trò là cầu nối giữa ứng dụng và cơ sở dữ liệu, giúp ánh xạ dữ liệu quan hệ sang các đối tượng Python, từ đó đơn giản hóa thao tác truy vấn, giảm lỗi và nâng cao hiệu quả phát triển.
- **Frontend:** Hệ thống giao diện người dùng được xây dựng bằng **Jinja2**

**Template** kết hợp với HTML, CSS và Bootstrap, cho phép hiển thị nội dung động, thân thiện với người dùng và dễ dàng tùy chỉnh.

- **Deployment:** Toàn bộ hệ thống được triển khai thực tế trên hạ tầng điện toán đám mây **AWS EC2** (Amazon Elastic Compute Cloud), đảm bảo khả năng truy cập ổn định qua Internet, đồng thời tạo tiền đề cho việc mở rộng và nâng cấp hệ thống trong tương lai.

Thông qua dự án này, nhóm thực hiện không chỉ xây dựng một hệ thống quản lý bệnh viện mang tính ứng dụng thực tế, mà còn có cơ hội tiếp cận quy trình thiết kế, phát triển và triển khai một ứng dụng Web hoàn chỉnh, làm nền tảng cho việc nghiên cứu và phát triển các hệ thống quản lý y tế thông minh trong thời gian tới.

## 1.3 Bài toán đặt ra

### 1.3.1 Hiện trạng và thách thức

Trong thực tế, tại nhiều cơ sở y tế quy mô nhỏ, phòng khám tư nhân hoặc các bệnh viện vẫn đang sử dụng những hệ thống quản lý cũ, chưa được số hóa một cách đồng bộ và toàn diện. Công tác quản lý hồ sơ bệnh án, thông tin bệnh nhân và quá trình khám chữa bệnh phần lớn vẫn dựa vào các phương pháp thủ công hoặc các công cụ rời rạc, thiếu sự liên kết giữa các bộ phận. Điều này gây ra nhiều khó khăn trong quá trình vận hành, quản lý và khai thác dữ liệu y tế.

Cụ thể, các vấn đề và thách thức thường gặp có thể kể đến như sau:

- **Dữ liệu phân mảnh:** Thông tin bệnh nhân, bác sĩ và lịch sử khám chữa bệnh thường được lưu trữ rải rác ở nhiều nguồn khác nhau như file Excel, phần mềm đơn lẻ hoặc sổ sách giấy. Việc này khiến cho quá trình tra cứu thông tin mất nhiều thời gian, đồng thời gây khó khăn trong việc tổng hợp, thống kê và phân tích dữ liệu phục vụ công tác quản lý.
- **Tính toàn vẹn và nhất quán dữ liệu:** Do dữ liệu được nhập liệu thủ công và không có cơ chế kiểm soát chặt chẽ, nguy cơ trùng lặp hồ sơ bệnh nhân, sai sót thông tin cá nhân hoặc nhầm lẫn trong lịch sử điều trị là rất cao. Những sai lệch này có thể ảnh hưởng trực tiếp đến chất lượng khám chữa bệnh và công tác ra quyết định của đội ngũ y tế.
- **Khả năng truy cập và chia sẻ thông tin:** Các hệ thống quản lý cục bộ (localhost hoặc Intranet) chỉ cho phép truy cập trong phạm vi nội bộ, hạn chế khả năng làm việc từ xa của đội ngũ quản lý, bác sĩ và nhân viên y tế. Điều này đặc biệt bất tiện trong các tình huống cần tra cứu nhanh thông tin bệnh nhân hoặc phối hợp giữa nhiều bộ phận.

- **Khả năng mở rộng và bảo trì hệ thống:** Nhiều hệ thống cũ được xây dựng theo kiến trúc đóng, khó mở rộng và nâng cấp khi số lượng bệnh nhân tăng lên hoặc khi phát sinh các yêu cầu nghiệp vụ mới. Việc bảo trì, sửa lỗi và cập nhật hệ thống vì thế cũng gặp nhiều khó khăn và tốn kém chi phí.
- **Bảo mật và an toàn dữ liệu:** Dữ liệu y tế là loại dữ liệu nhạy cảm, tuy nhiên trong nhiều trường hợp, các biện pháp bảo mật chưa được chú trọng đúng mức. Việc phân quyền người dùng chưa rõ ràng hoặc thiếu các cơ chế kiểm soát truy cập làm gia tăng nguy cơ rò rỉ và mất mát thông tin.

Những hạn chế nêu trên đặt ra yêu cầu cần phải xây dựng một hệ thống quản lý bệnh viện dựa trên nền tảng Web, có khả năng quản lý dữ liệu tập trung, đảm bảo tính toàn vẹn và bảo mật thông tin, đồng thời hỗ trợ truy cập linh hoạt và mở rộng trong tương lai.

### 1.3.2 Yêu cầu về tính sẵn sàng trên Cloud

Để khắc phục những hạn chế của các hệ thống quản lý truyền thống và đáp ứng nhu cầu vận hành liên tục trong môi trường y tế, bài toán đặt ra không chỉ dừng lại ở việc lưu trữ dữ liệu hiệu quả mà còn phải đảm bảo **tính sẵn sàng cao (High Availability)** của toàn bộ hệ thống. Trong bối cảnh hoạt động khám chữa bệnh diễn ra liên tục, việc hệ thống bị gián đoạn, ngừng hoạt động hoặc không thể truy cập có thể gây ảnh hưởng nghiêm trọng đến công tác quản lý và chất lượng dịch vụ y tế.

Do đó, việc triển khai ứng dụng Web và cơ sở dữ liệu trên nền tảng điện toán đám mây, cụ thể là **Amazon Web Services (AWS)**, được xem là một giải pháp phù hợp và cần thiết. Nền tảng Cloud cho phép hệ thống vận hành ổn định, linh hoạt và có khả năng thích ứng với các yêu cầu thay đổi trong tương lai. Việc đưa hệ thống lên Cloud nhằm đáp ứng các yêu cầu sau:

- **Đảm bảo tính sẵn sàng và hoạt động liên tục:** Hệ thống cần được thiết kế để hoạt động ổn định 24/7, hạn chế tối đa thời gian gián đoạn dịch vụ. Việc triển khai trên hạ tầng Cloud giúp giảm thiểu rủi ro do sự cố phần cứng và cho phép nhanh chóng khôi phục hệ thống khi xảy ra lỗi.
- **Khả năng truy cập linh hoạt và an toàn:** Hệ thống phải cho phép đội ngũ quản lý, bác sĩ và nhân viên y tế truy cập dữ liệu mọi lúc, mọi nơi thông qua Internet, đồng thời đảm bảo các cơ chế bảo mật, phân quyền và kiểm soát truy cập nhằm bảo vệ dữ liệu y tế nhạy cảm.
- **Khả năng mở rộng trong tương lai:** Khi quy mô bệnh viện mở rộng, số lượng bệnh nhân và dữ liệu phát sinh ngày càng tăng, hệ thống cần có khả năng dễ dàng nâng cấp và mở rộng tài nguyên phần cứng như CPU, bộ nhớ và dung lượng lưu trữ mà không làm gián đoạn hoạt động hiện tại.

Việc lựa chọn triển khai hệ thống trên nền tảng Cloud không chỉ giúp đáp ứng các yêu cầu về tính sẵn sàng và hiệu năng, mà còn tạo tiền đề cho việc tích hợp thêm các dịch vụ nâng cao trong tương lai, như sao lưu dữ liệu tự động, giám sát hệ thống và tăng cường bảo mật thông tin.

## 1.4 Mục tiêu đề tài

Mục tiêu chính của đồ án là xây dựng hoàn thiện một quy trình (pipeline) phát triển web từ thiết kế cơ sở dữ liệu đến triển khai thực tế. Các tính năng và mục tiêu cụ thể bao gồm:

- **Quản lý dữ liệu (CRUD):** Thực hiện đầy đủ các chức năng Thêm (Create), Đọc (Read), Sửa (Update), và Xóa (Delete) đối với các thực thể chính như Bác sĩ, Bệnh nhân, Khoa khám.
- **Tương tác cơ sở dữ liệu qua ORM:** Xây dựng mô hình dữ liệu chặt chẽ, thiết lập các mối quan hệ (Foreign Keys) và truy vấn dữ liệu hiệu quả thông qua SQLAlchemy thay vì viết SQL thủ công.
- **Giao diện quản trị trực quan:** Cung cấp Dashboard hiển thị danh sách và thông tin chi tiết, hỗ trợ tìm kiếm và lọc dữ liệu.
- **Triển khai thành công trên AWS:** Cấu hình Web Server (Gunicorn/Nginx) trên hệ điều hành Ubuntu để chạy ứng dụng Flask bền bỉ, kết nối ổn định với MySQL Database.

## 1.5 Kết luận chương 1

Chương 1 đã trình bày một cách tổng quan về bối cảnh chuyển đổi số trong lĩnh vực y tế, từ đó làm rõ lý do hình thành và sự cần thiết của dự án xây dựng hệ thống quản lý bệnh viện trên nền tảng Web. Thông qua việc phân tích hiện trạng và các thách thức trong công tác quản lý hồ sơ y tế truyền thống, chương này đã chỉ ra những hạn chế tồn tại liên quan đến dữ liệu phân mảnh, tính toàn vẹn thông tin, khả năng truy cập và yêu cầu về bảo mật dữ liệu.

Bên cạnh đó, chương 1 cũng đã xác định rõ bài toán cần giải quyết và các mục tiêu cốt lõi mà hệ thống hướng tới, đặc biệt là yêu cầu về tính sẵn sàng cao và khả năng mở rộng khi triển khai trên nền tảng điện toán đám mây. Việc lựa chọn môi trường Cloud, cụ thể là AWS, không chỉ giúp đảm bảo hệ thống hoạt động ổn định và liên tục, mà còn tạo điều kiện thuận lợi cho việc quản lý, bảo trì và phát triển hệ thống trong tương lai.

Những nội dung được trình bày trong chương này đóng vai trò là nền tảng quan trọng, làm cơ sở cho các bước phân tích và thiết kế kỹ thuật ở các chương tiếp theo. Trên cơ sở các yêu cầu đã được xác định, chương tiếp theo của báo

cáo sẽ tập trung trình bày chi tiết về kiến trúc tổng thể của hệ thống, sơ đồ quan hệ thực thể (ERD), cũng như cách tổ chức và triển khai mã nguồn trong ứng dụng Flask nhằm hiện thực hóa các mục tiêu đã đề ra.

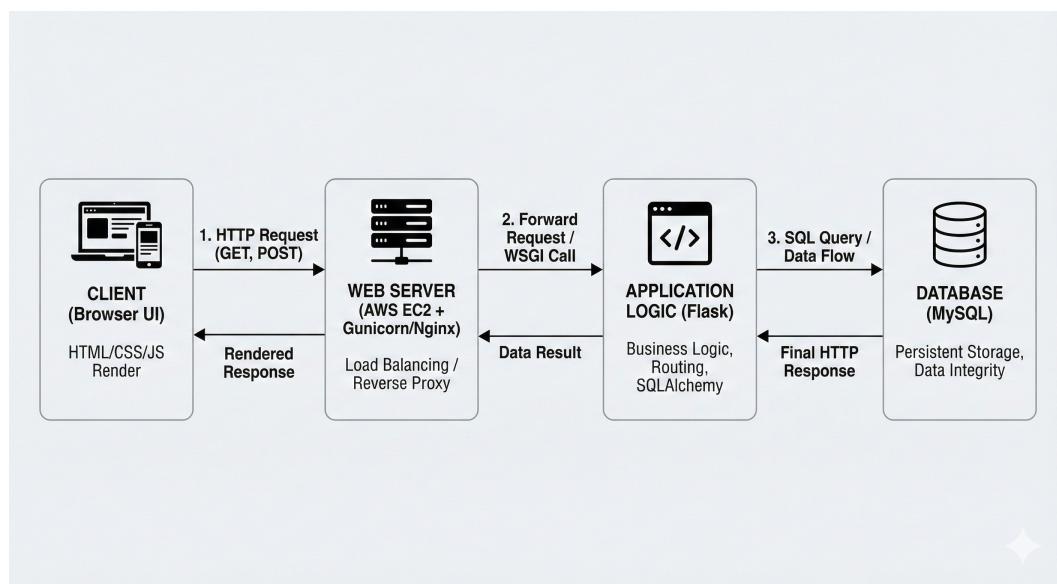
## Chương 2

# Kiến trúc Hệ thống và Thiết kế CSDL

Chương này trình bày chi tiết về phương pháp nghiên cứu và thiết kế kỹ thuật của hệ thống. Nội dung bao gồm mô hình kiến trúc tổng quát, thiết kế chi tiết cơ sở dữ liệu quan hệ và cách thức ánh xạ dữ liệu thông qua ORM trong ứng dụng Flask.

### 2.1 Kiến trúc tổng quát (System Pipeline)

Hệ thống được xây dựng dựa trên mô hình kiến trúc Client-Server (Khách-Chủ) tiêu chuẩn cho các ứng dụng web động. Luồng dữ liệu (Data flow) đi từ trình duyệt người dùng đến máy chủ ứng dụng và cuối cùng là cơ sở dữ liệu.



Hình 2.1: Sơ đồ luồng xử lý (System Pipeline) của hệ thống

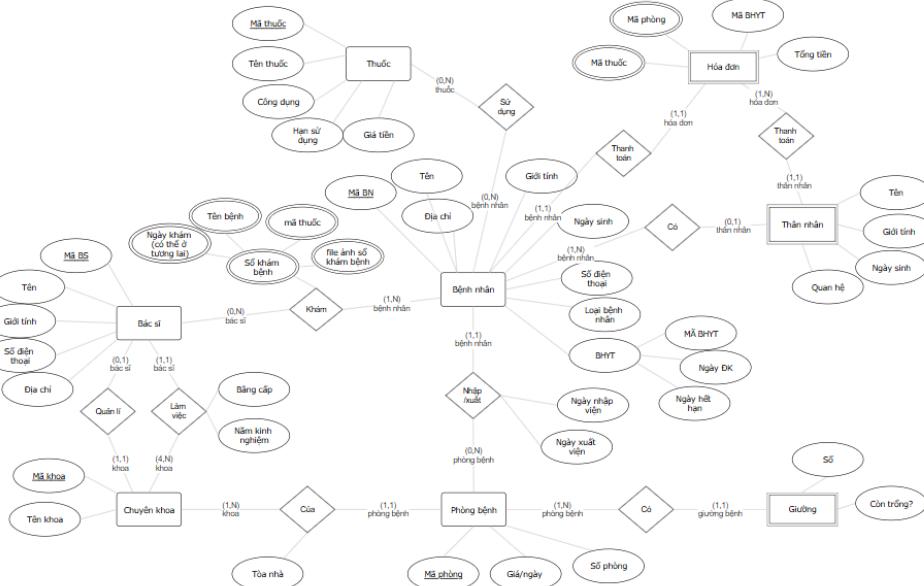
Các thành phần chính trong Pipeline bao gồm:

- **Client (Trình duyệt):** Gửi các HTTP Request (GET, POST) đến Server. Đây là nơi hiển thị giao diện người dùng (UI) được render từ HTML/CSS.
  - **Web Server (AWS EC2 + Gunicorn/Nginx):** Đóng vai trò tiếp nhận request, xử lý cân bằng tải và chuyển tiếp yêu cầu đến ứng dụng Flask.
  - **Application Logic (Flask):** Xử lý logic nghiệp vụ (Business Logic), định tuyến (Routing) và giao tiếp với cơ sở dữ liệu thông qua SQLAlchemy.
  - **Database (MySQL):** Nơi lưu trữ bền vững dữ liệu của bệnh viện, đảm bảo tính toàn vẹn và bảo mật.

## 2.2 Thiết kế Cơ sở dữ liệu

### 2.2.1 Sơ đồ thực thể mối quan hệ (ERD)

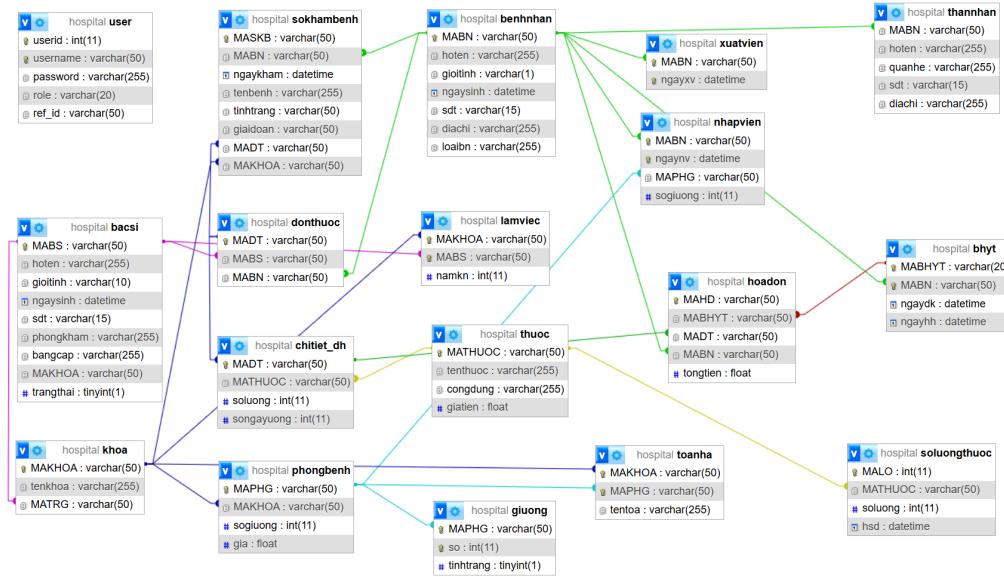
Cơ sở dữ liệu được thiết kế ở dạng chuẩn hóa (Normalization) để giảm thiểu dữ thừa dữ liệu. Dưới đây là sơ đồ ERD mô tả các thực thể và mối quan hệ trong hệ thống quản lý bệnh viện:



Hình 2.2: Sơ đồ thực thể mối quan hệ (ERD)

### 2.2.2 Mô tả các bảng dữ liệu và ràng buộc

Hệ thống bao gồm các bảng dữ liệu chính với các ràng buộc khóa chính (Primary Key - PK) và khóa ngoại (Foreign Key - FK). Một số bảng dữ liệu quan trọng được mô tả như sau:



Hình 2.3: Cơ sở dữ liệu quan hệ chuyển từ sơ đồ thực thể - MySQL

- **Bảng 'bacsi' (Bác sĩ):** Lưu trữ thông tin nhân sự y tế.
  - Các trường: *MABS (PK)*, *hoten*, *gioitinh*, *ngaysinh*, *sdt*, *phongkham*, *bangcap*, *MAKHOA*, *trangthai*.
  - Ràng buộc: *MABS* và *sdt* phải là duy nhất (Unique).
- **Bảng benhnhan (Bệnh nhân):** Lưu trữ hồ sơ hành chính của người bệnh.
  - Các trường: *MABN (PK)*, *hoten*, *ngaysinh*, *gioitinh*, *sdt*, *diachi*, *loainbn* (*Loại bệnh nhân*).
- **Bảng sokhambenh (Sổ khám bệnh):** Lưu trữ lịch sử khám chữa bệnh của bệnh nhân, với mỗi đơn thuốc được kê bởi bác sĩ điều trị tương ứng.
  - Các trường: *MASKB (PK)*, *MABN (FK)*, *MADT (FK)*, *MAKHOA (FK)*, *ngaykham*, *tenbenh*, *tinhtrang*, *giaidoan*.
  - Ràng buộc: *MABN* tham chiếu đến *benhnhan.MABN*, *MADT* tham chiếu đến *donthuoc.MADT*.

Và các bảng đặc trưng khác để lưu dữ liệu y tế như: bhyt, phongbenh, donthuoc, thanhhan,... Được liên kết một cách chặt chẽ, đảm bảo tính toàn vẹn và đồng bộ của hệ thống cơ sở dữ liệu.

## 2.3 Cấu trúc ORM (Object-Relational Mapping)

Thay vì sử dụng các câu lệnh SQL thuần (Raw SQL), dự án sử dụng **Flask-SQLAlchemy** làm công cụ ORM. Kỹ thuật này cho phép thao tác với cơ sở dữ

liệu thông qua các đối tượng (Objects) trong Python, đồng thời tích hợp các cơ chế tối ưu và bảo mật.

### 2.3.1 Ánh xạ Class sang Bảng

Mỗi bảng trong MySQL được ánh xạ thành một Class (Lớp) trong mã nguồn Python (trong file patient.py). Đặc biệt, dự án sử dụng cơ chế **Lazy Loading** trong các mối quan hệ (Relationships).

Ví dụ mô hình hóa mối quan hệ giữa Bệnh nhân và Đơn thuốc:

```

1 class Patient(db.Model):
2     __tablename__ = 'benhnhan'
3     MABN = db.Column(db.Integer, primary_key=True)
4     # ...
5     prescriptions = db.relationship('donthuoc',
6                                     backref='benhnhan',
7                                     lazy=True)

```

Việc thiết lập tham số `lazy=True` (hoặc `'select'`) mang lại hiệu quả lớn về hiệu năng:

- Khi truy vấn thông tin Bệnh nhân, hệ thống **không** tải ngay lập tức toàn bộ lịch sử khám bệnh của họ.
- Dữ liệu đơn thuốc chỉ được truy vấn từ Database khi và chỉ khi mã nguồn thực sự cần truy cập đến thuộc tính `patient.prescriptions`.
- Điều này giúp giảm tải băng thông và bộ nhớ khi danh sách dữ liệu liên quan trở nên quá lớn.

### 2.3.2 Bảo mật dữ liệu với Werkzeug Security

Vấn đề bảo mật thông tin đăng nhập (của Admin/Bác sĩ) được ưu tiên hàng đầu. Hệ thống không lưu mật khẩu dưới dạng văn bản thuần (plain text) mà sử dụng cơ chế băm (hashing).

Thư viện `werkzeug.security` được tích hợp ngay trong Model người dùng:

```

1 from werkzeug.security import generate_password_hash, check_password_hash
2
3 class User(db.Model):
4     # ...
5     password_hash = db.Column(db.String(256))
6
7     def set_password(self, password):
8         # Use default scrypt algorithm for high security
9         self.password_hash = generate_password_hash(password,
10                                                       method='scrypt')
11
12     def check_password(self, password):
13         return check_password_hash(self.password_hash, password)

```

- **Mã hóa một chiều:** Sử dụng hàm `generate_password_hash` với thuật toán `scrypt` (hoặc `pbkdf2`), đảm bảo rằng ngay cả khi cơ sở dữ liệu bị lộ, kẻ gian cũng không thể dịch ngược lại mật khẩu gốc.

- **Xác thực an toàn:** Khi đăng nhập, hệ thống sẽ băm mật khẩu người dùng nhập vào và so sánh với chuỗi băm trong Database thông qua `check_password_hash`.

### 2.3.3 Ưu điểm của cấu trúc ORM đã thiết kế

Việc kết hợp SQLAlchemy với các kỹ thuật trên mang lại lợi ích kép:

- **An toàn:** Ngăn chặn triệt để SQL Injection và lọt mật khẩu.
- **Hiệu năng:** Lazy Loading giúp hệ thống phản hồi nhanh hơn, tránh việc "N+1 Query" không cần thiết.
- **Dễ bảo trì:** Code Python tường minh, dễ dàng mở rộng logic nghiệp vụ.

## 2.4 Giao diện với Jinja2

Phần Frontend của ứng dụng được xây dựng dựa trên Jinja2 Templating Engine, tích hợp sẵn trong Flask.

- **Cấu trúc thư mục:** Các file HTML được đặt trong thư mục `templates/`, tài nguyên tĩnh (CSS, JS, Images) đặt trong `static/`.
- **Kế thừa giao diện (Template Inheritance):** Sử dụng file `base.html` chứa các thành phần chung (Navbar, Footer, Import thư viện). Các trang con (như `index.html`, `add_patient.html`) sẽ kế thừa lại khung này thông qua lệnh `{% extends "base.html"%}`.
- **Hiển thị dữ liệu động:** Dữ liệu truy vấn từ Database được truyền từ Flask sang HTML và hiển thị bằng cú pháp `{{ variable }}` hoặc vòng lặp `{% for item in list %}`.

## 2.5 Kết luận chương 2

Chương 2 đã trình bày chi tiết kiến trúc kỹ thuật của hệ thống quản lý bệnh viện, bao gồm mô hình triển khai tổng thể trên nền tảng Cloud cũng như cách tổ chức các thành phần chính của ứng dụng Web. Từ việc phân tích luồng dữ liệu tổng quát cho đến thiết kế cơ sở dữ liệu chi tiết trong MySQL, chương này đã làm rõ cách các thành phần trong hệ thống tương tác với nhau nhằm đảm bảo tính nhất quán, hiệu quả và an toàn của dữ liệu.

Bên cạnh đó, chương 2 cũng đã giới thiệu cách tiếp cận trong việc xây dựng ứng dụng bằng framework Flask, kết hợp với thư viện Flask-SQLAlchemy để quản lý và thao tác dữ liệu theo mô hình ORM. Việc áp dụng Jinja2 trong xây dựng giao diện người dùng giúp tách biệt rõ ràng giữa phần xử lý nghiệp vụ và phần hiển thị, từ đó góp phần tổ chức mã nguồn một cách khoa học, dễ đọc, dễ mở rộng và thuận tiện cho việc bảo trì trong tương lai.

Những nội dung thiết kế được trình bày trong chương này đóng vai trò là nền tảng kỹ thuật quan trọng, đảm bảo rằng hệ thống có thể được triển khai và vận hành một cách ổn định trong môi trường thực tế. Dựa trên cơ sở đó, Chương 3 của báo cáo sẽ tập trung trình bày quy trình triển khai hệ thống lên môi trường máy chủ **AWS EC2**, bao gồm cấu hình máy chủ, cài đặt các thành phần cần thiết và kiểm thử khả năng hoạt động của hệ thống sau khi triển khai.

## Chương 3

# Triển khai và Cấu hình

Sau khi hoàn thiện mã nguồn và kiểm thử ở môi trường cục bộ (Localhost), bước tiếp là đưa hệ thống lên môi trường thực tế (Production). Chương này trình bày quy trình thiết lập môi trường phát triển, cấu hình kết nối cơ sở dữ liệu và các bước triển khai ứng dụng lên hạ tầng đám mây AWS EC2.

### 3.1 Môi trường phát triển

Để đảm bảo tính nhất quán giữa các thành viên phát triển và môi trường server, dự án sử dụng môi trường ảo (Virtual Environment) và quản lý các thư viện phụ thuộc chặt chẽ.

#### 3.1.1 Cài đặt thư viện

Tất cả các thư viện cần thiết cho Flask và các tiện ích mở rộng được liệt kê trong file `requirements.txt`. Quy trình cài đặt bao gồm:

- Khởi tạo môi trường ảo để cách ly gói cài đặt với hệ thống chính.
- Cài đặt các gói như: `Flask`, `Flask-SQLAlchemy`, `Flask-Migrate`, `PyMySQL`, `gunicorn` (cho production).

Dưới đây là nội dung tóm tắt của file cấu hình thư viện:

```
1 Flask==3.1.2
2 Flask-SQLAlchemy==3.1.1
3 PyMySQL==1.1.2
4 Werkzeug==3.1.3
5 SQLAlchemy==2.0.45
```

Listing 3.1: File requirements.txt

### 3.2 Cấu hình kết nối MySQL

Việc quản lý thông tin nhạy cảm (như mật khẩu cơ sở dữ liệu, Secret Key) được thực hiện thông qua biến môi trường (Environment Variables) để đảm bảo bảo

mật, tránh hard-code trực tiếp vào mã nguồn.

### 3.2.1 Quản lý biến môi trường

Sử dụng thư viện python-dotenv để tải cấu hình từ file .env.

```

1 import os
2 from dotenv import load_dotenv
3
4 load_dotenv()
5
6 class Config:
7     SECRET_KEY = os.environ.get('SECRET_KEY') or 'dev-key'
8
9     # MySQL string connection
10    SQLALCHEMY_DATABASE_URI = (
11        f"mysql+pymysql://{{os.environ.get('DB_USER')}}:{os.environ.get('DB_PASS')}}@{{os.environ.get('DB_HOST')}}:3306/{{os.environ.get('DB_NAME')}}"
12    )
13    SQLALCHEMY_TRACK_MODIFICATIONS = False
14    #
15    ...
16
17

```

Listing 3.2: Cấu hình kết nối trong config.py

- **DB\_HOST:** Tại máy local là localhost, khi lên server sẽ là địa chỉ IP của EC2 hoặc localhost nếu cài MySQL ngay trên server.
- **DB\_USER / DB\_PASS:** Tài khoản quản trị cơ sở dữ liệu được cấp quyền.

## 3.3 Triển khai trên AWS EC2

Hệ thống được triển khai trên nền tảng Amazon Web Services (AWS) sử dụng dịch vụ Elastic Compute Cloud (EC2). Hệ điều hành được lựa chọn là **Ubuntu Server 22.04 LTS** nhờ tính ổn định và cộng đồng hỗ trợ lớn.

### 3.3.1 Cấu hình Instance và Security Groups

Để ứng dụng có thể truy cập được từ internet và quản trị viên có thể thao tác, Security Group (tường lửa ảo) của AWS được cấu hình mở các cổng sau:

- **Port 22 (SSH):** Cho phép truy cập từ xa để quản trị server.
- **Port 80 (HTTP):** Cổng web mặc định cho người dùng truy cập.
- **Port 5000 (Custom):** Cổng chạy thử nghiệm của Flask (thường được proxy qua Nginx).
- **Port 3306 (MySQL):** Mở giới hạn IP (tùy chọn) để kết nối tool quản lý DB từ máy cá nhân.

### 3.3.2 Thiết lập Web Server (Gunicorn và Nginx)

Trong môi trường Production, không sử dụng server tích hợp sẵn của Flask (development server) vì hiệu năng thấp và kém an toàn. Kiến trúc triển khai bao gồm:

- **Gunicorn (Green Unicorn):** Là một WSGI HTTP Server, đóng vai trò chạy mã nguồn Python Flask, xử lý các request động song song.
- **Nginx:** Đóng vai trò Reverse Proxy (Proxy ngược), tiếp nhận request từ cổng 80, phục vụ các file tĩnh (CSS/Javascript) và chuyển tiếp request động vào Gunicorn.

Cấu hình Nginx (/etc/nginx/sites-available/hospital):

```

1 #HTTP -> HTTPS
2 server {
3     server_name benhvientuhan.webredirect.org;
4
5     location / {
6         proxy_pass http://127.0.0.1:8000;
7         proxy_set_header Host $host;
8         proxy_set_header X-Real-IP $remote_addr;
9         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
10        proxy_set_header X-Forwarded-Proto $scheme;
11    }
12
13    listen 443 ssl; # managed by Certbot
14    ssl_certificate /etc/letsencrypt/live/benhvientuhan.webredirect.org/fullchain.pem
15    ;
16    ssl_certificate_key /etc/letsencrypt/live/benhvientuhan.webredirect.org/privkey.pem;
17    include /etc/letsencrypt/options-ssl-nginx.conf;
18    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
19 }
20
21 server {
22     if ($host = benhvientuhan.webredirect.org) {
23         return 301 https://$host$request_uri;
24     } # managed by Certbot
25
26
27     listen 80;
28     server_name benhvientuhan.webredirect.org;
29     return 404; # managed by Certbot
30
31 }
32 }
```

Listing 3.3: Cấu hình Nginx Reverse Proxy

### 3.3.3 Quản lý và Di chuyển Dữ liệu (Database Migration)

Trong mô hình này, MySQL Server được cài đặt và vận hành trực tiếp trên cùng Instance EC2 để tối ưu chi phí và độ trễ.

Quy trình khởi tạo dữ liệu trên Server:

1. Cài đặt MySQL Server trên Ubuntu: `sudo apt install mysql-server`.

2. Tạo Database và User tương ứng với cấu hình trong file .env.
3. Sử dụng **Flask-Migrate** để đồng bộ cấu trúc bảng (Schema) từ code lên Server:

```

1 # migration folder init
2 flask db init
3
4 # Create script migration from models
5 flask db migrate -m "Initial migration on EC2"
6
7 # Apply the changes to the MySQL Server.
8 flask db upgrade

```

Listing 3.4: Lệnh khởi tạo Database

- Nhờ cơ chế này, bất kỳ thay đổi nào về cấu trúc bảng ở máy local đều có thể cập nhật lên server một cách an toàn mà không cần xóa đi làm lại.

### 3.4 Kết luận chương 3

Chương 3 đã trình bày một cách hệ thống và chi tiết toàn bộ quy trình triển khai ứng dụng quản lý bệnh viện từ môi trường phát triển cục bộ lên môi trường Internet. Nội dung chương tập trung vào các bước cấu hình máy chủ, cài đặt môi trường thực thi, cũng như thiết lập các thành phần cần thiết để hệ thống có thể vận hành ổn định trên hạ tầng điện toán đám mây AWS EC2.

Việc kết hợp **Gunicorn** làm application server và **Nginx** làm web server giúp nâng cao hiệu năng và độ ổn định của hệ thống, đồng thời cải thiện khả năng xử lý đồng thời nhiều yêu cầu truy cập từ người dùng. Mô hình triển khai này cho phép ứng dụng Flask hoạt động bền bỉ hơn, phù hợp với các hệ thống Web yêu cầu tính sẵn sàng cao trong môi trường thực tế.

Bên cạnh đó, chương 3 cũng đã trình bày quy trình quản lý và cập nhật cơ sở dữ liệu thông qua cơ chế **Migration**. Việc áp dụng Migration không chỉ giúp kiểm soát tốt các thay đổi về cấu trúc dữ liệu theo từng phiên bản, mà còn hạn chế tối đa các sai sót phát sinh trong quá trình nâng cấp và mở rộng hệ thống. Đây là một yếu tố quan trọng góp phần nâng cao tính chuyên nghiệp và khả năng bảo trì lâu dài của ứng dụng.

Những kết quả đạt được trong quá trình triển khai là cơ sở để tiến hành đánh giá hiệu năng, độ ổn định và tính khả thi của hệ thống. Trên nền tảng đó, Chương 4 của báo cáo sẽ tập trung phân tích và đánh giá chi tiết kết quả thực nghiệm, cũng như mức độ đáp ứng của hệ thống đối với các yêu cầu đã đề ra ban đầu.

## Chương 4

# Kết quả thực nghiệm và đánh giá

### 4.1 Các kết quả đạt được

#### 4.1.1 Các tính năng

Hiện tại, website đang hoạt động tại đường dẫn [benhvientuthan.webredirect.org](http://benhvientuthan.webredirect.org) với các tài khoản demo như sau:

- **Admin:** Username: a, mật khẩu: a
- **Bác sĩ:** Username: b1, mật khẩu: 1
- **Bệnh nhân:** Username: p51, mật khẩu: 220925

Website hướng đến hỗ trợ quản lý nội bộ bệnh viện gồm các tính năng chính như dashboard phân tích, quản lý bác sĩ, bệnh nhân, các đơn thuốc, kho thuốc và phòng bệnh/giường bệnh.

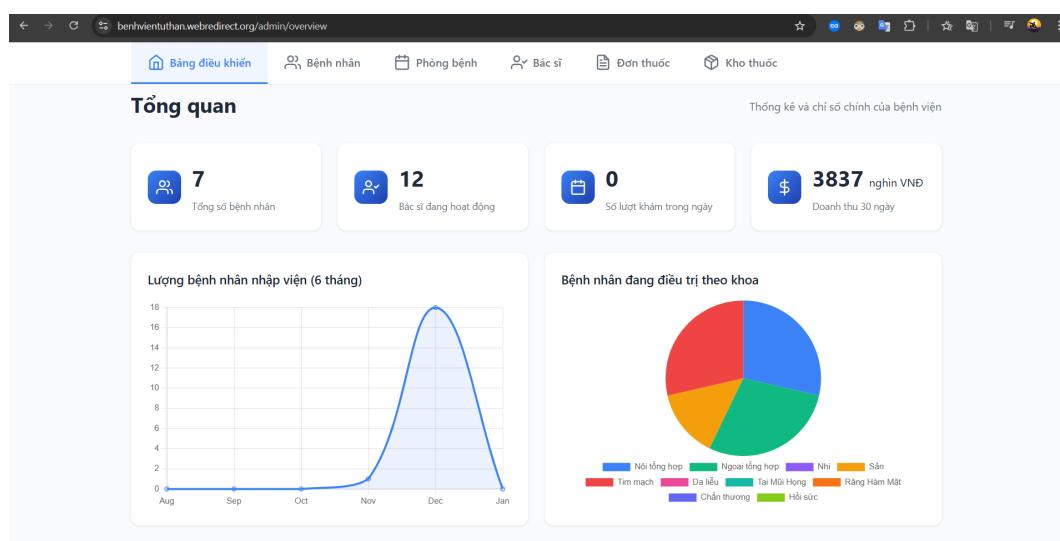
- **Dashboard:** Các biểu đồ phân tích kĩ thuật, tổng kết doanh thu, đánh giá hiệu suất làm việc của các bác sĩ,...
- **Quản lý bác sĩ:** Xem các thông tin, trạng thái cũng như hiệu suất làm việc của từng bác sĩ, các đơn thuốc được kê và tiến triển của tình trạng bệnh nhân của bác sĩ cụ thể. Các thao tác CRUD bác sĩ, bệnh nhân,... của role Admin.
- **Quản lý bệnh nhân:** Xem các thông tin, trạng thái cũng như các đơn thuốc và tiến triển tình trạng của bệnh nhân qua từng đơn thuốc. Các thao tác CRUD bệnh nhân của role Bác sĩ.
- **Quản lý đơn thuốc:** Theo dõi tất cả các đơn thuốc được tạo (bác sĩ sẽ xem được các đơn thuốc của khoa mình), đối với bác sĩ từ bệnh nhân được kê đơn có thể truy về các đơn thuốc được kê trước đó.
- **Quản lý kho thuốc:** Admin có thể CRUD trên các lô hàng trong kho thuốc (hạn sử dụng, giá bán) đồng thời nhận được cảnh báo khi có lô sắp hết hạn hoặc sắp hết hàng.

- **Quản lý phòng bệnh:** Admin có thể xem các phòng bệnh của bệnh viện, các bệnh nhân và các thông tin liên lạc, thân nhân trong phòng cụ thể.

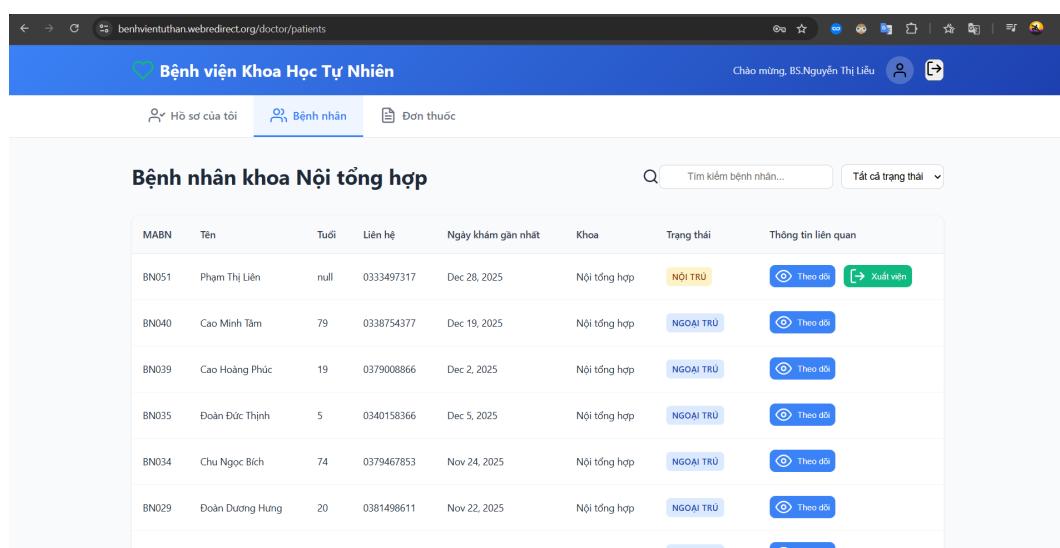
Bên cạnh đó, website hỗ trợ quản lý phiên làm việc bằng tính năng đăng nhập và lưu thông tin bằng cookie. Tài khoản bác sĩ được tạo tự động khi Admin thêm bác sĩ và tài khoản bệnh nhân được tạo tự động khi bác sĩ tạo đơn thuốc cho bệnh nhân mới.

#### 4.1.2 Demo giao diện

Các hình từ 3.1 đến 3.7 là các hình ảnh demo giao diện website.



Hình 4.1: Giao diện Dashboard phân tích tổng thể của role Admin



Hình 4.2: Giao diện trang bệnh nhân của role bác sĩ

Họ và tên	Trình độ	Khoa	Phòng khám	Liên hệ	Đang làm việc
Nguyễn Thị Liễu	BS	Nội tổng hợp	P104	0995822412	TRUE
Trần Minh Quân	ThS	Nội tổng hợp	P108	0946913810	TRUE
Lê Thị Thu Hà	CKII	Ngoại tổng hợp	P104	0928728463	TRUE
Phạm Đức Long	CKI	Ngoại tổng hợp	P119	0921668732	TRUE
Vũ Ngọc Mai	BS	Nhi	P101	0914265799	TRUE
Đặng Hoàng Nam	CKII	Nhi	P108	0939345092	TRUE
Hoàng Thị Kim Lan	CKII	Sản	P101	0990801586	TRUE

Hình 4.3: Giao diện trang bác sĩ của role Admin

## 4.2 Đánh giá hiệu năng trên máy chủ EC2

Hiện tại nhóm chưa thực sự có đánh giá khách quan để đối chiếu trong triển khai thực tế, các kết quả đo đạc được trình bày bên dưới là trên bộ dữ liệu tự tạo gồm 20 bác sĩ, 50 bệnh nhân, gần 100 sổ khám bệnh và các bản ghi của các bảng khác được sinh ngẫu nhiên tương ứng (đảm bảo logic).

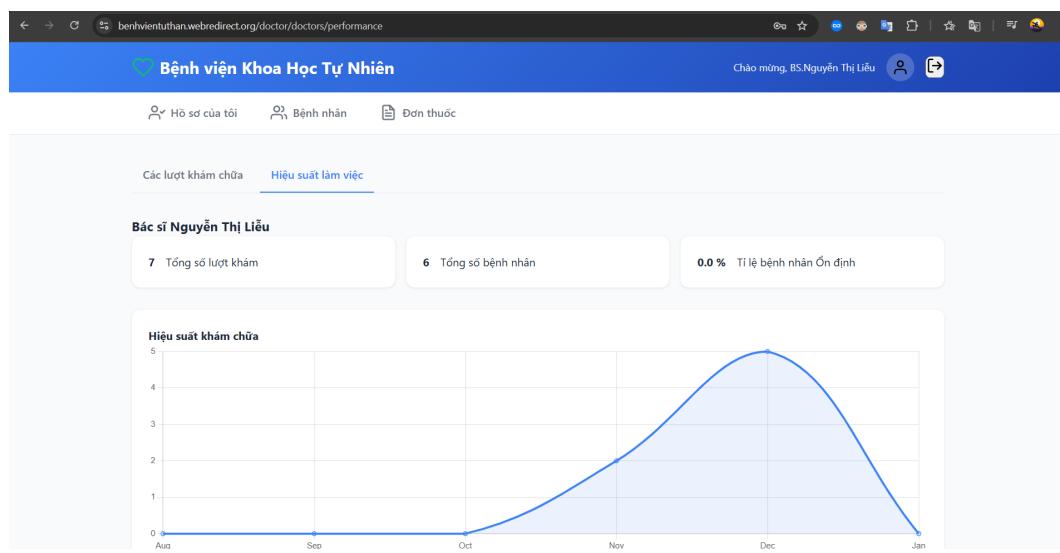
Với tập dữ liệu này, dù hạn chế của máy chủ EC2 (Free tier) hệ thống vẫn đạt hiệu suất truy vấn rất tốt. Dưới đây liệt kê các đo đạc trong role Admin - role nặng về truy vấn nhất.

- **Dashboard** : 0.09073s (do có lazy loading cho các biểu đồ)
- **Bệnh nhân** : 0.28707s
- **Bác sĩ** : 0.23320s
- **Đơn thuốc** : 0.39325s
- **Kho thuốc** : 0.19920s
- **Phòng bệnh**: 0.19803s

## 4.3 Các hạn chế

Hiện tại, website còn tồn tại một số hạn chế:

- **Bảo mật:** Tính năng quản lý phiên đăng nhập lưu thông tin người dùng bằng cookie tồn tại nhiều điểm hạn chế trước các loại tấn công đánh cắp thông tin; bị phụ thuộc vào trình duyệt của người dùng.



Hình 4.4: Giao diện phân tích hiệu suất làm việc của bác sĩ

- **Cấu hình máy chủ:** Máy chủ EC2 bản miễn phí của AWS với cấu hình gồm 1 cpu, 1GB RAM - khá thấp - dẫn đến hiệu suất truy vấn còn hạn chế.

Bệnh nhân	Tình trạng	Thuốc	Lи́ều lượng & Tần suất	Bác sĩ kê đơn	Ngày	Theo dõi bệnh nhân
Phạm Thị Liên	TRUNG BÌNH	Vitamin C	2 viên/ngày, 12 ngày	Nguyễn Thị Liễu	Dec 28, 2025	
Phạm Thị Liên	NHẸ	Vitamin C	2 viên/ngày, 12 ngày	Nguyễn Thị Liễu	Dec 24, 2025	
Nguyễn Thị Mai	NHẸ	Clarithromycin	7 viên/ngày, 5 ngày	Trần Minh Quân	Dec 20, 2025	
Cao Minh Tâm	TRUNG BÌNH	Ibuprofen	8 viên/ngày, 2 ngày	Trần Minh Quân	Dec 19, 2025	
Lưu Văn Sơn	TRUNG BÌNH	Cefixime	1 viên/ngày, 5 ngày	Trần Minh Quân	Dec 13, 2025	
Trương Thảo Vy	NHẸ	ORS	2 viên/ngày, 7 ngày	Nguyễn Thị Liễu	Dec 7, 2025	
Đoàn Đức Thịnh	TRUNG BÌNH	Prednisone	10 viên/ngày, 4 ngày	Nguyễn Thị Liễu	Dec 5, 2025	

Hình 4.5: Giao diện trang đơn thuốc của role bác sĩ

The screenshot shows a modal dialog titled "Thêm đơn thuốc mới" (Add new prescription) overlaid on a list of existing prescriptions. The dialog contains the following fields:

- Mã bệnh nhân \* (Patient ID): A dropdown menu showing "Chọn bệnh nhân" (Select patient).
- Tên bệnh \* (Disease name): A dropdown menu showing "Chọn danh mục" (Select category).
- Giai đoạn \* (Stage): A dropdown menu showing "Chọn giai đoạn" (Select stage).
- Tình trạng \* (Status): A dropdown menu showing "Chọn tình trạng" (Select status).
- Hình thức điều trị \* (Treatment form): A dropdown menu showing "Chọn loại" (Select type).

Hình 4.6: Giao diện thêm đơn thuốc mới của role bác sĩ

The screenshot shows the 'Bệnh viện Khoa Học Tự Nhiên' (Pharmacy Management System) interface. The top navigation bar includes links for 'Bảng điều khiển', 'Bệnh nhân', 'Phòng bệnh', 'Bác sĩ', 'Đơn thuốc', and 'Kho thuốc'. The 'Kho thuốc' tab is active. A blue button '+ Thêm lô thuốc mới' is visible. The main content area displays a table of drugs in stock, showing columns for Name, Quantity in lot, Stock level, Expiry date, Number of lots, Total quantity, and Total stock level. It also shows two warning messages: 'Tồn kho thấp: Amlodipine (147/1000)' and 'Sắp hết hạn: Omeprazole (Mã lô: 2) hết hạn vào Jan 31, 2026'.

Tên thuốc	Số lượng trong lô	Mức tồn kho	Hạn sử dụng	Số lô cùng loại	Số lượng tổng	Mức tồn kho tổng
Amlodipine	71	14%	Jan 22, 2028	2	147	15%
Amlodipine	76	15%	Feb 5, 2028	2	147	15%
Amoxicillin	354	71%	Aug 26, 2026	2	839	84%

Hình 4.7: Giao diện trang kho thuốc của role Admin

This screenshot shows the 'Thêm lô thuốc mới' (Add New Drug Batch) dialog box overlaid on the main inventory management page. The dialog has tabs for 'Thuốc mới' (New drug) and 'Thuốc đã có' (Existing drug). It contains fields for 'Tên thuốc \*' (Drug name), 'Công dụng \*' (Function), 'Giá bán (VND) \*' (Sale price), and 'Số lượng \*' (Quantity). The background table of drugs is partially visible.

Hình 4.8: Giao diện tạo lô thuốc mới của role Admin

The screenshot shows a web application interface for managing treatment rooms. At the top, there's a blue header bar with the title 'Bệnh viện Khoa Học Tự Nhiên'. Below it, a navigation bar includes links for 'Bảng điều khiển', 'Bệnh nhân', 'Phòng bệnh' (which is highlighted in blue), 'Bác sĩ', 'Đơn thuốc', and 'Kho thuốc'. The main content area is titled 'Quản lý phòng bệnh'. It displays a table with columns: Phòng (Room), Khoa (Department), Tòa nhà (Building), Tình trạng giường (Bed status), and Danh sách bệnh nhân (List of patients). The table contains six rows of data:

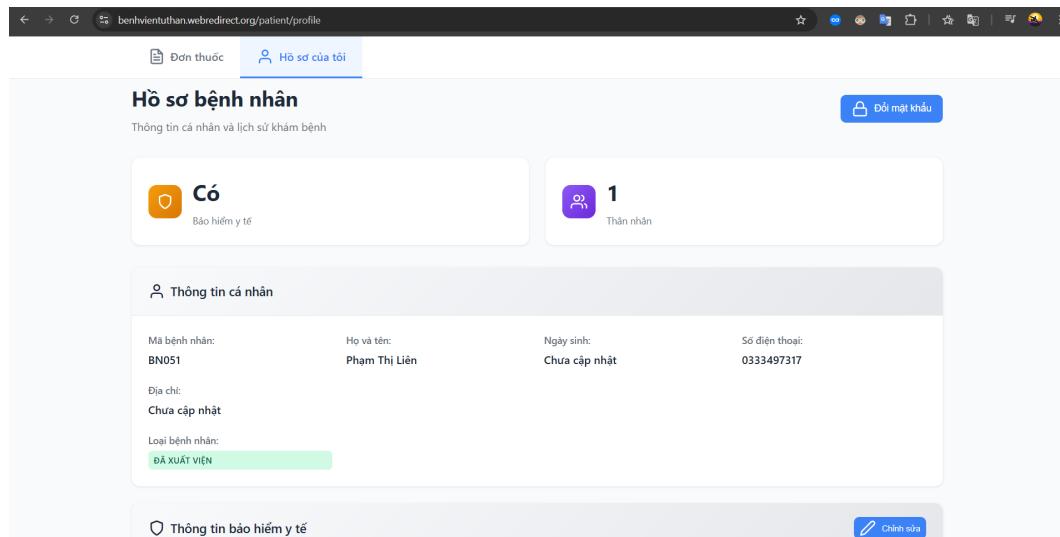
Phòng	Khoa	Tòa nhà	Tình trạng giường	Danh sách bệnh nhân
PHG001	Nội tổng hợp	Tòa A	2 / 4	<a href="#">Danh sách bệnh nhân</a>
PHG011	Nội tổng hợp	Tòa A	0 / 6	<a href="#">Danh sách bệnh nhân</a>
PHG002	Ngoại tổng hợp	Tòa B	0 / 4	<a href="#">Danh sách bệnh nhân</a>
PHG012	Ngoại tổng hợp	Tòa B	0 / 6	<a href="#">Danh sách bệnh nhân</a>
PHG003	Nhi	Tòa B	1 / 4	<a href="#">Danh sách bệnh nhân</a>
PHG013	Nhi	Tòa B	0 / 6	<a href="#">Danh sách bệnh nhân</a>

Hình 4.9: Giao diện trang phòng bệnh của role Admin

This screenshot shows a modal window titled 'Danh sách bệnh nhân - Phòng PHG004 (Tòa A)'. The modal lists patients with columns: Họ tên (Name), Số điện thoại (Phone number), Số điện thoại người thân (Relative phone number), and Tình trạng (Status). The data is as follows:

Họ tên	Số điện thoại	Số điện thoại người thân	Tình trạng
Bùi Thành Sơn	0388461803	0860867083	NÂNG
Bùi Thu Trang	0358586340	0855150390	NHẸ
Đặng Hữu Phúc	0322201654	0813852048	NÂNG
Đặng Mỹ Linh	0330514014	0876344695	NÂNG

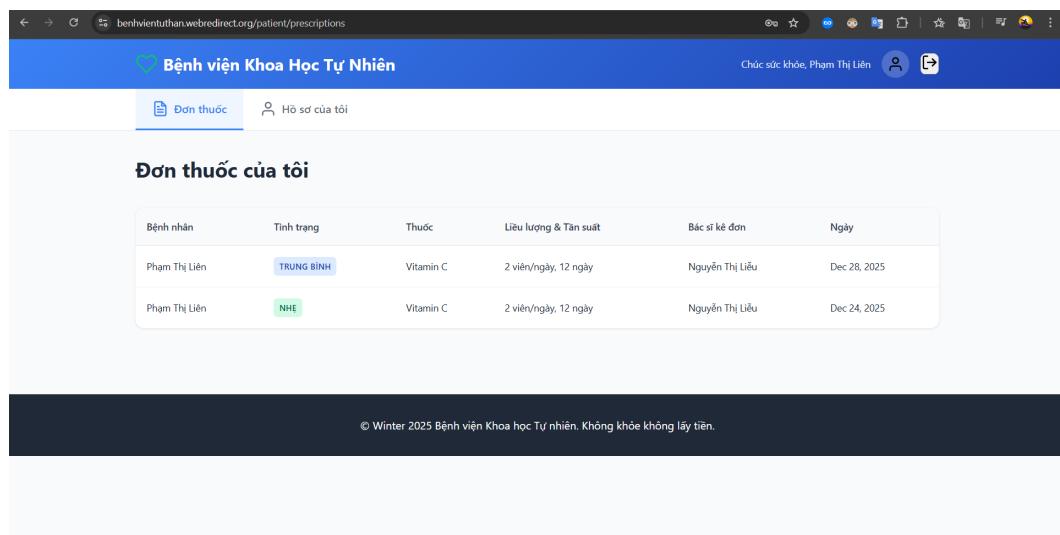
Hình 4.10: Giao diện danh sách bệnh nhân ở 1 phòng cụ thể của role Admin



Hình 4.11: Giao diện hồ sơ cá nhân của role bệnh nhân

MABN	Tên	Tuổi	Liên hệ	Ngày khám gần nhất	Khoa	Trạng thái	Thông tin liên quan
BN051	Phạm Thị Liên	null	0333497317	Dec 28, 2025	Nội tổng hợp	ĐÃ XUẤT VIỆN	<button> Theo dõi</button>
BN050	Hoàng Ngọc Mai	31	0333497317	Dec 4, 2025	Ngoại trú	NGOAI TRÙ	<button> Theo dõi</button>
BN049	Hoàng Quốc Thái	51	0333497317	Dec 4, 2025	Ngoại trú	NGOAI TRÙ	<button> Theo dõi</button>
BN048	Phạm Yến Nhi	41	0333497317	Dec 4, 2025	Ngoại trú	NGOAI TRÙ	<button> Theo dõi</button>
BN047	Phạm Hữu Dật	41	0333497317	Dec 24, 2025	Ngoại trú	NGOAI TRÙ	<button> Theo dõi</button>
BN046	Lê Thanh Văn	41	0333497317	Dec 28, 2025	TRUNG BÌNH	Vitamin C	Nguyễn Thị Liệu
BN045	Lê Đức Tài	41	0333497317	Dec 28, 2025	Ngoại trú	Vitamin C	Nguyễn Thị Liệu
BN044	Trần Ngọc Hân	44	0397705310	Dec 16, 2025	Sản	NGOAI TRÙ	<button> Theo dõi</button>
BN043	Trần Thành Bình	34	0366851760	Nov 20, 2025	Chấn thương	NGOAI TRÙ	<button> Theo dõi</button>

Hình 4.12: Giao diện theo dõi các đơn thuốc của bệnh nhân role Admin và bác sĩ



Hình 4.13: Giao diện trang các đơn thuốc được kê cho tôi của role bệnh nhân

## Chương 5

# Kết luận và hướng phát triển

### 5.1 Kết luận

Nhóm đã hoàn thành xây dựng hệ thống cơ sở dữ liệu quan hệ toàn diện từ triển khai, lưu trữ, khai thác, trực quan, thêm-sửa-xóa hỗ trợ công tác quản lý bệnh viện. Về mặt kỹ thuật, thành tựu nổi bật nhất của đề tài là việc thiết kế và khai thác thành công cơ sở dữ liệu quan hệ trong lĩnh vực y tế - một lĩnh vực phức tạp và cần sự chính xác cao. Bên cạnh đó, hệ thống back-end tổ chức theo kiến trúc ORM hoạt động với tính mở rộng và bảo trì dễ dàng hoạt động trên máy chủ cloud. Về mặt trực quan, giao diện website đơn giản, thân thiện, hỗ trợ tiếng Việt, dễ dùng đối với cả người lớn tuổi.

### 5.2 Hướng phát triển

#### 5.2.1 Ngắn hạn

Trong giai đoạn ngắn hạn, nhóm nghiên cứu sẽ tập trung vào việc tối ưu hóa các câu lệnh truy vấn trong ORM gồm các cách như: loại bỏ các trường không cần thiết trong các giai đoạn lấy dữ liệu đưa lên tầng Business Logic, lọc sớm, hạn chế join thừa. Song song với đó, bổ sung thêm dữ liệu vào cơ sở dữ liệu để kiểm tra hiệu năng truy vấn với lượng dữ liệu lớn.

#### 5.2.2 Dài hạn

Về dài hạn, nhóm đề xuất phát triển thêm các tính năng mới nhằm tăng sự tương tác giữa bác sĩ và bệnh nhân như tính năng nhắn tin, đặt lịch hẹn khám,... Mở rộng thêm các tính năng bên phía bệnh nhân như các tính năng thanh toán viện phí bằng QR, thông báo lịch uống thuốc, hỏi đáp với chatbot dựa trên đơn thuốc,...

### 5.3 Kết luận cuối cùng

Dự án “Website Quản lý Bệnh viện” không chỉ dừng lại ở việc xây dựng một hệ thống số hóa các quy trình nghiệp vụ y tế, mà còn thể hiện rõ tính khả thi và hiệu quả của việc ứng dụng công nghệ thông tin trong công tác quản lý và vận hành bệnh viện hiện đại. Với kiến trúc hệ thống linh hoạt, khả năng mở rộng cao và giao diện thân thiện với người dùng, website cho thấy tiềm năng lớn trong việc nâng cao hiệu quả quản lý hồ sơ bệnh án, tối ưu hóa quy trình khám chữa bệnh và cải thiện chất lượng dịch vụ y tế. Thành công của dự án góp phần khẳng định rằng công nghệ không nhằm thay thế vai trò của đội ngũ y bác sĩ và nhân viên y tế, mà đóng vai trò như một công cụ hỗ trợ đắc lực, giúp giảm tải công việc hành chính, nâng cao độ chính xác trong quản lý dữ liệu và từ đó hỗ trợ đưa ra các quyết định điều hành và chăm sóc sức khỏe dựa trên thông tin kịp thời và đáng tin cậy.

# Tài liệu tham khảo

- [1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2016.
- [2] Oracle Corporation, *MySQL 8.0 Reference Manual*. [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/>
- [3] Pallets Projects, *Flask Documentation (Version 3.0.x)*. [Online]. Available: <https://flask.palletsprojects.com/>
- [4] M. Bayer, *SQLAlchemy 2.0 Documentation - The Database Toolkit for Python*. [Online]. Available: <https://www.sqlalchemy.org/>
- [5] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. O'Reilly Media, 2018.
- [6] Pallets Projects, *Werkzeug Security Helpers (Password Hashing)*. [Online]. Available: <https://werkzeug.palletsprojects.com/en/3.0.x/utils/>
- [7] Amazon Web Services, *Amazon EC2 User Guide for Linux Instances*. [Online]. Available: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/>
- [8] Docker Inc., *Docker Compose Documentation*. [Online]. Available: <https://docs.docker.com/compose/>

## Phụ lục A

# Hướng dẫn Vận hành và Bảo trì

### A.1 Tài liệu cài đặt và hướng dẫn sử dụng

#### A.1.1 Yêu cầu hệ thống

Hệ thống web quản lý bệnh viện được triển khai theo mô hình client-server và được đóng gói bằng Docker nhằm đảm bảo tính nhất quán của môi trường, dễ dàng triển khai và thuận tiện cho quá trình đánh giá.

Các yêu cầu tối thiểu của hệ thống bao gồm:

- Hệ điều hành Linux (Ubuntu 20.04 trở lên) hoặc Windows có hỗ trợ Docker Desktop.
- Docker Engine phiên bản 20.10 trở lên.
- Docker Compose phiên bản 2.x.
- Trình duyệt web hiện đại như Google Chrome, Microsoft Edge hoặc Mozilla Firefox.

#### A.1.2 Cấu trúc mã nguồn

Mã nguồn của hệ thống được tổ chức theo kiến trúc phân tầng, tách biệt rõ ràng giữa phần giao diện người dùng, xử lý nghiệp vụ và quản lý dữ liệu, giúp nâng cao khả năng bảo trì và mở rộng hệ thống.

Cấu trúc thư mục chính của dự án bao gồm:

- `app/`: Chứa mã nguồn backend và frontend của ứng dụng web.
- `db_script/`: Chứa file `init_database.sql` dùng để khởi tạo cơ sở dữ liệu ban đầu.
- `Dockerfile`: Mô tả các bước xây dựng Docker image cho ứng dụng.
- `docker-compose.yml`: Cấu hình các dịch vụ container (ứng dụng web và cơ sở dữ liệu), mạng nội bộ và các biến môi trường.

### A.1.3 Hướng dẫn cài đặt và triển khai bằng Docker

Hệ thống được triển khai bằng Docker Compose nhằm tự động hóa quá trình cài đặt và giảm thiểu sự phụ thuộc vào cấu hình thủ công của máy chủ.

#### Bước 1: Chuẩn bị mã nguồn

Người dùng tải mã nguồn của dự án từ kho lưu trữ GitHub và truy cập vào thư mục gốc của dự án:

```
1 git clone https://github.com/<repository-name>/Hospital-Network.git  
2 cd Hospital-Network
```

#### Bước 2: Khởi tạo và chạy hệ thống

Tại thư mục gốc của dự án, sử dụng Docker Compose để khởi động toàn bộ hệ thống:

```
1 docker compose up -d
```

Lệnh trên sẽ khởi động các container của hệ thống ở chế độ nền (detached mode), bao gồm:

- Container ứng dụng web quản lý bệnh viện.
- Container hệ quản trị cơ sở dữ liệu MySQL.

Sau khi quá trình khởi động hoàn tất, hệ thống sẵn sàng để sử dụng thông qua trình duyệt web.

#### Tắt hệ thống

Khi không còn nhu cầu sử dụng, người dùng có thể dừng toàn bộ hệ thống bằng lệnh:

```
docker compose down
```

Lệnh này sẽ dừng và gỡ bỏ các container đang chạy, đồng thời giải phóng các tài nguyên mạng đã được Docker Compose tạo ra.

Trong trường hợp chỉ cần tạm dừng hệ thống mà không gỡ bỏ container, có thể sử dụng lệnh:

```
docker compose stop
```

Sau đó, hệ thống có thể được bật lại nhanh chóng bằng lệnh:

```
docker compose start
```

## Lưu ý vận hành

Trong quá trình triển khai trên máy chủ, việc bật và tắt hệ thống bằng Docker Compose giúp đảm bảo tính an toàn dữ liệu và hạn chế lỗi phát sinh so với việc tắt trực tiếp máy chủ.

Ngoài ra, cơ sở dữ liệu được lưu trữ thông qua Docker volume, do đó dữ liệu không bị mất khi hệ thống được tắt và khởi động lại.

### A.1.4 Hướng dẫn sử dụng hệ thống

Người dùng truy cập hệ thống thông qua trình duyệt web tại địa chỉ:

1 `http://localhost:5000`

Trong trường hợp triển khai trên máy chủ, hệ thống có thể được truy cập thông qua tên miền đã được cấu hình tương ứng.

Sau khi đăng nhập, người dùng có thể sử dụng các chức năng chính của hệ thống, bao gồm:

- Đăng nhập và phân quyền người dùng theo vai trò (quản trị viên, bác sĩ, nhân viên).
- Quản lý thông tin bệnh nhân và hồ sơ khám chữa bệnh.
- Quản lý thông tin bác sĩ, khoa phòng và lịch khám.
- Tra cứu, cập nhật và xử lý dữ liệu theo thời gian thực.

### A.1.5 Khả năng mở rộng và tái triển khai

Việc sử dụng Docker giúp hệ thống có khả năng tái triển khai dễ dàng trên nhiều môi trường khác nhau mà không cần thay đổi cấu hình nội bộ của ứng dụng.

Bên cạnh đó, kiến trúc dựa trên container cho phép hệ thống mở rộng trong tương lai bằng cách tích hợp thêm các dịch vụ mới như hệ thống phân tích dữ liệu, API bên thứ ba hoặc các dịch vụ giám sát và ghi log.

## A.2 Quy trình Backup và khôi phục dữ liệu

Hệ thống được triển khai trên AWS EC2 sử dụng Docker Compose để đóng gói ứng dụng và cơ sở dữ liệu. Do đó, quy trình sao lưu (Backup) và khôi phục (Restore) dữ liệu sẽ được thực hiện thông qua giao diện dòng lệnh của Docker (Docker CLI) thay vì thao tác trực tiếp trên hệ điều hành chủ.

Cấu hình `docker-compose.yml` có service database tên là `db` và database tên là `hospital_db`.

### A.2.1 Truy cập Server

Trước khi thực hiện bất kỳ thao tác nào, cần truy cập vào instance EC2 thông qua SSH:

```
1 ssh -i "/home/khachuy/.ssh/Hospital.pem" ubuntu@ec2-3-1-222-241.ap-southeast-1.compute.amazonaws.com
2 cd /home/ubuntu/Hospital-Network
```

Listing A.1: Truy cập SSH vào EC2

### A.2.2 Sao lưu dữ liệu (Backup)

Để xuất dữ liệu từ container MySQL đang chạy ra file .sql trên máy chủ (Host), ta sử dụng lệnh `mysqldump` thông qua `docker compose exec`.

- Câu lệnh dưới đây sẽ tạo ra một file backup kèm theo ngày tháng hiện tại để dễ quản lý.

```
1 # Backup with datetime
2 docker compose exec db mysqldump -u root -p[ROOT_PASSWORD] hospital_db > backup_$(date +%F).sql
```

Listing A.2: Lệnh Backup dữ liệu từ Container

*Lưu ý: Không có khoảng trắng giữa tham số `-p` và mật khẩu.*

### A.2.3 Khôi phục dữ liệu (Restore/Import)

Quy trình này thường được sử dụng khi cần nạp dữ liệu mẫu ban đầu hoặc khôi phục lại hệ thống khi gặp sự cố.

- **Bước 1:** Chuẩn bị file .sql cần import (ví dụ: `data_dump.sql`). Nếu file này nằm ở máy cá nhân, cần upload lên EC2 bằng lệnh `scp`.
- **Bước 2:** Sử dụng lệnh `docker compose exec` với tham số `-T` (tắt pseudo-tty) để nhận luồng dữ liệu từ file vào container.

```
1 # Upload file from pc
2 scp -i "/home/khachuy/.ssh/Hospital.pem" data_dump.sql ubuntu@ec2-3-1-222-241.ap-southeast-1.compute.amazonaws.com:/home/ubuntu/Hospital-Network/
3
4 # Import data to MySQL Container
5 cat data_dump.sql | docker compose exec -T db mysql -u root -p[ROOT_PASSWORD] hospital_db
```

Listing A.3: Lệnh Import file SQL vào Container

#### A.2.4 Tự động hóa (Automation)

Để đảm bảo an toàn dữ liệu, có thể thiết lập Cronjob trên Ubuntu để tự động backup định kỳ vào 2:00 sáng hàng ngày:

```
1 # Open crontab editor
2 crontab -e
3
4 # Append this into tail of file:
5 0 2 * * * cd /home/ubuntu/project && docker compose exec -T db mysqldump -u root -
   ppassword hospital_db > /home/ubuntu/backups/backup_$(date +\%F).sql
```

Listing A.4: Cấu hình Cronjob tự động backup

## Phụ lục B

# Danh sách API (API Endpoints)

Dưới đây là danh sách các Endpoint chính của hệ thống, được phân loại theo quyền hạn người dùng. Các tham số chi tiết (Request Body/Response JSON) được mô tả đầy đủ trong tài liệu kỹ thuật đi kèm mã nguồn, tham khảo chi tiết tại API-doc.

### B.1 Authentication (Dùng chung)

- [POST] /auth/login: Xác thực người dùng (Admin, Doctor, Patient) và điều hướng về trang Dashboard tương ứng.
- [GET] /auth/logout: Xóa session hiện tại và đăng xuất khỏi hệ thống.

### B.2 Admin API (Quản trị viên)

Tất cả các endpoint yêu cầu quyền ADMIN.

#### Quản lý Bác sĩ & Bệnh nhân

- [GET] /admin/doctors: Truy xuất danh sách và thông tin bác sĩ.
- [POST] /admin/doctors/add: Thêm bác sĩ mới (Hệ thống tự động sinh username/password).
- [GET] /admin/doctors/<id>/performent: Xem thống kê hiệu suất khám chữa bệnh.
- [GET] /admin/patients: Danh sách toàn bộ bệnh nhân trong hệ thống.
- [GET] /admin/patients/<id>: Lấy chi tiết lịch sử khám và thông tin thân nhân.
- [POST] /admin/patients/<id>/discharge: Thực hiện thủ tục xuất viện.

- [GET] /admin/room/<id>: Xem danh sách bệnh nhân đang điều trị tại phòng bệnh cụ thể.

### Quản lý Kho thuốc (Pharmacy)

- [GET] /admin/pharmacy: Xem danh sách tồn kho thuốc.
- [POST] /admin/add-medicine-batch: Nhập thêm lô hàng (batch) cho thuốc đã tồn tại.
- [POST] /admin/add-new-medicine-batch: Tạo mã thuốc mới và nhập lô hàng đầu tiên.

## B.3 Doctor API (Bác sĩ)

Tất cả các endpoint yêu cầu quyền DOCTOR.

- [GET] /doctor/examinations: Xem lịch sử các ca khám do bác sĩ thực hiện.
- [POST] /doctor/prescriptions/add-new: Kê đơn cho **bệnh nhân mới**. (Tự động tạo hồ sơ, tài khoản và xếp giường nếu là Nội trú).
- [POST] /doctor/prescriptions/add-existing: Kê đơn cho **bệnh nhân cũ**. (Cập nhật bệnh án, trừ tồn kho thuốc).
- [GET] /doctor/prescriptions/<id>: Lấy dữ liệu theo dõi sức khỏe (Monitoring Data).
- [GET] /doctor/prescriptions/check-patient/<id>: Kiểm tra sự tồn tại của bệnh nhân qua mã BN hoặc CCCD.

## B.4 Patient API (Bệnh nhân)

Tất cả các endpoint yêu cầu quyền BENHNHAN.

- [GET] /patient/prescriptions: Xem lịch sử khám bệnh và đơn thuốc của cá nhân.
- [GET] /patient/profile: Xem thông tin cá nhân, thân nhân và bảo hiểm y tế.
- [POST] /patient/change-password: Thay đổi mật khẩu đăng nhập.
- [POST] /patient/bhyt: Thêm mới hoặc cập nhật thông tin thẻ BHYT.
- [POST] /patient/relative: Bổ sung thông tin người thân/người giám hộ.