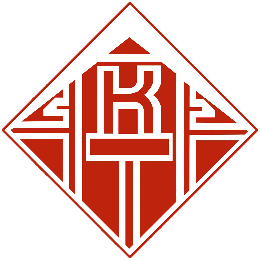
**TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG**

**Đoàn Tiến Lợi TÊN ĐỀ TÀI: NGHIÊN CỨU CÔNG NGHỆ FASTAPI – REACTJS XÂY DỰNG WEBSITE QUẢN LÝ HỒ SƠ BỆNH ÁN 2024**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI:**

**NGHIÊN CỨU CÔNG NGHỆ**

**FASTAPI – REACTJS XÂY DỰNG WEBSITE QUẢN LÝ HỒ SƠ BỆNH ÁN**

Người hướng dẫn: **ThS. NGUYỄN TẤT PHÚ CƯỜNG**

Sinh viên thực hiện: **ĐOÀN TIẾN LỢI**

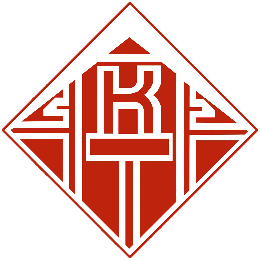
Mã số sinh viên: **2051220102**

Lớp**: 20CT3**

**Đà Nẵng, tháng 12/2024**

**TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI:**

**NGHIÊN CỨU CÔNG NGHỆ**

**FASTAPI – REACTJS XÂY DỰNG WEBSITE QUẢN LÝ HỒ SƠ BỆNH ÁN**

Người hướng dẫn: **ThS. NGUYỄN TẤT PHÚ CƯỜNG**

Sinh viên thực hiện: **ĐOÀN TIẾN LỢI**

Mã số sinh viên: **2051220102**

Lớp**: 20CT3**

**Đà Nẵng, tháng 12/2024**

# **TÓM TẮT**

Tên đề tài: Nghiên cứu Công nghệ FastAPI – ReactJS và Xây dựng Website Quản lý Hồ Sơ Bệnh Án

Sinh viên thực hiện: Đoàn Tiến Lợi

Mã số SV: 2051220102 Lớp: 20CT3

Đề tài nghiên cứu tập trung phát triển một hệ thống quản lý hồ sơ bệnh án điện tử (EHIS) hiện đại, nhằm cải thiện toàn diện quy trình chăm sóc sức khỏe và nâng cao trải nghiệm cho cả bác sĩ lẫn bệnh nhân. Hệ thống được thiết kế dựa trên các công nghệ tiên tiến như FastAPI để xây dựng backend mạnh mẽ và hiệu suất cao, ReactJS để tạo giao diện người dùng trực quan và thân thiện, cùng cơ sở dữ liệu PostgreSQL đảm bảo lưu trữ thông tin an toàn và đáng tin cậy.

Không chỉ dừng lại ở việc đặt lịch khám, quản lý thông tin bệnh án như lưu trữ, cập nhật, và theo dõi tiến trình điều trị, hệ thống còn tích hợp các công cụ giao tiếp từ xa như nhắn tin, chat bot hỗ trợ bác sĩ trong việc phân tích dữ liệu để dự đoán các vấn đề sức khỏe và tối ưu hóa quá trình kê đơn thuốc, đặc biệt hữu ích trong các tình huống khẩn cấp hoặc khi bệnh nhân không thể đến trực tiếp cơ sở y tế, từ đó nâng cao hiệu quả điều trị.

Hệ thống này không chỉ đáp ứng nhu cầu số hóa trong ngành y tế mà còn tạo bước đột phá trong việc ứng dụng công nghệ hiện đại vào chăm sóc sức khỏe, góp phần cải thiện hiệu quả vận hành tại các bệnh viện và cơ sở y tế, đồng thời đưa trải nghiệm của người bệnh lên một tầm cao mới.

# **LỜI NÓI ĐẦU**

Để hoàn thành được Đồ án tốt nghiệp này em xin chân thành cảm ơn quý thầy cô trong khoa Công Nghệ Thông Tin - Trường Đại Học Kiến Trúc Đà Nẵng đã tận tình truyền đạt kiến thức trong những năm học vừa qua. Với vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình làm đồ án tốt nghiệp mà còn là hành trang quý báu để em bước vào đời một cách vững chắc và tự tin.

Em cũng xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc của thầy Nguyễn Tất Phú Cường trong suốt thời gian vừa qua đã nhiệt tình chỉ dạy, giúp đỡ em để có thể hoàn thành tốt đồ án tốt nghiệp.

Trong quá trình làm bài đồ án tốt nghiệp, do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của quý báu của các thầy cô để em có thể học hỏi thêm được nhiều kinh nghiệm, kiến thức.

***Em xin trân trọng cảm ơn!***

# **CAM ĐOAN**

Em xin cam đoan rằng tất cả các kết quả và thông tin trình bày trong đề tài này là thành quả của công việc của em dưới sự hướng dẫn của ThS. Nguyễn Tất Phú Cường. Em đã tuân thủ mọi nguyên tắc và quy định về nghiên cứu khoa học và đạo đức trong việc thực hiện khóa luận này.

Tất cả các nguồn tham khảo và công trình nghiên cứu của người khác đã được em trích dẫn và trình bày một cách rõ ràng và chi tiết trong danh mục tài liệu tham khảo của khóa luận. Em đã tuân thủ nguyên tắc không sao chép, không vi phạm bản quyền, và đã tuân thủ đúng quy định về việc chỉ rõ nguồn gốc thông tin từ nguồn khác.

Các kết quả đánh giá và thống kê được trình bày trong khóa luận này đều là kết quả của những thực nghiệm mà em đã tự thực hiện, và không có sự can thiệp từ bất kỳ nguồn nào khác. Em cam đoan tính trung thực và chính xác trong việc trình bày kết quả và thông tin trong khóa luận này.

Sinh viên thực hiện

**MỤC LỤC**

[TÓM TẮT i](#_Toc183825419)

[LỜI NÓI ĐẦU ii](#_Toc183825420)

[CAM ĐOAN iii](#_Toc183825421)

[MỤC LỤC iv](#_Toc183825422)

[DANH SÁCH CÁC BẢNG, HÌNH VẼ xii](#_Toc183825423)

[DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT xv](#_Toc183825424)

[Chương 1: TỔNG QUAN VỀ ĐỀ TÀI 1](#_Toc183825425)

**[1.1.](#_Toc183825426)****[Lý do chọn đề tài](#_Toc183825426)** [1](#_Toc183825426)

**[1.2.](#_Toc183825427)****[Mục tiêu đề tài](#_Toc183825427)** [2](#_Toc183825427)

**[1.3.](#_Toc183825428)****[Đối tượng và phạm vi nghiên cứu](#_Toc183825428)** [2](#_Toc183825428)

***[1.3.1.](#_Toc183825429)******[Đối tượng nghiên cứu](#_Toc183825429)*** [2](#_Toc183825429)

***[1.3.2.](#_Toc183825430)******[Phạm vi nghiên cứu](#_Toc183825430)*** [3](#_Toc183825430)

**[1.4.](#_Toc183825431)****[Phương pháp nghiên cứu](#_Toc183825431)** [4](#_Toc183825431)

**[1.5.](#_Toc183825432)****[Cấu trúc báo cáo](#_Toc183825432)** [4](#_Toc183825432)

[Chương 2: CƠ SỞ LÝ THUYẾT 5](#_Toc183825433)

**[2.1.](#_Toc183825434)****[Giới thiệu về ngôn ngữ Python](#_Toc183825434)** [5](#_Toc183825434)

***[2.1.1. Khái niệm](#_Toc183825435)*** [5](#_Toc183825435)

[Hình 2. 1: Logo Python 5](#_Toc183825436)

***[2.1.2.](#_Toc183825437)******[Ưu điểm](#_Toc183825437)*** [5](#_Toc183825437)

***[2.1.3.](#_Toc183825439)******[Nhược điểm](#_Toc183825439)*** [6](#_Toc183825439)

**[2.2.](#_Toc183825440)****[Giới thiệu về framework FastAPI](#_Toc183825440)** [7](#_Toc183825440)

***[2.2.1.](#_Toc183825441)******[Khái niệm](#_Toc183825441)*** [7](#_Toc183825441)

[Hình 2. 2: Logo FastAPI 8](#_Toc183825442)

***[2.2.2.](#_Toc183825443)******[Ưu điểm](#_Toc183825443)*** [8](#_Toc183825443)

***[2.2.3.](#_Toc183825444)******[Nhược điểm](#_Toc183825444)*** [8](#_Toc183825444)

**[2.3.](#_Toc183825445)****[Giới thiệu về framework ReactJS](#_Toc183825445)** [9](#_Toc183825445)

***[2.3.1.](#_Toc183825446)******[Khái niệm](#_Toc183825446)*** [9](#_Toc183825446)

***[2.3.2.](#_Toc183825447)******[Ưu điểm](#_Toc183825447)*** [9](#_Toc183825447)

***[2.3.3.](#_Toc183825448)******[Nhược điểm](#_Toc183825448)*** [10](#_Toc183825448)

**[2.4.](#_Toc183825449)****[Giới thiệu về Uvicorn](#_Toc183825449)** [10](#_Toc183825449)

***[2.4.1.](#_Toc183825450)******[Khái niệm](#_Toc183825450)*** [10](#_Toc183825450)

***[2.4.2.](#_Toc183825451)******[Ưu điểm](#_Toc183825451)*** [11](#_Toc183825451)

***[2.4.3.](#_Toc183825452)******[Nhược điểm](#_Toc183825452)*** [11](#_Toc183825452)

**[2.5.](#_Toc183825453)****[Giới thiệu về Docker](#_Toc183825453)** [12](#_Toc183825453)

***[2.5.1.](#_Toc183825454)******[Khái niệm](#_Toc183825454)*** [12](#_Toc183825454)

***[2.5.2.](#_Toc183825455)******[Ưu điểm](#_Toc183825455)*** [12](#_Toc183825455)

***[2.5.3.](#_Toc183825456)******[Nhược điểm](#_Toc183825456)*** [13](#_Toc183825456)

**[2.6.](#_Toc183825457)****[Giới thiệu về Github](#_Toc183825457)** [13](#_Toc183825457)

***[2.6.1.](#_Toc183825458)******[Khái niệm](#_Toc183825458)*** [13](#_Toc183825458)

[Hình 2. 4: Logo GitHub 14](#_Toc183825459)

***[2.6.2.](#_Toc183825460)******[Ưu điểm](#_Toc183825460)*** [14](#_Toc183825460)

***[2.6.3.](#_Toc183825461)******[Nhược điểm](#_Toc183825461)*** [15](#_Toc183825461)

**[2.7.](#_Toc183825462)****[Giới thiệu về Celery](#_Toc183825462)** [15](#_Toc183825462)

***[2.7.1.](#_Toc183825463)******[Khái niệm](#_Toc183825463)*** [15](#_Toc183825463)

[Hình 2. 5: Logo Celery 16](#_Toc183825464)

***[2.7.2.](#_Toc183825465)******[Ưu điểm](#_Toc183825465)*** [16](#_Toc183825465)

***[2.7.3.](#_Toc183825466)******[Nhược điểm](#_Toc183825466)*** [16](#_Toc183825466)

**[2.8.](#_Toc183825467)****[Giới thiệu về Pusher](#_Toc183825467)** [17](#_Toc183825467)

***[2.8.1.](#_Toc183825468)******[Khái niệm](#_Toc183825468)*** [17](#_Toc183825468)

***[2.8.2.](#_Toc183825469)******[Ưu điểm](#_Toc183825469)*** [17](#_Toc183825469)

***[2.8.3.](#_Toc183825470)******[Nhược điểm](#_Toc183825470)*** [18](#_Toc183825470)

**[2.9.](#_Toc183825471)****[Giới thiệu về Open Policy Agent](#_Toc183825471)** [19](#_Toc183825471)

**[2.9.1.](#_Toc183825472)*****[Khái niệm](#_Toc183825472)*** [19](#_Toc183825472)

[Hình 2. 6: Logo Open Policy Agent 19](#_Toc183825473)

***[2.9.2.](#_Toc183825474)******[Ưu điểm](#_Toc183825474)*** [19](#_Toc183825474)

***[2.9.3.](#_Toc183825475)******[Nhược điểm](#_Toc183825475)*** [19](#_Toc183825475)

**[2.10.](#_Toc183825476)****[Giới thiệu về Json Web Token](#_Toc183825476)** [20](#_Toc183825476)

***[2.10.1.](#_Toc183825477)******[Khái niệm](#_Toc183825477)*** [20](#_Toc183825477)

***[2.10.2.](#_Toc183825478)******[Ưu điểm](#_Toc183825478)*** [20](#_Toc183825478)

***[2.10.3.](#_Toc183825479)******[Nhược điểm](#_Toc183825479)*** [21](#_Toc183825479)

[Chương 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 22](#_Toc183825480)

**[3.1. Xác định yêu cầu](#_Toc183825481)** [22](#_Toc183825481)

**[3.2. Phân tích yêu cầu](#_Toc183825482)** [22](#_Toc183825482)

***[3.2.1.](#_Toc183825483)******[Chức năng Đăng ký( 3 role)](#_Toc183825483)*** [22](#_Toc183825483)

***[3.2.2.](#_Toc183825484)******[Chức năng Đăng nhập(3 role)](#_Toc183825484)*** [22](#_Toc183825484)

***[3.2.3.](#_Toc183825489)******[Chức năng Đặt lịch đến lễ tân (role bệnh nhân)](#_Toc183825489)*** [23](#_Toc183825489)

***[3.2.4.](#_Toc183825490)******[Chức năng Xem lịch của bản thân (role bệnh nhân)](#_Toc183825490)*** [23](#_Toc183825490)

***[3.2.5.](#_Toc183825491)******[Chức năng Quản lý thanh toán của bản thân (role bệnh nhân)](#_Toc183825491)*** [23](#_Toc183825491)

***[3.2.6.](#_Toc183825492)******[Chức năng Thanh toán online (role bệnh nhân)](#_Toc183825492)*** [23](#_Toc183825492)

***[3.2.7.](#_Toc183825493)******[Chức năng Đặt lịch đến bác sĩ (role lễ tân)](#_Toc183825493)*** [23](#_Toc183825493)

***[3.2.8.](#_Toc183825494)******[Chức năng Quản lý thanh toán lễ tân (role lễ tân)](#_Toc183825494)*** [24](#_Toc183825494)

***[3.2.9.](#_Toc183825495)******[Chức năng Thanh Toán cho bệnh nhân (role lễ tân)](#_Toc183825495)*** [24](#_Toc183825495)

***[3.2.10.](#_Toc183825496)******[Chức năng Quản lý lịch hẹn của bệnh nhân (role lễ tân)](#_Toc183825496)*** [24](#_Toc183825496)

***[3.2.11.](#_Toc183825497)******[Chức năng Quản lý lịch hẹn (role bác sĩ)](#_Toc183825497)*** [24](#_Toc183825497)

***[3.2.12.](#_Toc183825498)******[Chức năng Đặt lịch đến bác sĩ (role bác sĩ)](#_Toc183825498)*** [24](#_Toc183825498)

***[3.2.13.](#_Toc183825499)******[Chức năng Tạo hồ sơ bệnh án (role bác sĩ)](#_Toc183825499)*** [24](#_Toc183825499)

***[3.2.14.](#_Toc183825500)******[Chức năng Nhận thông báo gửi cho bản thân (3 role)](#_Toc183825500)*** [24](#_Toc183825500)

***[3.2.15.](#_Toc183825501)******[Chức năng Nhận thông báo thời gian thực từ pusher (3 role)](#_Toc183825501)*** [24](#_Toc183825501)

***[3.2.16.](#_Toc183825502)******[Chức năng gửi thông báo (3 role)](#_Toc183825502)*** [24](#_Toc183825502)

**[3.3.](#_Toc183825503)****[Mô hình hóa yêu cầu](#_Toc183825503)** [24](#_Toc183825503)

***[3.3.1.](#_Toc183825504)******[Chức năng Đăng ký](#_Toc183825504)*** [24](#_Toc183825504)

***[3.3.2.](#_Toc183825513)******[Chức năng Đặt lịch khám](#_Toc183825513)*** [28](#_Toc183825513)

***[3.3.3.](#_Toc183825518)******[Chức năng Thông báo](#_Toc183825518)*** [32](#_Toc183825518)

***[3.3.4.](#_Toc183825523)******[Chức năng Quản lý và đặt lịch khám cho bệnh nhân](#_Toc183825523)*** [34](#_Toc183825523)

***[3.3.5.](#_Toc183825528)******[Chức năng Khám bệnh](#_Toc183825528)*** [36](#_Toc183825528)

***[3.3.6.](#_Toc183825533)******[Chức năng Thanh toán](#_Toc183825533)*** [39](#_Toc183825533)

**[3.4](#_Toc183825539)****[Thiết kế cơ sơ dữ liệu](#_Toc183825539)** [43](#_Toc183825539)

**[3.4.1 Mô hình thực thể liên kết](#_Toc183825540)** [43](#_Toc183825540)

**[3.4.2 Mô hình dữ liệu quan hệ](#_Toc183825542)** [43](#_Toc183825542)

**[3.4.3 Danh mục các bảng](#_Toc183825543)** [44](#_Toc183825543)

**[3.5](#_Toc183825550)****[Kết quả đạt được](#_Toc183825550)** [47](#_Toc183825550)

**[3.5.1](#_Toc183825551)****[Giao diện người dùng](#_Toc183825551)** [47](#_Toc183825551)

**[3.5.2](#_Toc183825565)****[Giao diện lễ tân](#_Toc183825565)** [53](#_Toc183825565)

**[3.5.3](#_Toc183825571)****[Giao diện bác sĩ](#_Toc183825571)** [55](#_Toc183825571)

[Chương 4: XÂY DỰNG WEBSITE QUẢN LÝ HỒ SƠ BỆNH ÁN 60](#_Toc183825579)

**[4.1. Môi trường phát triển và công cụ lập trình](#_Toc183825580)** [60](#_Toc183825580)

***[4.1.1. Cài đặt python](#_Toc183825581)*** [60](#_Toc183825581)

***[4.1.2. Cài đặt Pycharm](#_Toc183825582)*** [60](#_Toc183825582)

***[4.1.3. Cài đặt ReactJs](#_Toc183825583)*** [61](#_Toc183825583)

***[4.1.4. Cài đặt PostgreSQL](#_Toc183825584)*** [62](#_Toc183825584)

**[4.2. Xây dựng API cho hệ thống](#_Toc183825585)** [62](#_Toc183825585)

[KẾT LUẬN 75](#_Toc183825586)

[TÀI LIỆU THAM KHẢO 78](#_Toc183825587)

# **DANH SÁCH CÁC BẢNG**

[Bảng 3. 1: Use Case đăng ký 25](#_Toc183825506)

[Bảng 3. 2: Use Case dòng chính đăng ký 25](#_Toc183825507)

[Bảng 3. 3: Use Case dòng chính đăng nhập 27](#_Toc183825510)

[Bảng 3. 4: Use Case dòng chính đăng nhập 27](#_Toc183825511)

[Bảng 3. 5: Use Case đặt lịch khám 28](#_Toc183825515)

[Bảng 3. 6: Use Case dòng chính đặt lịch khám 30](#_Toc183825516)

[Bảng 3. 7: Use Case thông báo 32](#_Toc183825520)

[Bảng 3. 8: Use Case Dòng chính thông báo 32](#_Toc183825521)

[Bảng 3. 9: Use Case Quản lý và đặt lịch khám 34](#_Toc183825525)

[Bảng 3. 10: Use Case Dòng chính Quản lý và đặt lịch khám 36](#_Toc183825526)

[Bảng 3. 11: Use Case Khám bệnh 37](#_Toc183825530)

[Bảng 3. 12: Use Case Dòng chính Khám bệnh 38](#_Toc183825531)

[Bảng 3.13 Mô tả Use Case thêm thanh toán 40](#_Toc183825535)

[Bảng 3.14 Dòng chính Use Case thanh toán 40](#_Toc183825536)

[Bảng 3.15 Mô tả các bảng cơ sở dữ liệu 44](#_Toc183825544)

[Bảng 3.16 Mô tả bảng User 44](#_Toc183825545)

[Bảng 3.17 Mô tả bảng Payment 45](#_Toc183825546)

[Bảng 3.18 Mô tả bảng Appointment 45](#_Toc183825547)

[Bảng 3.19 Mô tả bảng MedicalRecordDoctor 46](#_Toc183825548)

[Bảng 3.20 Mô tả bảng LabTest 46](#_Toc183825549)

# **DANH SÁCH CÁC HÌNH VẼ**

[Hình 3. 1: Use Case đăng ký 25](#_Toc183825505)

[Hình 3. 2: Tuần Tự đăng ký 26](#_Toc183825508)

[Hình 3. 3: Use Case đăng nhập 26](#_Toc183825509)

[Hình 3. 4: Tuần Tự đăng nhập 28](#_Toc183825512)

[Hình 3. 5: Use Case đặt lịch khám 28](#_Toc183825514)

[Hình 3. 6: Tuần Tự đặt lịch khám 30](#_Toc183825517)

[Hình 3. 7: Tuần Tự Use Case thông báo 32](#_Toc183825519)

[Hình 3. 8: Tuần Tự chức năng thông báo 34](#_Toc183825522)

[Hình 3. 9: Use Case Quản lý đặt lịch 34](#_Toc183825524)

[Hình 3. 10: Tuần Tự đặt lịch khám 36](#_Toc183825527)

[Hình 3. 11: Use case Khám bệnh 37](#_Toc183825529)

[Hình 3. 12: Tuần Tự khám bệnh 39](#_Toc183825532)

[Hình 3.13 Biểu đồ Use Case thanh toán 40](#_Toc183825534)

[Hình 3.14 Biểu đồ tuần tự thanh toán khoá học 41](#_Toc183825537)

[Hình 3.15 Mô hình ERD 44](#_Toc183825541)

[Hình 3.16. Giao diện trang đăng nhập 47](#_Toc183825552)

[Hình 3.17. Giao diện trang đăng ký 48](#_Toc183825553)

[Hình 3.18. Giao diện trang đăng ký 48](#_Toc183825554)

[Hình 3.19. Giao diện trang chủ bệnh nhân 48](#_Toc183825555)

[Hình 3.20. Giao diện trang đặt lịch khám bệnh nhân 49](#_Toc183825556)

[Hình 3.21. Giao diện trang xem lịch khám bệnh nhân 49](#_Toc183825557)

[Hình 3.22. Giao diện trang thanh toán bệnh nhân 50](#_Toc183825558)

[Hình 3.23. Giao diện trang tất cả thanh toán bệnh nhân 50](#_Toc183825559)

[Hình 3.24. Giao diện trang tất cả thanh toán bệnh nhân 51](#_Toc183825560)

[Hình 3.25. Giao diện trang nhập thông tin zalo pay 51](#_Toc183825561)

[Hình 3.26. Giao diện thanh toán zalo pay thành công 52](#_Toc183825562)

[Hình 3.27. Giao diện thanh toán thành công 52](#_Toc183825563)

[Hình 3.28. Giao diện chức năng thông báo 53](#_Toc183825564)

[Hình 3.29. Giao diện chủ lễ tân 53](#_Toc183825566)

[Hình 3.30. Giao diện lịch hẹn Lễ Tân 54](#_Toc183825567)

[Hình 3.31. Giao diện chi tiết lịch khám 54](#_Toc183825568)

[Hình 3.32. Giao diện thanh toán lễ tân 55](#_Toc183825569)

[Hình 3.33. Giao diện chi tiết thanh toán lễ tân 55](#_Toc183825570)

[Hình 3.34. Giao diện trang chủ bác sĩ 56](#_Toc183825572)

[Hình 3.35. Giao diện trang xem lịch hẹn bác sĩ 56](#_Toc183825573)

[Hình 3.36. Giao diện trang tạo hồ sơ bệnh án bác sĩ 57](#_Toc183825574)

[Hình 3.37. Giao diện trang tạo lịch khám tiếp theo 57](#_Toc183825575)

[Hình 3.38. Giao diện trang lịch khám tiếp theo 58](#_Toc183825576)

[Hình 3.39. Giao diện trang thông tin cá nhân 58](#_Toc183825577)

[Hình 3.40. Giao diện trang thông báo thời gian thực 59](#_Toc183825578)

# **DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| KÝ HIỆU | TÊN GỌI | DIỄN GIẢI |
| API | Application Programming Interface | Giao diện lập trình ứng dụng, cho phép các phần mềm khác nhau giao tiếp và trao đổi dữ liệu với nhau. |
| HTTP | Hypertext Transfer Protocol | Giao thức truyền tải siêu văn bản, được sử dụng để trao đổi dữ liệu giữa trình duyệt và máy chủ web. |
| UI | User Interface | Giao diện người dùng, phần hiển thị mà người dùng tương tác trực tiếp trong một phần mềm hoặc ứng dụng. |
| OPA | Open Policy Agent | Một công cụ mã nguồn mở giúp thực thi và quản lý chính sách kiểm soát truy cập (authorization) độc lập với ngôn ngữ hoặc dịch vụ. |
| CI/CD | Continuous Integration/Continuous Deployment | Quy trình tự động hóa tích hợp và triển khai liên tục trong phát triển phần mềm. |
| MVC | Model-View-Controller | Kiến trúc phần mềm chia ứng dụng thành ba phần: Model (dữ liệu), View (giao diện), và Controller (xử lý logic). |
| JS | JavaScript | Ngôn ngữ lập trình phía client-side (và server-side) dùng để tạo ra các tính năng động cho trang web. |
| JSX | JavaScript XML | Một cú pháp mở rộng của JavaScript, thường được sử dụng trong React để mô tả cấu trúc giao diện giống như XML hoặc HTML. |
| HTML | Hypertext Markup Language | Ngôn ngữ đánh dấu siêu văn bản, được sử dụng để xây dựng cấu trúc cơ bản của một trang web. |
| DOM | |  | | --- | |  |  |  | | --- | | Document Object Model | | Mô hình đối tượng tài liệu, một giao diện lập trình mà các ngôn ngữ như JavaScript sử dụng để thao tác nội dung của trang web được xây dựng bằng HTML hoặc XML. |

# **Chương 1: TỔNG QUAN VỀ ĐỀ TÀI**

* 1. **Lý do chọn đề tài**

- Trong bối cảnh chuyển đổi số đang lan tỏa mạnh mẽ ở nhiều lĩnh vực, ngành y tế cũng không nằm ngoài xu thế này. Tuy nhiên, thực tế cho thấy, việc quản lý hồ sơ bệnh án tại nhiều cơ sở y tế vẫn còn dựa trên phương pháp truyền thống, gây ra không ít khó khăn trong việc lưu trữ, tìm kiếm và chia sẻ thông tin. Những hạn chế này không chỉ làm giảm hiệu quả làm việc của đội ngũ y tế mà còn ảnh hưởng đến chất lượng dịch vụ chăm sóc sức khỏe, đặc biệt khi các yêu cầu về tính chính xác và tốc độ xử lý thông tin ngày càng gia tăng.

- Hơn thế nữa, nhu cầu kết nối và giao tiếp nhanh chóng giữa bác sĩ và bệnh nhân ngày càng trở nên quan trọng. Không phải lúc nào bệnh nhân cũng có thể đến trực tiếp cơ sở y tế, đặc biệt trong các tình huống khẩn cấp hoặc những trường hợp ở xa. Vì vậy, việc xây dựng một hệ thống quản lý hồ sơ bệnh án kết hợp các công cụ hỗ trợ giao tiếp từ xa không chỉ là giải pháp nâng cao trải nghiệm của người bệnh mà còn góp phần giảm áp lực cho hệ thống y tế truyền thống.

- Để đáp ứng các yêu cầu này, việc ứng dụng các công nghệ hiện đại như FastAPI và ReactJS vào hệ thống quản lý hồ sơ bệnh án mang lại nhiều lợi ích thiết thực. Những công nghệ này không chỉ giúp xây dựng nền tảng ổn định, linh hoạt mà còn tối ưu hóa khả năng lưu trữ và xử lý dữ liệu. Đồng thời, việc tích hợp các công cụ giao tiếp như nhắn tin hay video call tạo điều kiện thuận lợi cho việc tư vấn và theo dõi từ xa, giúp bác sĩ và bệnh nhân duy trì sự kết nối mọi lúc, mọi nơi.

- Tại Việt Nam, dù đã có nhiều nỗ lực trong việc cải thiện hạ tầng y tế, nhưng việc chuyển đổi số tại các cơ sở y tế vẫn còn gặp nhiều rào cản. Với đề tài này, chúng tôi không chỉ hướng tới việc xây dựng một hệ thống quản lý hiệu quả mà còn kỳ vọng góp phần thúc đẩy sự phát triển của ngành y tế, mang lại những cải tiến tích cực trong quy trình chăm sóc sức khỏe và phục vụ cộng đồng tốt hơn.

* 1. **Mục tiêu đề tài**

- Đề tài này hướng đến việc phát triển một hệ thống quản lý hồ sơ bệnh án nhằm nâng cao hiệu quả trong việc quản lý hồ sơ bệnh án, kê đơn thuốc, và hỗ trợ giao tiếp giữa bác sĩ và bệnh nhân. Mục tiêu cụ thể của đề tài bao gồm:

- Phát triển hệ thống quản lý hồ sơ bệnh án: Xây dựng nền tảng số để quản lý hồ sơ bệnh án, bao gồm lưu trữ thông tin bệnh nhân, cập nhật tình trạng sức khỏe, và theo dõi tiến trình điều trị. Hệ thống sẽ giúp bác sĩ và nhân viên y tế quản lý thông tin bệnh án một cách hiệu quả.

- Cải thiện giao tiếp giữa bác sĩ và bệnh nhân: Phát triển các công cụ nhắn tin và video call để nâng cao khả năng giao tiếp từ xa. Điều này sẽ giúp bác sĩ và bệnh nhân tương tác hiệu quả hơn, đồng thời tạo điều kiện thuận lợi cho việc tư vấn và theo dõi điều trị từ xa.

- Ứng dụng và triển khai trong cơ sở y tế: Hệ thống sẽ được triển khai trong các bệnh viện, phòng khám, và cơ sở y tế khác để tối ưu hóa quy trình quản lý hồ sơ bệnh án, cải thiện chất lượng dịch vụ y tế và tăng cường khả năng tương tác giữa bác sĩ và bệnh nhân.

* 1. **Đối tượng và phạm vi nghiên cứu**
     1. ***Đối tượng nghiên cứu***

- Đối tượng nghiên cứu trong đề tài này là hệ thống quản lý hồ sơ bệnh án và các công cụ hỗ trợ khám chữa bệnh được áp dụng tại các cơ sở y tế, bao gồm bệnh viện và phòng khám. Cụ thể, đối tượng nghiên cứu sẽ bao gồm:

- Bệnh nhân: Những người sử dụng hệ thống để quản lý hồ sơ sức khỏe, đặt lịch khám, nhận tư vấn từ bác sĩ, và thực hiện các giao dịch thanh toán trực tuyến.

- Bác sĩ và nhân viên y tế: Các đối tượng sử dụng hệ thống để lưu trữ, truy cập và cập nhật thông tin bệnh án, kê đơn thuốc, cũng như hỗ trợ bệnh nhân qua các kênh giao tiếp như nhắn tin

- Quản lý cơ sở y tế: Những người có nhiệm vụ quản lý và giám sát việc triển khai và vận hành hệ thống trong bệnh viện hoặc phòng khám, đảm bảo hệ thống hoạt động hiệu quả.

- Hệ thống này sẽ sử dụng các công nghệ như FastAPI cho backend, ReactJS cho frontend, PostgreSQL cho cơ sở dữ liệu và các công cụ bảo mật để đảm bảo an toàn thông tin người dùng.

* + 1. ***Phạm vi nghiên cứu***

- Phạm vi không gian:Nghiên cứu được triển khai và thử nghiệm tại các cơ sở y tế vừa và nhỏ ở Việt Nam, bao gồm bệnh viện, phòng khám tư nhân và các trung tâm y tế cộng đồng. Đây là những nơi thường gặp nhiều khó khăn trong việc quản lý hồ sơ bệnh án do nguồn lực hạn chế và hạ tầng công nghệ chưa phát triển đồng bộ. Hệ thống sẽ được thiết kế để phù hợp với môi trường làm việc thực tế tại các cơ sở này, đảm bảo khả năng triển khai và ứng dụng hiệu quả trong điều kiện hiện có.

- Khách thể nghiên cứu: Khách thể nghiên cứu bao gồm các quy trình và hệ thống hiện hành tại các cơ sở y tế, đặc biệt là tại các bệnh viện và phòng khám vừa và nhỏ. Nghiên cứu tập trung vào việc cải thiện hiệu quả quản lý hồ sơ bệnh án, tối ưu hóa giao tiếp giữa bác sĩ và bệnh nhân, và hỗ trợ các hoạt động chăm sóc sức khỏe bằng công nghệ số.

- Các chức năng chính:

* Quản lý hồ sơ bệnh án: Lưu trữ, cập nhật và theo dõi toàn bộ thông tin bệnh nhân
* Đặt lịch khám: đặt lịch khám trực tuyến, cho phép bệnh nhân lựa chọn thời gian phù hợp và bác sĩ phù hợp
* Nhắn tin tư vấn và hỗ trợ khám bệnh: Cung cấp công cụ nhắn tin để bác sĩ và bệnh nhân có thể trao đổi thông tin về tình trạng sức khỏe, yêu cầu tư vấn trước và sau khi khám, đặc biệt trong trường hợp bệnh nhân không thể trực tiếp đến cơ sở y tế.
* Thanh toán online: Tích hợp hệ thống thanh toán trực tuyến qua ZaloPay
* Bảo mật và quản lý dữ liệu: Đảm bảo an toàn thông tin bệnh nhân xác thực người dùng, và phân quyền truy cập.

- Công nghệ nghiên cứu:

* FastAPI: Phát triển backend, xử lý các API
* ReactJS: Xây dựng giao diện người dùng thân thiện
* PostgreSQL: Quản lý dữ liệu bệnh nhân, hồ sơ bệnh án, và các thông tin liên quan
* Pusher: Hỗ trợ tính năng thông báo cho người dùng
  1. **Phương pháp nghiên cứu**

- Nghiên cứu tài liệu: Thu thập và phân tích các tài liệu, báo cáo liên quan đến các hệ thống quản lý bệnh viện và ứng dụng y tế trực tuyến, từ đó xây dựng cơ sở lý thuyết cho hệ thống

- Nghiên cứu thực nghiệm: Phát triển và kiểm thử các chức năng của hệ thống như đặt lịch khám, tư vấn nhắn tin, kê đơn, và thanh toán trực tuyến, đảm bảo tính hiệu quả và chính xác.

- Mô phỏng: Kiểm tra các kịch bản sử dụng hệ thống qua mô phỏng để đánh giá tính khả thi và hiệu suất.

- Phân tích hệ thống: Đánh giá các yêu cầu và thiết kế cơ sở dữ liệu, tối ưu quy trình làm việc của hệ thống.

- Lập trình và thiết kế**:** Sử dụng các công nghệ như FastAPI, ReactJS, và PostgreSQL để phát triển hệ thống.

* 1. **Cấu trúc báo cáo**

- Nội dung báo cáo được tổ chức thành bốn chương chính với các nội dung cụ thể như sau:

* Chương 1: Giới thiệu tổng quan về đề tài – Trình bày lý do chọn đề tài, mục tiêu nghiên cứu và tầm quan trọng của hệ thống quản lý hồ sơ bệnh án trong ngành y tế.
* Chương 2: Cơ sở lý thuyết – Cung cấp nền tảng lý thuyết về hệ thống bệnh án điện tử, công nghệ sử dụng và xu hướng ứng dụng công nghệ trong y tế.
* Chương 3: Phân tích thiết kế hệ thống thông tin – Phân tích yêu cầu và thiết kế cấu trúc hệ thống, các chức năng chính, và cơ sở dữ liệu cần thiết.
* Chương 4: Xây dựng Website Quản lý Hồ Sơ Bệnh Án – Triển khai xây dựng website với các công nghệ sử dụng, phát triển giao diện người dùng và các chức năng backend.

# **Chương 2: CƠ SỞ LÝ THUYẾT**

## **Giới thiệu về ngôn ngữ Python**

***2.1.1. Khái niệm***

- Python là [ngôn ngữ lập trình bậc cao](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh_b%E1%BA%ADc_cao) đa năng. Triết lý thiết kế của nó nhấn mạnh khả năng đọc mã bằng cách sử dụng thụt lề đáng kể.[1][https://vi.wikipedia.org/wiki/Python\_(ngôn\_ngữ\_lập\_trình)](https://vi.wikipedia.org/wiki/Python_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)#cite_note-AutoNT-7-32)

- Python có [kiểu động](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_ki%E1%BB%83u) và [thu gom rác](https://vi.wikipedia.org/wiki/Thu_gom_r%C3%A1c_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)). Ngôn ngữ này hỗ trợ nhiều mô hình lập trình, bao gồm [lập trình cấu trúc](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_c%E1%BA%A5u_tr%C3%BAc) (đặc biệt là [lập trình thủ tục](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_th%E1%BB%A7_t%E1%BB%A5c)), [lập trình hướng đối tượng](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) và lập trình chức năng. Nó thường được mô tả là ngôn ngữ "bao gồm pin" do có thư viện tiêu chuẩn toàn diện.



Hình 2. 1: Logo Python

* + 1. ***Ưu điểm***

- Dễ đọc và dễ học: Python là một ngôn ngữ đơn giản để đọc và học. Nó không có cú pháp phức tạp như các ngôn ngữ cấp cao khác như C hoặc C ++. Nhờ ít phức tạp hơn, Python cho phép bạn suy nghĩ rõ ràng hơn và tập trung vào việc xây dựng logic.

- Giảm chi phí bảo trì: Do tính đơn giản của nó, Python giúp bảo trì ứng dụng dễ dàng hơn và do đó, giảm chi phí liên quan, đây là một lợi thế lớn.

- Tránh tác hại từ lỗi phần mềm: Python không để lỗi mã xảy ra phân đoạn trong ứng dụng. Do đó, nó được coi là một ngôn ngữ ưa thích.

- Khả năng ứng dụng rộng rãi: Một tính năng thiết yếu khác của ngôn ngữ này là nó có thể áp dụng rộng rãi. Các kỹ sư, nhà khoa học và nhà toán học sử dụng rộng rãi nó.

- Quản lý bộ nhớ: Python có một thư viện rộng lớn với khả năng quản lý bộ nhớ, điều này làm cho nó nổi bật so với các ngôn ngữ lập trình khác. Nó bao gồm một heap riêng chứa tất cả các đối tượng và cấu trúc dữ liệu Python, một trình quản lý bộ nhớ tích hợp để duy trì heap riêng tư này.

- Đơn giản và nhanh chóng: Cộng đồng Python cung cấp hỗ trợ nhanh chóng và thiết thực cho người dùng cũng như khả năng thích ứng nhanh của mã. Một số chuyên gia thích đặt biệt danh cho Python là “ngôn ngữ sẵn sàng để chạy” vì nó chỉ yêu cầu mã đơn giản để được thực thi. Nâng cao và kiểm tra mã thoải mái hơn nhiều với Python.

- Mã hóa không đồng bộ: Mã hóa không đồng bộ sử dụng một vòng lặp sự kiện duy nhất để hoàn thành công việc trong những khoảng thời gian nhỏ. Python rất hữu ích để viết mã không đồng bộ vì nó dễ viết và dễ bảo trì. Nó không yêu cầu bất kỳ nội dung nghiên cứu phức tạp, bế tắc hoặc bất kỳ sự phức tạp nào khác.

- Tích hợp với các ngôn ngữ khác: Python có các thư viện như Cython và Jython, cho phép tích hợp với các ngôn ngữ khác như C, C ++ và Java để phát triển đa nền tảng. Đây là một trong những đặc quyền chính của Python vì không có ngôn ngữ nào là hoàn hảo và đôi khi sự phát triển đòi hỏi các chức năng ngôn ngữ đa dạng.

- Tích hợp ứng dụng doanh nghiệp: Python là lựa chọn tốt nhất cho Tích hợp ứng dụng doanh nghiệp, cung cấp các tính năng kiểm soát quy trình đáng tin cậy và thực hiện các định dạng, giao thức dữ liệu internet. Hơn nữa, Python giúp người dùng xử lý các ngôn ngữ đánh dấu, thực thi thông qua cùng một mã byte trên các hệ điều hành nâng cao và có thể được sử dụng như một ngôn ngữ kịch bản.

* + 1. ***Nhược điểm***

- Tốc độ thực thi chậm: Python là một ngôn ngữ thông dịch, có nghĩa là nó hoạt động với trình thông dịch, không phải với trình biên dịch. Do đó, nó thực thi tương đối chậm hơn C, C ++, Java và nhiều ngôn ngữ khác.

- Tiêu thụ bộ nhớ lớn: Các cấu trúc của Python đòi hỏi nhiều không gian bộ nhớ hơn. Ngôn ngữ này không thích hợp để sử dụng cho sự phát triển trong điều kiện bộ nhớ hạn chế.

- Không thích hợp cho phát triển trò chơi và thiết bị di động: Python chủ yếu được sử dụng trong phát triển máy tính để bàn và web phía máy chủ. Nó không được coi là lý tưởng để phát triển ứng dụng di động và phát triển trò chơi do tiêu tốn nhiều bộ nhớ hơn và tốc độ xử lý chậm so với các ngôn ngữ lập trình khác.

- Hạn chế của Nhà phát triển: Một khi nhà phát triển đã quen với sự dễ dàng và đơn giản của ngôn ngữ này, họ sẽ khó sử dụng các ngôn ngữ khác.

- Phát hiện lỗi trong mã: Vì Python được thực thi thông qua trình thông dịch thay vì trình biên dịch, nên không thể phát hiện lỗi trong quá trình biên dịch và điều đó không tốt cho các nhà phát triển.

- Quyền truy cập cơ sở dữ liệu: Python được coi là không an toàn cao và có nguy cơ bảo mật. Có một số hạn chế khi sử dụng Python để truy cập cơ sở dữ liệu. So với các công nghệ phổ biến khác như JDBC và ODBC, lớp truy cập cơ sở dữ liệu Python hơi kém phát triển và sơ khai.

- Hạn chế thiết kế: Một trong những vấn đề quan trọng của Python là các hạn chế về thiết kế của nó.

- Khó kiểm tra: Vì nó là một ngôn ngữ dựa trên trình thông dịch, rất khó để chạy các bài kiểm tra trên mã được viết bằng Python. Tất cả các lỗi chỉ xuất hiện trong thời gian chạy, điều này khiến việc kiểm tra các đoạn mã được viết bằng Python rất khó khăn.

* 1. **Giới thiệu về framework FastAPI**
     1. ***Khái niệm***

- FastAPI là một framework web hiệu suất cao dùng để xây dựng các API dịch vụ HTTP trong Python 3.8 trở lên [2]. Framework này sử dụng Pydantic và các gợi ý kiểu (type hints) để xác thực, tuần tự hóa và giải tuần tự dữ liệu. FastAPI cũng tự động tạo tài liệu OpenAPI cho các API được xây dựng bằng nó [3]. FastAPI được phát hành lần đầu vào năm 2018.



Hình 2. 2: Logo FastAPI

* + 1. ***Ưu điểm***

- High performance: FastAPI là một khung phát triển hiện đại dựa trên lập trình không đồng bộ, nổi tiếng với hiệu suất xuất sắc và thời gian trễ thấp.

- Automatic documentation generation: FastAPI có thể tạo tài liệu tương tác dựa trên mã của bạn, nâng cao hiệu quả phát triển và kiểm thử API.

- Type annotation support: FastAPI hỗ trợ việc sử dụng chú thích kiểu để tăng cường tính đọc và bảo trì mã.

- Rapid development: FastAPI vượt trội trong việc phát triển nhanh chóng, làm cho nó phù hợp cho việc xây dựng các bản prototype, Proof of Concept (POC), và các ứng dụng có các vòng lặp nhanh.

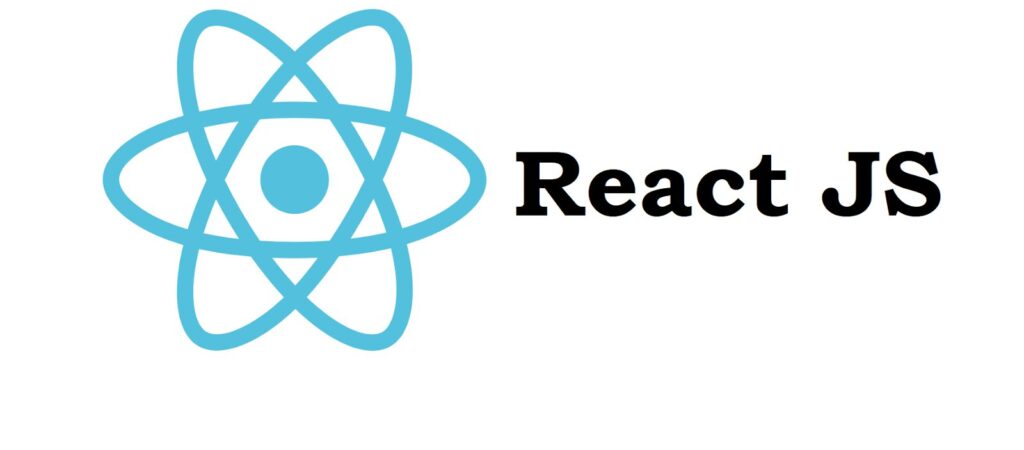
* + 1. ***Nhược điểm***

- Relatively new project: FastAPI là một dự án tương đối mới và có thể thiếu các giải pháp chín muồi và sự hỗ trợ từ cộng đồng trong một số lĩnh vực.

- Learning curve: Đối với các developer không có kinh nghiệm trong lập trình không đồng bộ, FastAPI có thể có độ cong học tập cao hơn.

* 1. **Giới thiệu về framework ReactJS**
     1. ***Khái niệm***

**- React** (hay còn được gọi là React.js hoặc ReactJS) là một [thư viện JavaScript](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_JavaScript&action=edit&redlink=1) [front-end](https://vi.wikipedia.org/wiki/Front-end_v%C3%A0_back-end) [https://vi.wikipedia.org/wiki/Phần\_mềm\_tự\_do\_nguồn\_mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do_ngu%E1%BB%93n_m%E1%BB%9F) [4]  để xây dựng [giao diện người dùng](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_ng%C6%B0%E1%BB%9Di_d%C3%B9ng) dựa trên các thành phần UI riêng lẻ. Nó được phát triển và duy trì bởi [Meta](https://vi.wikipedia.org/wiki/Meta_(c%C3%B4ng_ty)) (trước đây là Facebook) và cộng đồng các nhà phát triển và công ty cá nhân. [5][6][7] React có thể được sử dụng làm cơ sở để phát triển các ứng dụng [SPA](https://vi.wikipedia.org/w/index.php?title=SPA&action=edit&redlink=1) (Single page application), thiết bị di động hoặc ứng dụng được kết xuất bằng máy chủ với các thư viện khác như Next.js. Tuy nhiên, React chỉ hướng tới việc quản lý trạng thái và hiển thị trạng thái đó cho [DOM](https://vi.wikipedia.org/wiki/Document_Object_Model), vì vậy việc tạo ứng dụng bằng React thường yêu cầu sử dụng thêm các thư viện bổ sung để thực hiện định tuyến trang, cũng như thêm một số chức năng ở phía máy khách. [8][9]

Hình 2. 3: Logo ReactJS

* + 1. ***Ưu điểm***

- Reactjs cực kì hiệu quả: Reactjs tạo ra cho chính nó DOM ảo – nơi mà các component thực sự tồn tại trên đó. Điều này sẽ giúp cải thiện hiệu suất rất nhiều. Reactjs cũng tính toán những thay đổi nào cần cập nhật len DOM và chỉ thực hiện chúng. Điều này giúp Reactjs tránh những thao tác cần trên DOM mà nhiều chi phí.

- Reactjs giúp việc viết các đoạn code JS dễ dàng hơn: Nó dung cú pháp đặc biệt là JSX (Javascript mở rộng) cho phép ta trộn giữa code HTML và Javascript. Ta có thể them vào các đoạn HTML vào trong hàm render mà không cần phải nối chuỗi. Đây là đặc tính thú vị của Reactjs. Nó sẽ chuyển đổi các đoạn HTML thành các hàm khởi tạo đối tượng HTML bằng bộ biến đổi JSX.

- Nó có nhiều công cụ phát triển: Khi bạn bắt đầu Reactjs, đừng quên cài đặt ứng dụng mở rộng của Chrome dành cho Reactjs. Nó giúp bạn debug code dễ dàng hơn. Sau khi bạn cài đặt ứng dụng này, bạn sẽ có cái nhìn trực tiếp vào virtual DOM như thể bạn đang xem cây DOM thông thường.

- Làm việc với vấn đề test giao diện: Nó cực kì dễ để viết các test case giao diện vì virtual DOM được cài đặt hoàn toàn bằng JS.

- Hiệu năng cao đối với các ứng dụng có dữ liệu thay đổi liên tục, dễ dàng cho bảo trì và sửa lỗi.

* + 1. ***Nhược điểm***

- Reactjs chỉ phục vụ cho tầng View. React chỉ là View Library nó không phải là một MVC framework như những framework khác. Đây chỉ là thư viện của Facebook giúp render ra phần view. Vì thế React sẽ không có phần Model và Controller, mà phải kết hợp với các thư viện khác. React cũng sẽ không có 2-way binding hay là Ajax

- Tích hợp Reactjs vào các framework MVC truyền thống yêu cầu cần phải cấu hình lại.

- React khá nặng nếu so với các framework khác React có kích thước tương tương với Angular (Khoảng 35kb so với 39kb của Angular). Trong khi đó Angular là một framework hoàn chỉnh

- Khó tiếp cận cho người mới học Web

* 1. **Giới thiệu về Uvicorn**
     1. ***Khái niệm***

- Uvicorn là một máy chủ ứng dụng/máy chủ web tối giản ở mức độ thấp dành cho các framework bất đồng bộ, tuân theo đặc tả ASGI. Về mặt kỹ thuật, nó triển khai mô hình đa tiến trình với một tiến trình chính, chịu trách nhiệm quản lý một nhóm các tiến trình worker và phân phối các yêu cầu HTTP đến chúng. Số lượng các tiến trình worker được cấu hình trước, nhưng cũng có thể điều chỉnh tăng hoặc giảm trong thời gian chạy. [10]

* + 1. ***Ưu điểm***

- Hiệu năng cao: Uvicorn được thiết kế để tối ưu hóa hiệu năng, đặc biệt khi xử lý các ứng dụng bất đồng bộ (asynchronous) dựa trên Python. Nó sử dụng thư viện uvloop (dựa trên libuv) và httptools, giúp xử lý HTTP nhanh hơn nhiều so với các máy chủ đồng bộ truyền thống.

- Nhẹ và nhanh: Uvicorn rất gọn nhẹ và khởi động nhanh, phù hợp với các ứng dụng hiện đại và các microservice. Điều này giúp giảm thời gian phản hồi và cải thiện khả năng mở rộng.

- Tuân theo ASGI: Việc hỗ trợ ASGI (Asynchronous Server Gateway Interface) cho phép Uvicorn làm việc với các framework hiện đại như FastAPI và Starlette, giúp tận dụng tối đa các tính năng bất đồng bộ.

- Hỗ trợ WebSocket: Uvicorn hỗ trợ các giao thức thời gian thực như WebSocket, rất hữu ích cho các ứng dụng như chat, thông báo trực tiếp, hoặc luồng dữ liệu.

- Dễ tích hợp với Docker: Uvicorn dễ dàng tích hợp với Docker, làm cho nó trở thành một lựa chọn phổ biến trong các môi trường triển khai containerized.

- Cộng đồng mạnh mẽ: Với sự phổ biến của các framework như FastAPI, Uvicorn đã xây dựng được một cộng đồng lớn, cung cấp nhiều tài liệu và hỗ trợ.

* + 1. ***Nhược điểm***

- Quản lý tiến trình giới hạn: Uvicorn sử dụng mô hình đa tiến trình cơ bản nhưng không phức tạp như các máy chủ web khác (như Gunicorn). Điều này có thể làm giảm hiệu quả trong một số môi trường đòi hỏi khả năng quản lý tiến trình tiên tiến hơn.

- Không tối ưu cho ứng dụng đồng bộ: Nếu ứng dụng không sử dụng các hàm bất đồng bộ, Uvicorn không mang lại lợi thế hiệu suất đáng kể so với các máy chủ đồng bộ truyền thống như Gunicorn.

- Cần cấu hình cho sản xuất (production): Khi triển khai trên môi trường sản xuất, người dùng cần phải điều chỉnh các tham số (như số worker hoặc cài đặt timeout) để đạt được hiệu năng tốt nhất, điều này có thể khó khăn với người mới.

- Phụ thuộc vào Python Async: Mặc dù mạnh mẽ, nhưng Uvicorn chỉ thực sự tỏa sáng trong các ứng dụng bất đồng bộ. Nếu đội ngũ phát triển chưa quen với lập trình async, việc triển khai Uvicorn có thể tạo ra đường cong học tập lớn.

- Ít tính năng so với Gunicorn/NGINX: So với Gunicorn hoặc NGINX, Uvicorn thiếu một số tính năng tích hợp như cân bằng tải hoặc quản lý SSL, khiến nó phù hợp hơn như một máy chủ ứng dụng thay vì một máy chủ web đầy đủ chức năng.

* 1. **Giới thiệu về Docker**
     1. ***Khái niệm***

- Docker là một tập hợp các sản phẩm nền tảng như một dịch vụ (Platform as a Service - PaaS) sử dụng ảo hóa cấp hệ điều hành để phân phối phần mềm trong các gói được gọi là container. [11] Dịch vụ này có cả phiên bản miễn phí và trả phí. Phần mềm lưu trữ các container được gọi là Docker Engine. [12] Docker lần đầu tiên được phát hành vào năm 2013 và được phát triển bởi công ty Docker, Inc. [13]

- Docker là một công cụ được sử dụng để tự động hóa việc triển khai các ứng dụng trong các container nhẹ, cho phép các ứng dụng hoạt động hiệu quả trong các môi trường khác nhau mà vẫn giữ tính cô lập.

* + 1. ***Ưu điểm***

- Tính di động cao: Docker sử dụng container để đóng gói ứng dụng cùng với tất cả các phụ thuộc (libraries, dependencies, configuration files). Điều này giúp ứng dụng chạy ổn định trên mọi môi trường từ máy tính cá nhân, máy chủ, đến các dịch vụ cloud.

- Tiết kiệm tài nguyên: Docker sử dụng ảo hóa cấp hệ điều hành thay vì ảo hóa toàn bộ máy như các máy ảo (VM). Điều này giúp Docker tiêu thụ ít tài nguyên hơn, khởi động nhanh hơn và hiệu suất cao hơn so với máy ảo.

- Khả năng mở rộng linh hoạt: Docker dễ dàng tích hợp với các công cụ như Kubernetes để triển khai và quản lý hàng nghìn container, giúp mở rộng ứng dụng một cách nhanh chóng.

- Dễ dàng triển khai và quản lý: Docker tự động hóa quy trình xây dựng, triển khai và quản lý ứng dụng thông qua các tệp Dockerfile và công cụ quản lý Docker Compose. Điều này làm giảm thời gian thiết lập và quản lý môi trường phát triển.

- Hệ sinh thái phong phú: Docker Hub, kho lưu trữ container chính thức của Docker, cung cấp hàng ngàn hình ảnh (images) đã được chuẩn bị sẵn từ cộng đồng và các nhà phát triển, giúp tiết kiệm thời gian khi bắt đầu dự án.

- Hỗ trợ CI/CD tốt: Docker tích hợp liền mạch với các công cụ tích hợp liên tục (CI) và triển khai liên tục (CD) như Jenkins, GitLab CI/CD, GitHub Actions, giúp tăng tốc độ phát triển và triển khai phần mềm.

- Tăng cường bảo mật ứng dụng: Các container hoạt động độc lập, cách ly ứng dụng và các phụ thuộc của chúng khỏi hệ điều hành chính, giúp giảm nguy cơ ảnh hưởng lẫn nhau giữa các ứng dụng***.***

* + 1. ***Nhược điểm***

- Hiệu năng kém hơn so với hệ điều hành gốc: Mặc dù hiệu quả hơn máy ảo, Docker vẫn không đạt được hiệu năng tương đương với việc chạy trực tiếp trên hệ điều hành gốc, đặc biệt là trong các ứng dụng yêu cầu hiệu năng cao.

- Khó quản lý trong môi trường lớn: Khi số lượng container tăng lên, việc quản lý chúng trở nên phức tạp. Cần các công cụ bổ trợ như Kubernetes hoặc Docker Swarm để quản lý và cân bằng tải.

- Không phù hợp cho tất cả các ứng dụng: Docker không phải lúc nào cũng là giải pháp tốt nhất cho các ứng dụng nặng về đồ họa, hệ điều hành Windows hoặc các ứng dụng yêu cầu tính toàn vẹn dữ liệu cao.

- Phụ thuộc vào nhân Linux: Docker hoạt động tốt nhất trên hệ điều hành Linux. Trên các hệ điều hành khác như Windows hoặc macOS, Docker chạy thông qua một máy ảo, có thể làm giảm hiệu suất.

- Cần đường cong học tập: Đối với người mới, việc học cách viết Dockerfile, quản lý image và container, hoặc xử lý các vấn đề mạng trong Docker có thể đòi hỏi nhiều thời gian và công sức.

- Rủi ro bảo mật nếu không quản lý tốt: Nếu sử dụng các hình ảnh từ nguồn không đáng tin cậy trên Docker Hub, bạn có thể đưa mã độc hoặc lỗ hổng bảo mật vào ứng dụng.

* 1. **Giới thiệu về Github**
     1. ***Khái niệm***

**- GitHub** là một dịch vụ cung cấp [kho lưu trữ mã nguồn](https://vi.wikipedia.org/w/index.php?title=Kho_l%C6%B0u_tr%E1%BB%AF_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)) dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) sẽ được cung cấp kho lưu trữ miễn phí. Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn [14], làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới



Hình 2. 4: Logo GitHub

* + 1. ***Ưu điểm***

- Quản lý mã nguồn hiệu quả: GitHub cung cấp một nền tảng mạnh mẽ để quản lý mã nguồn thông qua Git, cho phép theo dõi, lưu trữ và quản lý lịch sử thay đổi của dự án một cách rõ ràng và dễ dàng.

- Hỗ trợ làm việc nhóm: Tính năng Pull Requests, Issues, Projects, và Code Review giúp các nhóm cộng tác hiệu quả, thảo luận về mã nguồn, và theo dõi tiến độ dự án.

- Dễ dàng tích hợp: GitHub tích hợp tốt với nhiều công cụ DevOps và CI/CD như Jenkins, CircleCI, Travis CI, GitLab CI/CD, giúp tự động hóa quy trình phát triển và triển khai.

- Hỗ trợ kho lưu trữ công khai và riêng tư: GitHub cho phép tạo kho lưu trữ công khai miễn phí, lý tưởng cho các dự án mã nguồn mở, và cũng cung cấp tùy chọn lưu trữ riêng tư cho các dự án thương mại.

- Cộng đồng lớn và tài nguyên phong phú: Là nền tảng phổ biến nhất, GitHub có cộng đồng rộng lớn, cung cấp hàng triệu dự án mã nguồn mở và tài liệu hữu ích. Điều này làm cho nó trở thành một nơi lý tưởng để học tập và chia sẻ kinh nghiệm.

- Giao diện trực quan và dễ sử dụng: Giao diện web và ứng dụng GitHub Desktop thân thiện, hỗ trợ quản lý mã nguồn ngay cả với những người không quen với dòng lệnh.

- Hỗ trợ tài liệu và Wiki: GitHub cho phép tạo Wiki và tài liệu chi tiết cho dự án, giúp các nhóm và cộng đồng dễ dàng hiểu và sử dụng dự án.

- Bảo mật mạnh mẽ: Các tính năng như quét lỗ hổng bảo mật, kiểm tra phụ thuộc, và bảo vệ nhánh (Branch Protection) giúp tăng cường bảo mật cho các dự án.

- Triển khai GitHub Pages: GitHub hỗ trợ triển khai các trang web tĩnh thông qua GitHub Pages, rất tiện lợi cho việc lưu trữ tài liệu hoặc trang giới thiệu dự án.

* + 1. ***Nhược điểm***

- Tốn phí cho nhiều tính năng nâng cao: Dù có gói miễn phí, nhưng các tính năng nâng cao như phân quyền chi tiết, tích hợp bảo mật mở rộng, và lưu trữ dung lượng lớn đòi hỏi phải trả phí.

- Giới hạn dung lượng: GitHub không được tối ưu hóa cho lưu trữ file lớn (hình ảnh, video, dữ liệu). Lưu trữ file nhị phân lớn cần dùng thêm công cụ như Git LFS, có giới hạn miễn phí và có thể phát sinh chi phí.

- Không phải giải pháp toàn diện cho CI/CD: Mặc dù có GitHub Actions, các dự án lớn vẫn cần các công cụ CI/CD chuyên nghiệp như Jenkins hoặc GitLab CI/CD để đáp ứng nhu cầu phức tạp hơn.

- Khả năng tùy chỉnh hạn chế: So với các đối thủ như GitLab, GitHub có ít khả năng tùy chỉnh hơn, đặc biệt trong việc cài đặt và quản lý trên máy chủ riêng.

- Độ phức tạp với người mới: Người mới làm quen với Git và GitHub có thể gặp khó khăn trong việc sử dụng các khái niệm như commit, branch, pull request, merge.

- Phụ thuộc vào internet: GitHub chủ yếu hoạt động trên nền tảng cloud, do đó yêu cầu kết nối internet liên tục để truy cập và làm việc với các kho lưu trữ.

- Quyền kiểm soát bị giới hạn (khi không trả phí): Trong các kho lưu trữ công khai, mọi người đều có thể xem mã nguồn. Với kho lưu trữ riêng tư, việc phân quyền phức tạp hơn mà không sử dụng gói trả phí.

* 1. **Giới thiệu về Celery**
     1. ***Khái niệm***

- Celery là một hàng đợi tác vụ bất đồng bộ hoặc hàng đợi công việc mã nguồn mở, dựa trên việc truyền tin nhắn phân tán. Mặc dù hỗ trợ lập lịch, Celery tập trung chủ yếu vào các hoạt động thời gian thực.[15]

- Các đơn vị thực thi, được gọi là tác vụ, được thực thi đồng thời trên một hoặc nhiều nút công nhân bằng cách sử dụng đa xử lý, eventlet[16] hoặc gevent.[17] Các tác vụ có thể thực thi không đồng bộ (ở chế độ nền) hoặc đồng bộ (chờ cho đến khi sẵn sàng). Celery được sử dụng trong các hệ thống sản xuất, cho các dịch vụ như Instagram, để xử lý hàng triệu tác vụ mỗi ngày.[15]



Hình 2. 5: Logo Celery

* + 1. ***Ưu điểm***

- Xử lý không đồng bộ: Celery cho phép tách biệt việc xử lý các tác vụ nặng hoặc dài ra khỏi luồng chính của ứng dụng, giúp cải thiện hiệu suất và trải nghiệm người dùng.

- Khả năng mở rộng: Celery dễ dàng mở rộng bằng cách thêm nhiều worker để xử lý khối lượng công việc lớn hơn. Bạn có thể phân phối các worker trên nhiều máy chủ.

- Hỗ trợ nhiều message broker: Celery tương thích với nhiều message broker phổ biến như RabbitMQ, Redis, Amazon SQS, v.v., giúp linh hoạt trong việc lựa chọn hệ thống phù hợp với nhu cầu.

- Quản lý tác vụ mạnh mẽ: Celery cung cấp các tính năng như lập lịch (periodic tasks), retry khi lỗi, theo dõi trạng thái tác vụ, và nhóm các tác vụ (task chaining, chord, group, map-reduce).

- Hỗ trợ nhiều ngôn ngữ: Mặc dù Celery chủ yếu được sử dụng với Python, các ngôn ngữ khác cũng có thể giao tiếp với Celery thông qua giao thức AMQP.

- Cộng đồng mạnh và tài liệu tốt: Celery có cộng đồng người dùng rộng lớn và nhiều tài liệu hỗ trợ, giúp dễ dàng tìm kiếm giải pháp hoặc nhận sự hỗ trợ.

* + 1. ***Nhược điểm***

- Cài đặt phức tạp: Quá trình thiết lập Celery với một message broker (ví dụ: RabbitMQ hoặc Redis) và cấu hình các worker ban đầu có thể phức tạp, đặc biệt đối với người mới.

- Phụ thuộc vào message broker: Celery yêu cầu một message broker để hoạt động, điều này có thể làm tăng độ phức tạp trong triển khai và bảo trì.

- Sử dụng tài nguyên: Celery worker tiêu tốn khá nhiều tài nguyên hệ thống, đặc biệt là khi chạy nhiều worker đồng thời hoặc xử lý các tác vụ nặng.

- Khó debug: Debug các lỗi trong hệ thống phân tán như Celery có thể khó khăn, đặc biệt khi xảy ra các vấn đề liên quan đến message broker hoặc task chaining.

- Giới hạn cho real-time: Celery không phải là lựa chọn tối ưu cho các tác vụ yêu cầu thời gian thực (real-time) vì có độ trễ trong việc xếp hàng và phân phối tác vụ.

- Hạn chế về monitor mặc định: Dù Celery hỗ trợ theo dõi trạng thái của tác vụ, giao diện mặc định không thân thiện và cần các công cụ bổ sung như Flower để cải thiện trải nghiệm quản lý.

- Khó khăn trong scaling phức tạp: Khi hệ thống trở nên phức tạp, việc quản lý, giám sát và tối ưu Celery để đảm bảo hiệu suất có thể đòi hỏi nhiều công sức và kiến thức sâu.

* 1. **Giới thiệu về Pusher**
     1. ***Khái niệm***

- Pusher là một dịch vụ cloud, tạo ra một server trung gian giúp chúng ta có thể xử lý các tác vụ thời gian thực. Dữ liệu được gửi tới pusher, và pusher lại gửi nó đi tới các client đã subscribe (đăng ký) và các channel. Trong đó Pusher Channel cung cấp giao tiếp thời gian thực giữa các máy chủ, ứng dụng và thiết bị. Các kênh được sử dụng cho các biểu đồ thời gian thực, danh sách người dùng thời gian thực, bản đồ thời gian thực, chơi trò chơi nhiều người chơi và nhiều loại cập nhật giao diện người dùng khác. Nó có một thư viện hỗ trợ mọi thứ như trình duyệt web, ứng dụng iOS và Android, khung PHP, chức năng đám mây, tập lệnh bash, thiết bị IoT. Pusher Channel hoạt động ở mọi nơi vì nó sử dụng WebSockets và HTTP và cung cấp dự phòng cho các thiết bị không hỗ trợ WebSockets.

* + 1. ***Ưu điểm***

- Hỗ trợ thời gian thực mạnh mẽ: Pusher Channels sử dụng WebSockets, một giao thức lý tưởng cho giao tiếp hai chiều với độ trễ thấp, rất phù hợp cho các ứng dụng thời gian thực như chat, thông báo, hoặc cập nhật trạng thái.

- Dễ dàng tích hợp: Cung cấp thư viện hỗ trợ cho nhiều nền tảng (JavaScript, iOS, Android, PHP, Python, IoT, v.v.), giúp việc tích hợp trở nên nhanh chóng và thuận tiện.

- Dự phòng cho thiết bị không hỗ trợ WebSockets: Nếu thiết bị hoặc trình duyệt không hỗ trợ WebSockets, Pusher tự động chuyển sang sử dụng các giao thức thay thế như HTTP long-polling, đảm bảo dịch vụ luôn hoạt động.

- Dịch vụ được quản lý hoàn toàn: Là một dịch vụ cloud, Pusher Channels giúp bạn tránh việc phải tự triển khai hoặc quản lý các máy chủ WebSocket phức tạp, tiết kiệm công sức và tài nguyên.

- Khả năng mở rộng: Dịch vụ được thiết kế để xử lý lượng lớn người dùng và kết nối cùng lúc, phù hợp với các ứng dụng lớn cần mở rộng nhanh chóng.

- Tính năng phong phú: Hỗ trợ các kênh công khai (public channels), kênh riêng tư (private channels), và kênh hiện diện (presence channels) giúp quản lý người dùng và bảo mật hiệu quả.

- Cung cấp webhook để tích hợp các sự kiện thời gian thực vào các hệ thống khác.

- Thời gian hoạt động và độ tin cậy cao: Pusher được triển khai trên cơ sở hạ tầng mạnh mẽ, đảm bảo thời gian hoạt động (uptime) cao và độ tin cậy tốt cho các ứng dụng.

- Tài liệu và cộng đồng mạnh: Pusher Channels có tài liệu phong phú và cộng đồng tích cực, giúp dễ dàng tìm kiếm sự hỗ trợ và học cách sử dụng hiệu quả.

* + 1. ***Nhược điểm***

- Chi phí cao: Là một dịch vụ đám mây, Pusher có chi phí theo mô hình trả phí dựa trên số kết nối đồng thời, sự kiện được gửi đi, và các tính năng bổ sung. Đối với các ứng dụng lớn hoặc nhiều người dùng, chi phí có thể tăng nhanh chóng.

- Phụ thuộc vào dịch vụ bên thứ ba: Sử dụng Pusher Channels đồng nghĩa với việc phụ thuộc vào một dịch vụ đám mây, điều này có thể là vấn đề nếu dịch vụ gặp sự cố hoặc không có sẵn ở một số khu vực.

- Hạn chế trong tùy chỉnh: Vì là dịch vụ được quản lý hoàn toàn, khả năng tùy chỉnh sâu vào các tính năng của hệ thống có thể bị giới hạn so với việc tự triển khai máy chủ WebSocket.

- Độ trễ ở các vùng xa: Mặc dù Pusher cung cấp độ trễ thấp, khoảng cách địa lý giữa máy khách và trung tâm dữ liệu Pusher vẫn có thể gây ra độ trễ không đáng kể trong một số trường hợp.

- Không phù hợp cho ứng dụng nhạy cảm bảo mật cao: Dữ liệu phải truyền qua server của Pusher, nên có nguy cơ liên quan đến bảo mật và quyền riêng tư, đặc biệt với các ứng dụng yêu cầu kiểm soát dữ liệu tuyệt đối.

- Giới hạn bởi các gói dịch vụ: Các gói miễn phí hoặc cơ bản của Pusher có giới hạn về số kết nối, sự kiện và các tính năng nâng cao như kênh hiện diện, khiến việc thử nghiệm trên quy mô lớn bị hạn chế

* 1. **Giới thiệu về Open Policy Agent**
     1. ***Khái niệm***

- Open Policy Agent (OPA, phát âm là “oh-pa”) là một công cụ chính sách nguồn mở, mục đích chung, thống nhất việc thực thi chính sách trên toàn bộ ngăn xếp. OPA cung cấp một ngôn ngữ khai báo cấp cao cho phép bạn chỉ định chính sách dưới dạng mã và API đơn giản để chuyển giao việc ra quyết định chính sách khỏi phần mềm của bạn. Bạn có thể sử dụng OPA để thực thi chính sách trong các dịch vụ vi mô, Kubernetes, CI/CD pipeline, API gateway, v.v.



Hình 2. 6: Logo Open Policy Agent

* + 1. ***Ưu điểm***

- Thống nhất quản lý chính sách trên nhiều hệ thống như Kubernetes, API Gateway, CI/CD.

- Ngôn ngữ Rego mạnh mẽ, hỗ trợ logic phức tạp và dễ tích hợp.

- Hiệu suất cao với caching, không phụ thuộc công nghệ cụ thể.

- Dễ mở rộng và tích hợp tốt với các công cụ như Terraform, Kubernetes Gatekeeper.

- Hỗ trợ quan sát và gỡ lỗi, giúp xác minh và kiểm tra chính sách.

* + 1. ***Nhược điểm***

- Khó học ngôn ngữ Rego, đặc biệt với người mới.

- Thiết lập phức tạp trong hệ thống lớn hoặc không đồng nhất.

- Không phù hợp cho thời gian thực, có thể gặp độ trễ nếu xử lý phức tạp.

- Quản lý chính sách phức tạp khi số lượng quy tắc tăng cao.

- Thiếu GUI thân thiện, chủ yếu qua CLI và mã nguồn

* 1. **Giới thiệu về Json Web Token**
     1. ***Khái niệm***

- JSON Web Token (JWT, đề xuất phát âm là /dʒɒt/, giống từ "jot") là một tiêu chuẩn Internet được đề xuất để tạo dữ liệu với chữ ký và/hoặc mã hóa tùy chọn, trong đó payload chứa JSON xác nhận một số tuyên bố (claims). Các token được ký bằng cách sử dụng một bí mật riêng hoặc một cặp khóa công khai/khóa riêng tư.

* + 1. ***Ưu điểm***

- Tự chứa thông tin (Self-Contained): JWT chứa thông tin xác thực và các dữ liệu cần thiết (claims) bên trong chính nó. Điều này giúp giảm thiểu truy vấn đến cơ sở dữ liệu để xác minh token, từ đó tăng tốc hiệu suất hệ thống.

- Khả năng mở rộng: Do tính chất không cần lưu trữ trạng thái trên server (stateless), JWT phù hợp với các hệ thống phân tán hoặc microservices, nơi mà việc quản lý trạng thái phức tạp.

- Bảo mật mạnh mẽ: JWT có thể được ký bằng thuật toán HMAC (HS256) hoặc mã hóa không đối xứng (RS256), giúp đảm bảo tính toàn vẹn và xác thực của token.

- Với mã hóa không đối xứng, chỉ cần public key là đủ để xác thực, không cần lưu trữ private key trên server.

- Đa nền tảng và tiêu chuẩn: JWT là một tiêu chuẩn mở (RFC 7519) và hỗ trợ hầu hết các ngôn ngữ lập trình phổ biến. Điều này làm cho việc triển khai trên nhiều nền tảng trở nên dễ dàng.

- Tính linh hoạt: Token có thể chứa các thông tin tùy chỉnh như quyền truy cập (roles), tên người dùng, hoặc thời gian hết hạn, giúp tích hợp linh hoạt vào nhiều hệ thống khác nhau.

- Giảm tải trên server: Với cơ chế lưu trữ thông tin trực tiếp trong token, server không cần phải duy trì phiên làm việc (session) trên bộ nhớ, giúp giảm gánh nặng lưu trữ.

- Khả năng sử dụng rộng rãi: JWT có thể sử dụng trong các hệ thống xác thực (authentication), ủy quyền (authorization), và truyền tải dữ liệu an toàn giữa các bên (secure data exchange).

* + 1. ***Nhược điểm***

- Kích thước lớn hơn: So với các session token thông thường, JWT thường lớn hơn do chứa thêm thông tin tự chứa (claims) và các chữ ký (signature). Điều này có thể gây tốn băng thông khi truyền qua mạng, đặc biệt trong hệ thống có nhiều request.

- Không thể thu hồi (Stateless): Một khi token được phát hành, server không thể thu hồi nó trừ khi có cơ chế thêm như danh sách đen (blacklist). Điều này có thể dẫn đến rủi ro nếu token bị đánh cắp.

- Quản lý thời gian hết hạn phức tạp: Nếu thời gian hết hạn (expiration time) quá dài, token bị lộ có thể bị sử dụng trong khoảng thời gian dài. Nếu thời gian quá ngắn, người dùng cần đăng nhập lại thường xuyên, gây khó chịu.

- Cần thêm cơ chế refresh token để giải quyết vấn đề này.

- Dễ bị tấn công nếu khóa bí mật bị lộ: Nếu private key hoặc secret key bị lộ, kẻ tấn công có thể giả mạo và tạo ra các token hợp lệ, gây hậu quả nghiêm trọng.

- Không mã hóa dữ liệu mặc định: JWT chỉ đảm bảo tính toàn vẹn và xác thực của dữ liệu, nhưng không mặc định mã hóa payload. Nếu chứa thông tin nhạy cảm, cần kết hợp thêm cơ chế mã hóa.

- Chi phí xác thực cao hơn với thuật toán RSA: Khi sử dụng thuật toán mã hóa không đối xứng như RS256, việc xác thực token tiêu tốn tài nguyên CPU hơn so với HMAC, gây ảnh hưởng đến hiệu suất hệ thống.

- Quá phụ thuộc vào token: Khi sử dụng JWT, toàn bộ dữ liệu phụ thuộc vào token. Nếu không có cơ chế kiểm tra bổ sung, dữ liệu trong token có thể trở thành nguồn duy nhất để xác thực và ủy quyền, gây rủi ro nếu token bị giả mạo.

# **Chương 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

## **3.1. Xác định yêu cầu**

- Hệ thống ứng dụng quản lý y tế trực tuyến nhằm hỗ trợ các vai trò: bệnh nhân, lễ tân, và bác sĩ. Yêu cầu chung của hệ thống:

* Đơn giản hóa quy trình quản lý và đặt lịch hẹn.
* Cung cấp các công cụ hiện đại như thanh toán online, thông báo thời gian thực, và tích hợp nhận diện khuôn mặt.
* Đảm bảo tính bảo mật thông tin và phân quyền rõ ràng giữa các vai trò.

- Yêu cầu chi tiết từ các bên liên quan:

* Bệnh nhân:
  + Dễ dàng đặt lịch, thanh toán, và quản lý thông tin cá nhân.
  + Tiện lợi trong việc nhận thông báo nhắc lịch và các thông tin từ bác sĩ/lễ tân.
* Lễ tân:
  + Quản lý lịch hẹn của bệnh nhân và bác sĩ hiệu quả.
  + Thực hiện các thao tác thanh toán nhanh chóng, chính xác.
* Bác sĩ:
  + Quản lý lịch làm việc và hồ sơ bệnh án.
  + Tương tác dễ dàng với bệnh nhân và lễ tân thông qua thông báo thời gian thực.

## **3.2. Phân tích yêu cầu**

* + 1. ***Chức năng Đăng ký( 3 role)***

- Khi đăng ký, bệnh nhân cần cung cấp căn cước công dân số điện thoại, mật khẩu và xác thực thông qua AI .Sau khi xác thực thành công, hệ thống truyền thông tin tài khoản bệnh nhân, phân quyền và lưu thông tin vào cơ sở dữ liệu.

* + 1. ***Chức năng Đăng nhập(3 role)***

- Người dùng**:** Khi đăng nhập, người dùng nhập email hoặc số điện thoại và mật khẩu. Sau khi thông tin xác thực hợp lệ, hệ thống lưu lại token của thiết bị để duy trì phiên làm việc.

- Lễ tân: Khi đăng nhập, lễ tân được chuyển đến giao diện quản lý lịch hẹn và thanh toán. Hệ thống lưu token để duy trì bảo mật thông tin.

- Bác sĩ: Sau khi đăng nhập, bác sĩ truy cập giao diện quản lý lịch hẹn, bệnh án và hồ sơ bệnh nhân. Token được lưu lại để quản lý phiên làm việc.

* + 1. ***Chức năng Đặt lịch đến lễ tân (role bệnh nhân)***

- Bệnh nhân truy cập hệ thống, chọn ngày giờ mong muốn và gửi yêu cầu đặt lịch với lễ tân.

- Hệ thống kiểm tra khoảng thời gian rảnh của lễ tân và xác nhận lịch hẹn.

- Thông báo xác nhận được gửi đến bệnh nhân qua hệ thống hoặc email.

* + 1. ***Chức năng Xem lịch của bản thân (role bệnh nhân)***

- Khi truy cập mục "Lịch của tôi," bệnh nhân sẽ thấy danh sách các lịch hẹn đã đặt.

- Hệ thống hiển thị thông tin chi tiết, bao gồm ngày, giờ, địa điểm, trạng thái (đã xác nhận, hủy).

* + 1. ***Chức năng Quản lý thanh toán của bản thân (role bệnh nhân)***

- Bệnh nhân có thể xem lịch sử thanh toán, trạng thái hóa đơn (đã thanh toán, chờ xử lý).

- Hệ thống cho phép tải xuống biên lai hoặc xem chi tiết từng giao dịch.

* + 1. ***Chức năng Thanh toán online (role bệnh nhân)***

- Bệnh nhân chọn hóa đơn cần thanh toán. Hệ thống chuyển hướng đến cổng thanh toán trực tuyến.

- Sau khi hoàn tất giao dịch, hệ thống cập nhật trạng thái hóa đơn và gửi thông báo xác nhận.

* + 1. ***Chức năng Đặt lịch đến bác sĩ (role lễ tân)***

- Lễ tân chọn bệnh nhân, bác sĩ, ngày giờ và nhập lý do hẹn.

- Hệ thống tự động lọc các bác sĩ rảnh hoặc đề xuất bác sĩ có thời gian trống gần nhất.

- Sau khi xác nhận, thông báo được gửi đến bệnh nhân và bác sĩ.

* + 1. ***Chức năng Quản lý thanh toán lễ tân (role lễ tân)***

- Lễ tân có quyền truy cập danh sách thanh toán của tất cả bệnh nhân.

* + 1. ***Chức năng Thanh Toán cho bệnh nhân (role lễ tân)***

- Lễ tân xử lý thanh toán bằng tiền mặt hoặc thẻ tại quầy.

- Sau khi thanh toán, hệ thống cập nhật trạng thái hóa đơn và in biên lai.

* + 1. ***Chức năng Quản lý lịch hẹn của bệnh nhân (role lễ tân)***

- Lễ tân xem danh sách lịch hẹn.

* + 1. ***Chức năng Quản lý lịch hẹn (role bác sĩ)***

- Bác sĩ truy cập danh sách lịch hẹn của mình theo ngày/tuần/tháng.

- Hệ thống hiển thị trạng thái của từng lịch hẹn và cho phép cập nhật trạng thái (đã khám, hủy).

* + 1. ***Chức năng Đặt lịch đến bác sĩ (role bác sĩ)***

- Bác sĩ thêm lịch khám mới, chỉ định thời gian và loại dịch vụ.

* + 1. ***Chức năng Tạo hồ sơ bệnh án (role bác sĩ)***

- Bác sĩ nhập thông tin của bệnh nhân, kết quả chẩn đoán và kế hoạch điều trị.

- Hệ thống lưu trữ và cho phép cập nhật hồ sơ khi cần thiết.

* + 1. ***Chức năng Nhận thông báo gửi cho bản thân (3 role)***

- Hệ thống gửi thông báo cá nhân hóa liên quan đến lịch hẹn, thanh toán, hoặc thông tin quan trọng.

* + 1. ***Chức năng Nhận thông báo thời gian thực từ pusher (3 role)***

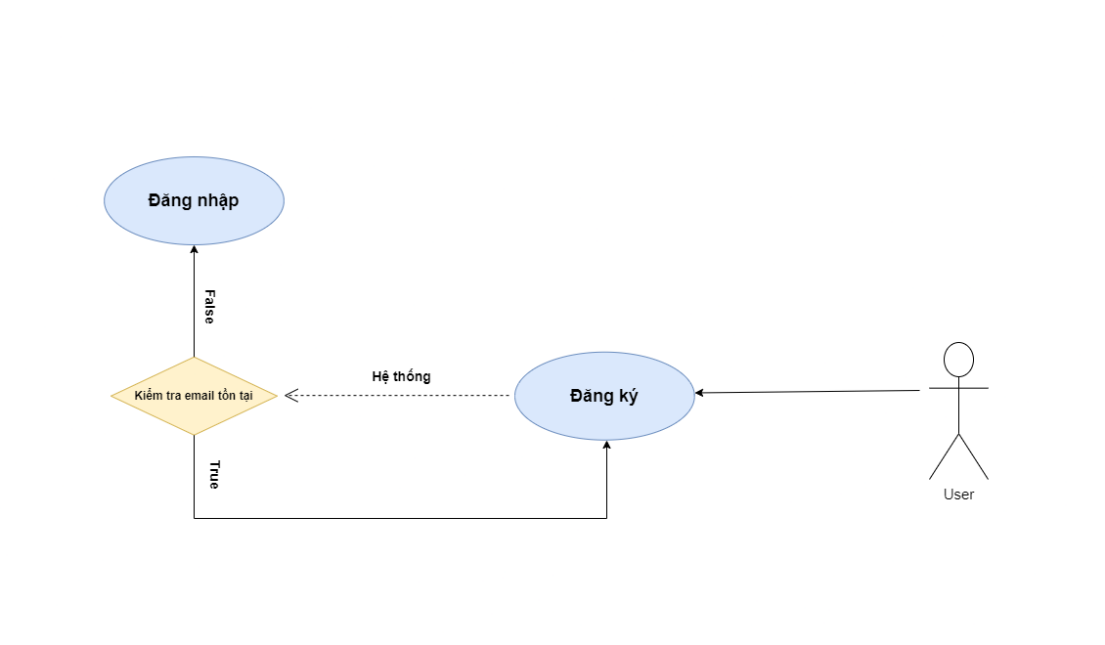
- Thông báo thời gian thực qua Pusher giúp cập nhật ngay các sự kiện quan trọng như xác nhận lịch hẹn hoặc xác nhận thanh toán.

* + 1. ***Chức năng gửi thông báo (3 role)***

- Gửi thông báo tự động đến người dùng khác khi có sự kiện cần thiết

* 1. **Mô hình hóa yêu cầu**
     1. ***Chức năng Đăng ký***

- Biểu đồ Use Case đăng ký



Hình 3. 1: Use Case đăng ký

- Mô tả Use Case đăng ký

# Bảng 3. 1: Use Case đăng ký

|  |  |
| --- | --- |
| Tên Use Case | Đăng ký |
| Tác nhân | Bệnh nhân |
| Điều kiện đầu vào | Người dùng chưa đăng nhập |
| Kết quả đầu ra | Người dùng đăng ký thành công |
| Mô tả | Use case cho phép người dùng đăng ký |

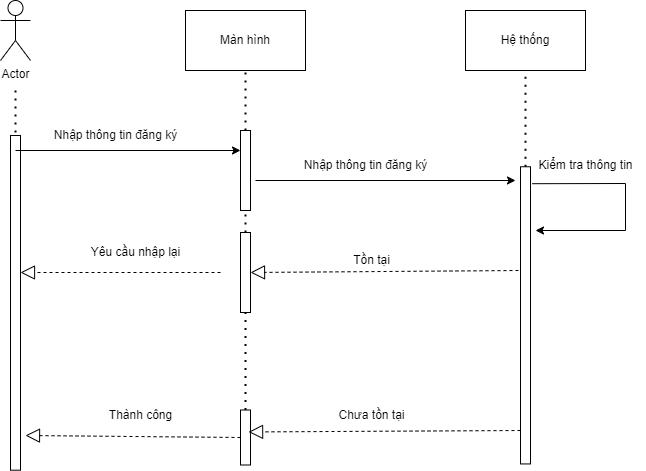
- Dòng chính Use Case đăng ký

# Bảng 3. 2: Use Case dòng chính đăng ký

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Người dùng truy cập vào web |
| 2 | Hệ thống | Kiểm tra email có tồn không |
| 3 | Tác nhân | Nhập email, mật khẩu |

# 

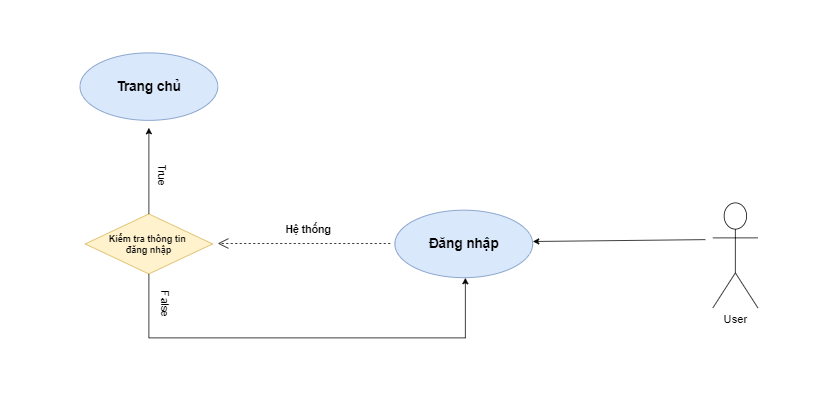
- Biểu đồ tuần tự đăng ký



Hình 3. 2: Tuần Tự đăng ký

***Chức năng Đăng nhập***

- Biểu đồ Use Case đăng nhập



Hình 3. 3: Use Case đăng nhập

Mô tả Use Case đăng nhập

# Bảng 3. 3: Use Case dòng chính đăng nhập

|  |  |
| --- | --- |
| Tên Use Case | Đăng nhập |
| Tác nhân | Bác sĩ, bệnh nhân, lễ tân |
| Điều kiện đầu vào | Người dùng chưa đăng nhập |
| Kết quả đầu ra | Người dùng đăng nhập thành công |
| Mô tả | Use case cho phép người dùng đăng nhập |

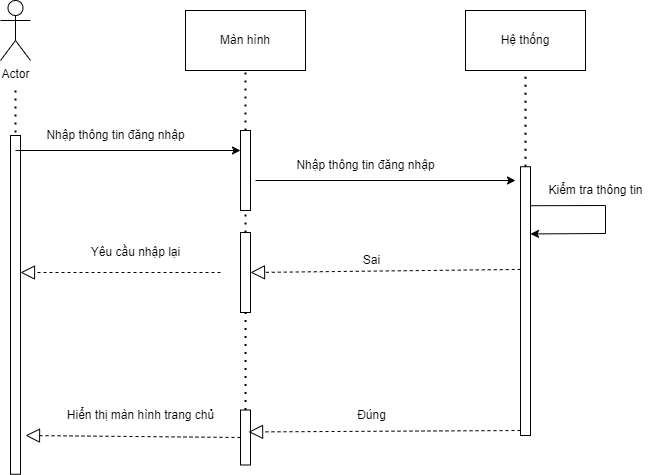
- Dòng chính Use Case đăng nhập

# Bảng 3. 4: Use Case dòng chính đăng nhập

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Người dùng truy cập vào web |
| 2 | Hệ thống | Kiểm tra số điện thoại và mật khẩu có đúng không |
| 3 | Tác nhân | Nhập email, mật khẩu |

# 

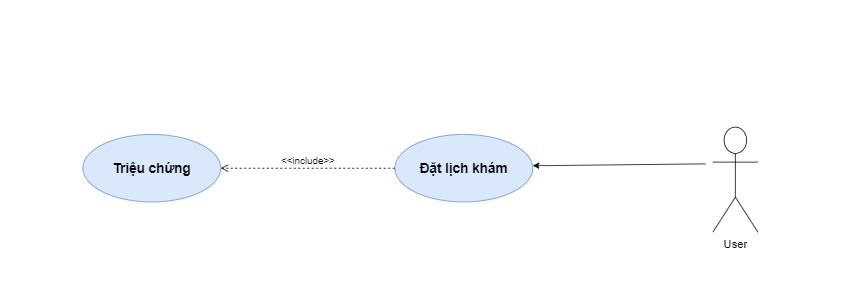
- Biểu đồ tuần tự đăng nhập



Hình 3. 4: Tuần Tự đăng nhập

* + 1. ***Chức năng Đặt lịch khám***

- Biểu đồ Use Case Đặt lịch khám



Hình 3. 5: Use Case đặt lịch khám

- Mô tả Use Case Đặt lịch khám

# Bảng 3. 5: Use Case đặt lịch khám

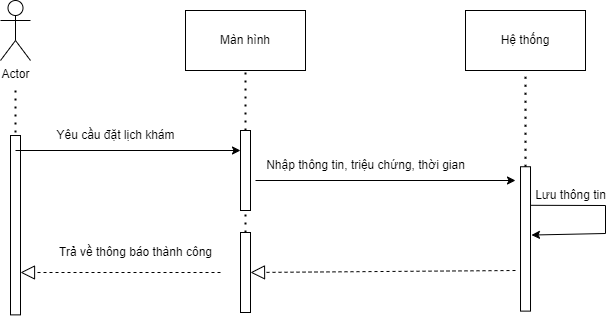
|  |  |
| --- | --- |
| Tên Use Case | Đặt lịch khám |
| Tác nhân | Bệnh nhân |
| Điều kiện đầu vào | Người dùng đã đăng nhập  Người dùng nhấn vào đặt lịch khám và nhập triệu chứng bệnh |
| Kết quả đầu ra | Người dùng đặt lịch thành công |
| Mô tả | Use case cho phép người dùng đăng lịch khám bệnh |

- Dòng chính Use Case đặt lịch khám

# Bảng 3. 6: Use Case dòng chính đặt lịch khám

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Người dùng đăng nhập thành công |
| 2 | Tác nhân | Nhập triệu chứng |
| 3 | Hệ thống | Lưu thông tin triệu chứng của bệnh nhân |

- Biểu đồ tuần tự đặt lịch khám



Hình 3. 6: Tuần Tự đặt lịch khám

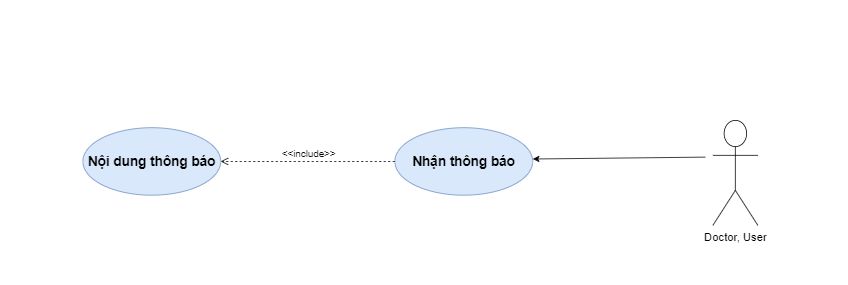






* + 1. ***Chức năng Thông báo***

- Biểu đồ Use Case Thông báo



Hình 3. 7: Tuần Tự Use Case thông báo

- Mô tả Use Case Thông báo

# Bảng 3. 7: Use Case thông báo

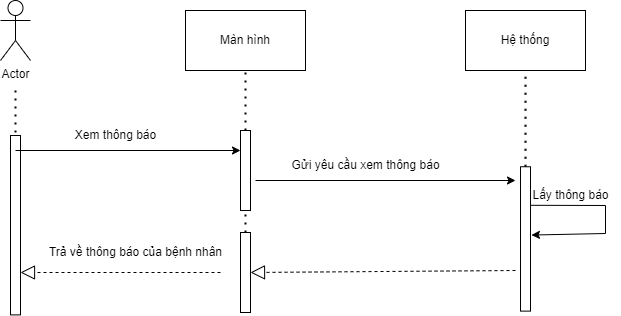
|  |  |
| --- | --- |
| Tên Use Case | Thông báo |
| Tác nhân | Bệnh nhân, Bác sĩ |
| Điều kiện đầu vào | Người dùng đã đăng nhập  Nhấn vào thông báo |
| Kết quả đầu ra | Người dùng và Bác sĩ xem được những thông báo đã nhận |
| Mô tả | Use case cho phép người dùng và Bác sĩ xem được những thông báo đã nhận |

- Dòng chính Use Case Thông báo

# Bảng 3. 8: Use Case Dòng chính thông báo

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Người dùng đăng nhập thành công |
| 2 | Tác nhân | Nhấn xem thông báo |
| 3 | Hệ thống | Trả về thông báo |

- Biểu đồ tuần tự chức năng thông báo



Hình 3. 8: Tuần Tự chức năng thông báo

* + 1. ***Chức năng Quản lý và đặt lịch khám cho bệnh nhân***

- Biểu đồ Use Case Quản lý và đặt lịch khám cho bệnh nhân



Hình 3. 9: Use Case Quản lý đặt lịch

- Mô tả Use Case Quản lý và đặt lịch khám cho bệnh nhân

# Bảng 3. 9: Use Case Quản lý và đặt lịch khám

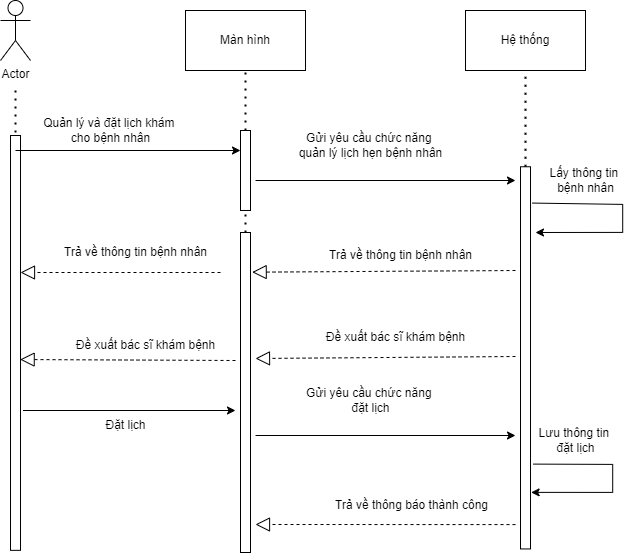
|  |  |
| --- | --- |
| Tên Use Case | Quản lý và đặt lịch khám cho bệnh nhân |
| Tác nhân | Lễ Tân |
| Điều kiện đầu vào | Lễ Tân đã đăng nhập  Nhấn vào quản lý lịch khám |
| Kết quả đầu ra | Lễ Tân có thể đặt lịch khám và gửi thông báo cho bệnh nhân và bác sĩ thành công |
| Mô tả | Use case cho phép Lễ tân đặt lịch khám cho bệnh nhân |

- Dòng chính Use Case Quản lý và đặt lịch khám cho bệnh nhân

# Bảng 3. 10: Use Case Dòng chính Quản lý và đặt lịch khám

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Lễ tân đăng nhập thành công |
| 2 | Tác nhân | Nhấn quản lý lịch khám |
| 3 | Hệ thống | Trả về thông tin bệnh nhân, triệu chứng |
| 4 | Hệ thống | Đề xuất bác sĩ và phòng khám |
| 5 | Tác nhân | Chọn bác sĩ |
| 6 | Tác nhân | Đặt lịch |
| 7 | Hệ thống | Đặt lịch thành công và trả về thông báo |

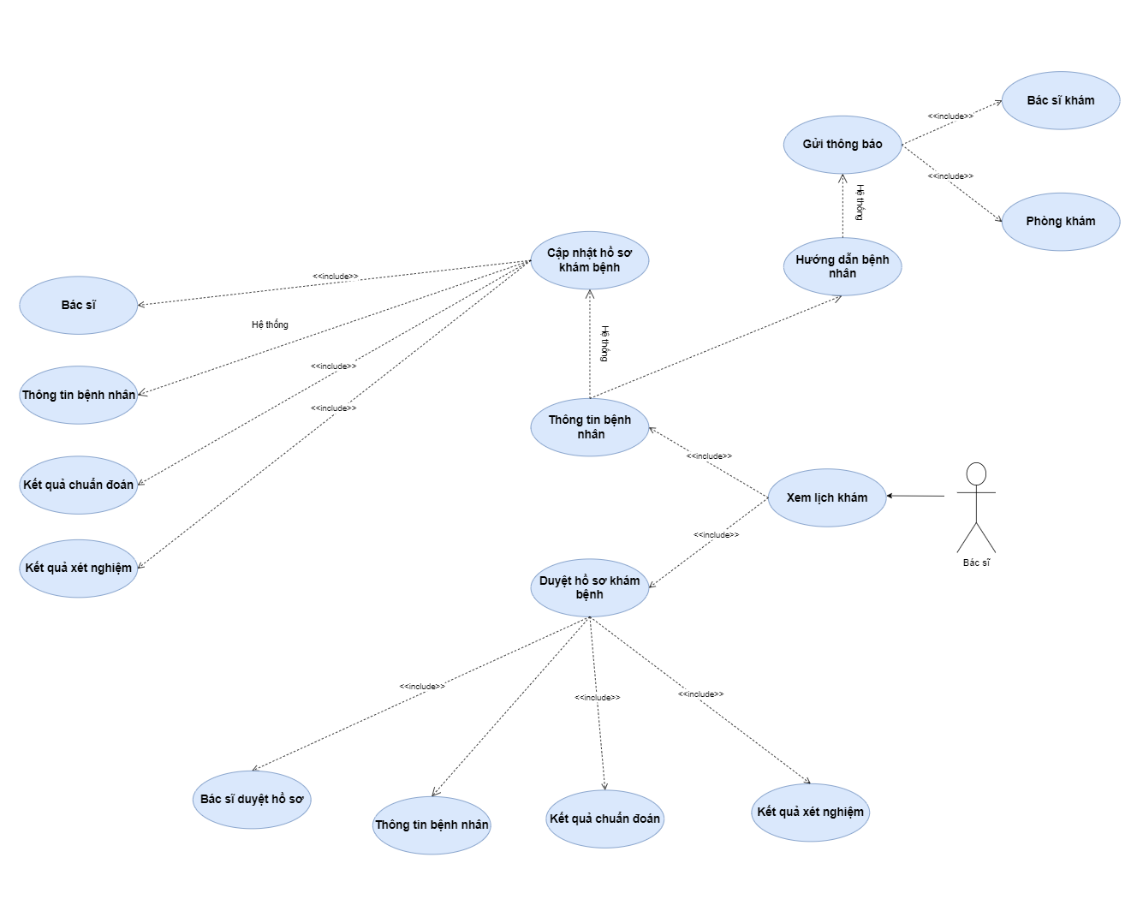
- Biểu đồ tuần tự đặt lịch khám cho bệnh nhân



Hình 3. 10: Tuần Tự đặt lịch khám

* + 1. ***Chức năng Khám bệnh***

- Biểu đồ Use Case Khám bệnh



Hình 3. 11: Use case Khám bệnh

- Mô tả Use Case Khám bệnh

# Bảng 3. 11: Use Case Khám bệnh

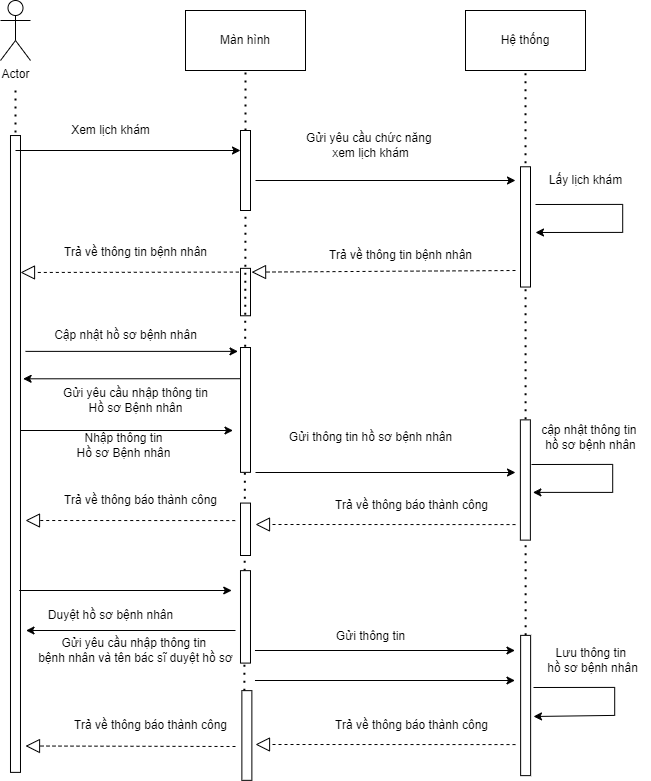
|  |  |
| --- | --- |
| Tên Use Case | Khám bệnh |
| Tác nhân | Bác sĩ |
| Điều kiện đầu vào | Bác sĩ đã đăng nhập  Nhấn vào xem lịch khám |
| Kết quả đầu ra | Bác sĩ có thể cập nhật hồ sơ và duyệt hồ sơ cho bệnh nhân và gửi thông báo thành công |
| Mô tả | Use case cho phép xem thông tin bệnh nhân và cập nhật hồ sơ bệnh nhân |

- Dòng chính Use Case khám bệnh

# Bảng 3. 12: Use Case Dòng chính Khám bệnh

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Bác sĩ đăng nhập thành công |
| 2 | Tác nhân | Nhấn xem lịch khám |
| 3 | Hệ thống | Trả về thông tin bệnh nhân, triệu chứng |
| 4 | Tác nhân | Hướng dẫn bệnh nhân |
| 5 | Hệ thống | Gửi thông báo về bác sĩ khám và phòng khám |
| 6 | Tác nhân | Cập nhật hồ sơ bệnh nhân |
| 4 | Hệ thống | Lưu vào cơ sở dữ liệu |
| 5 | Tác nhân | Duyệt hồ sơ khám bệnh |
| 6 | Hệ thống | Lưu cơ sở dữ liệu |

- Biểu đồ tuần tự khám bệnh



# Hình 3. 12: Tuần Tự khám bệnh

* + 1. ***Chức năng Thanh toán***

- Biểu đồ Use Case Chức năng thanh toán



# Hình 3.13 Biểu đồ Use Case thanh toán

- Mô tả Use Case thêm thanh toán

# Bảng 3.13 Mô tả Use Case thêm thanh toán

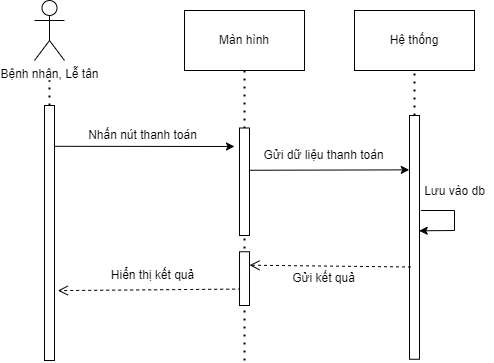
|  |  |
| --- | --- |
| Tên Use Case | Thanh toán |
| Tác nhân | Lễ tân |
| Điều kiện đầu vào | Lễ tân chọn thanh toán viện phí |
| Kết quả đầu ra | Thanh toán viện phí thành công |
| Mô tả | Use case cho phép người dùng và lễ tân thanh toán viện phí |

- Dòng chính Use Case thanh toán

# Bảng 3.14 Dòng chính Use Case thanh toán

|  |  |  |
| --- | --- | --- |
| STT | Thực hiện | Hành động |
| 1 | Tác nhân | Lễ tân và Bệnh nhân |
| 2 | Hệ thống | Hiển thị các danh sách cần thanh toán |
| 3 | Tác nhân | chọn bệnh nhân cần thanh toán |
| 4 | Hệ thống | Hiển thị nút thanh toán |
| 6 | Hệ thống | Lưu thông tin thanh toán vào cơ sở dữ liệu |
| 7 | Hệ thống | Hiển thị thông báo thanh toán thành công |

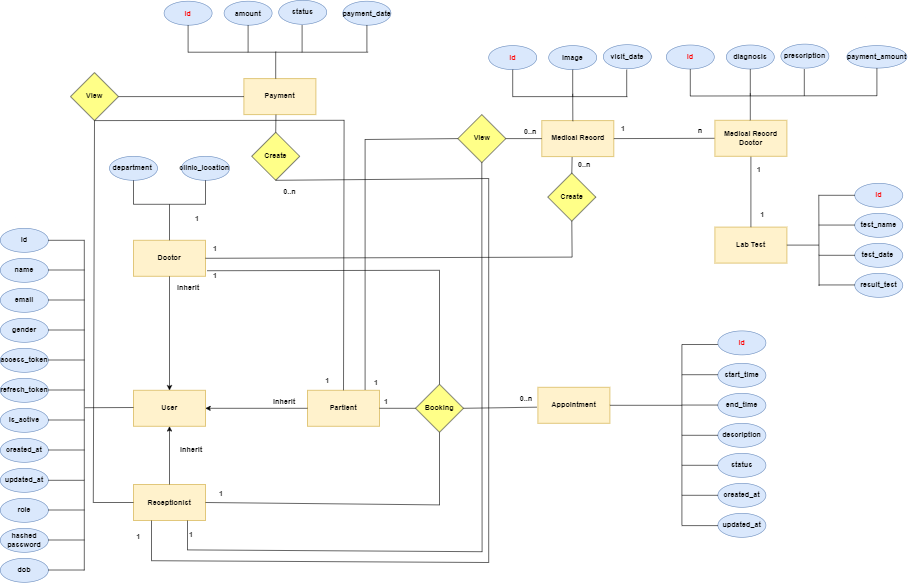
- Biểu đồ tuần tự thanh toán khoá học



# Hình 3.14 Biểu đồ tuần tự thanh toán khoá học

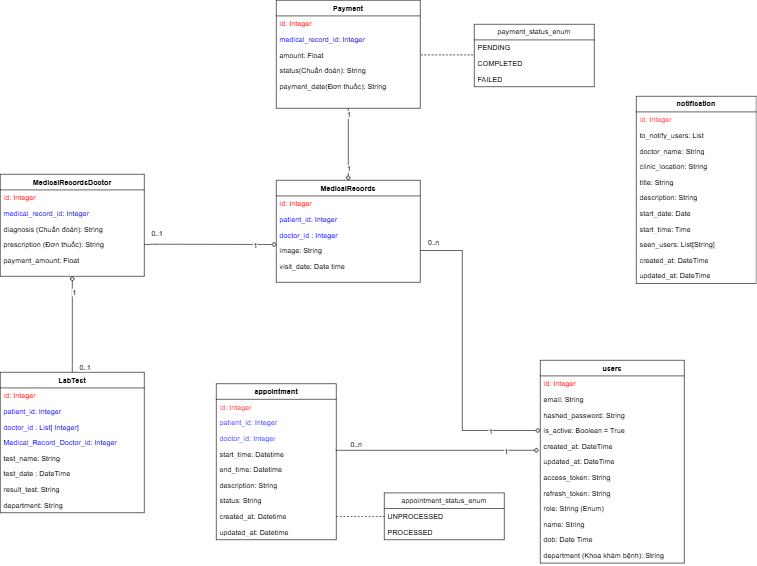
## **Thiết kế cơ sơ dữ liệu**

### **3.4.1 Mô hình thực thể liên kết**



Hình 3.15 Mô hình ERD

### **3.4.2 Mô hình dữ liệu quan hệ**

**

Hình 3.16 Mô hình ER

### **3.4.3 Danh mục các bảng**

# Bảng 3.15 Mô tả các bảng cơ sở dữ liệu

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên bảng** | **Mô tả** |
| 1 | User | Lưu trữ thông tin người dùng (email, tên, phòng ban, ngày sinh, giới tính, quyền, v.v.) |
| 2 | Appointment | Lưu trữ thông tin các lịch hẹn giữa bệnh nhân và bác sĩ |
| 3 | MedicalRecord | Lưu trữ thông tin hồ sơ bệnh án (ngày khám, bệnh nhân, bác sĩ, thanh toán, v.v.) |
| 4 | MedicalRecordDoctor | Lưu trữ thông tin khám và chẩn đoán của bác sĩ (chẩn đoán, đơn thuốc, số tiền thanh toán) |
| 5 | LabTest | Lưu trữ thông tin các xét nghiệm (tên xét nghiệm, phòng ban, ngày thực hiện, kết quả, v.v.) |
| 6 | Payment | Lưu trữ thông tin thanh toán cho hồ sơ bệnh án (số tiền, trạng thái, ngày thanh toán) |

Bảng User

# Bảng 3.16 Mô tả bảng User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Null | Key | Ghi chú |
| Id | Int | Không | Primary Key | Mã người dùng |
| name | String | Có |  | Tên người dùng |
| email | String | Không | Unique Key | Email người dùng |
| hashed\_password | String | Không |  | Mật khẩu đã mã hóa |
| clinic\_location | String | Có |  | Địa chỉ nơi làm việc |
| phone\_number | String | Không |  | Số điện thoại người dùng |
| residence | String | Có |  | Nơi cư trú của người dùng |
| role | Enum | Không |  | Quyền của user |
| gender | Enum | Không |  | Giới tính của người dùng |
| is\_active | Boolean | Không |  | Trạng thái hoạt động |
| cccd\_id | BigInteger | Có |  | CCCD (Chứng minh nhân dân) |
| created\_at | DateTime | Không |  | Ngày tạo tài khoản |
| updated\_at | DateTime | Không |  | Ngày cập nhật tài khoản |

Bảng Payment

# Bảng 3.17 Mô tả bảng Payment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Null | Key | Ghi chú |
| Id | Int | Không | Primary Key | |  | | --- | |  |  |  | | --- | | Mã thanh toán | |
| medical\_record\_id | Int | Có | Foreign Key | Liên kết tới hồ sơ bệnh án |
| amount | Float | Không |  | Số tiền thanh toán |
| status | Enum | Không |  | Trạng thái (PENDING, ...) |
| payment\_date | DateTime | Có |  | Ngày thanh toán |

Bảng Appointment

# Bảng 3.18 Mô tả bảng Appointment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Null | Key | Ghi chú |
| Id | Int | Không | Primary Key | |  | | --- | |  |  |  | | --- | | Mã lịch hẹn | |
| patient\_id | Int | Có | Foreign Key | Người bệnh (User ID) |
| doctor\_id | Float | Có | Foreign Key | Bác sĩ (User ID) |
| description | Enum | Không |  | Mô tả lịch hẹn |
| status | DateTime | Không |  | Trạng thái (UNPROCESSED, ...) |
| start\_time | DateTime | Có |  | Thời gian bắt đầu |
| end\_time | DateTime | Có |  | Thời gian kết thúc |
| created\_at | DateTime | Không |  | Ngày tạo lịch hẹn |
| updated\_at | DateTime | Không |  | Ngày cập nhật lịch hẹn |

Bảng MedicalRecordDoctor

# Bảng 3.19 Mô tả bảng MedicalRecordDoctor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Null | Key | Ghi chú |
| Id | Int | Không | Primary Key | |  | | --- | |  |  |  | | --- | | Mã bản ghi bác sĩ | |
| medical\_record\_id | Int | Không | Foreign Key | Liên kết tới hồ sơ bệnh án |
| diagnosis | String | Có |  | |  | | --- | |  |  |  | | --- | | Chẩn đoán | |
| prescription | String | Có |  | Đơn thuốc |
| payment\_amount | Float | Có |  | Số tiền liên quan |

Bảng LabTest

# Bảng 3.20 Mô tả bảng LabTest

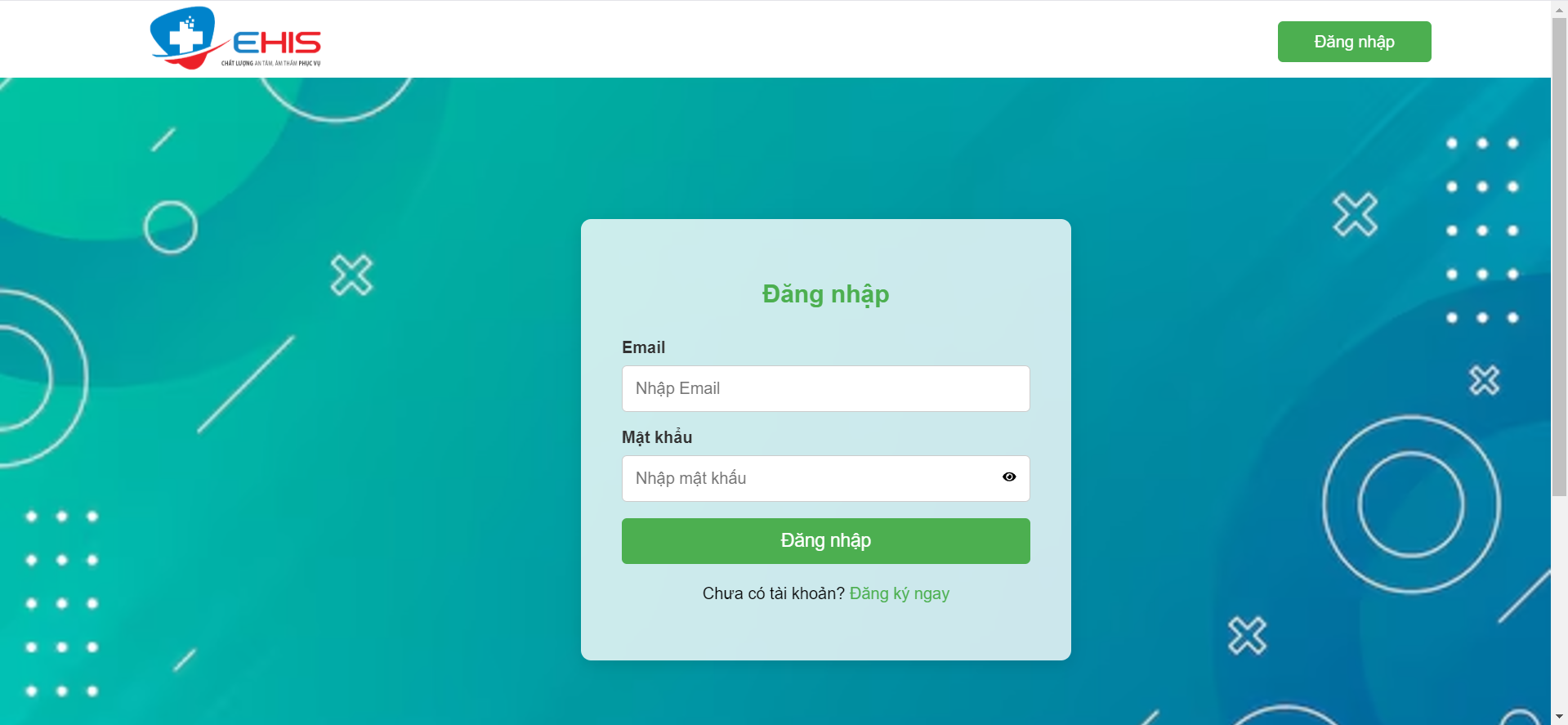
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Null | Key | Ghi chú |
| Id | Int | Không | Primary Key | |  | | --- | |  |  |  | | --- | | Mã xét nghiệm | |
| medical\_record\_doctor\_id | Int | Không | Foreign Key | ID bản ghi bác sĩ |
| test\_name | String | Không |  | |  | | --- | |  |  |  | | --- | | Tên xét nghiệm | |
| department | String | Không |  | Phòng ban thực hiện |
| test\_date | DateTime | Không |  | |  | | --- | |  |  |  | | --- | | Ngày thực hiện xét nghiệm | |
| result\_test | String | Không |  | Kết quả xét nghiệm |

## **Kết quả đạt được**

### **Giao diện người dùng**

#### **3.5.1.1 Giao diện trang đăng nhập**

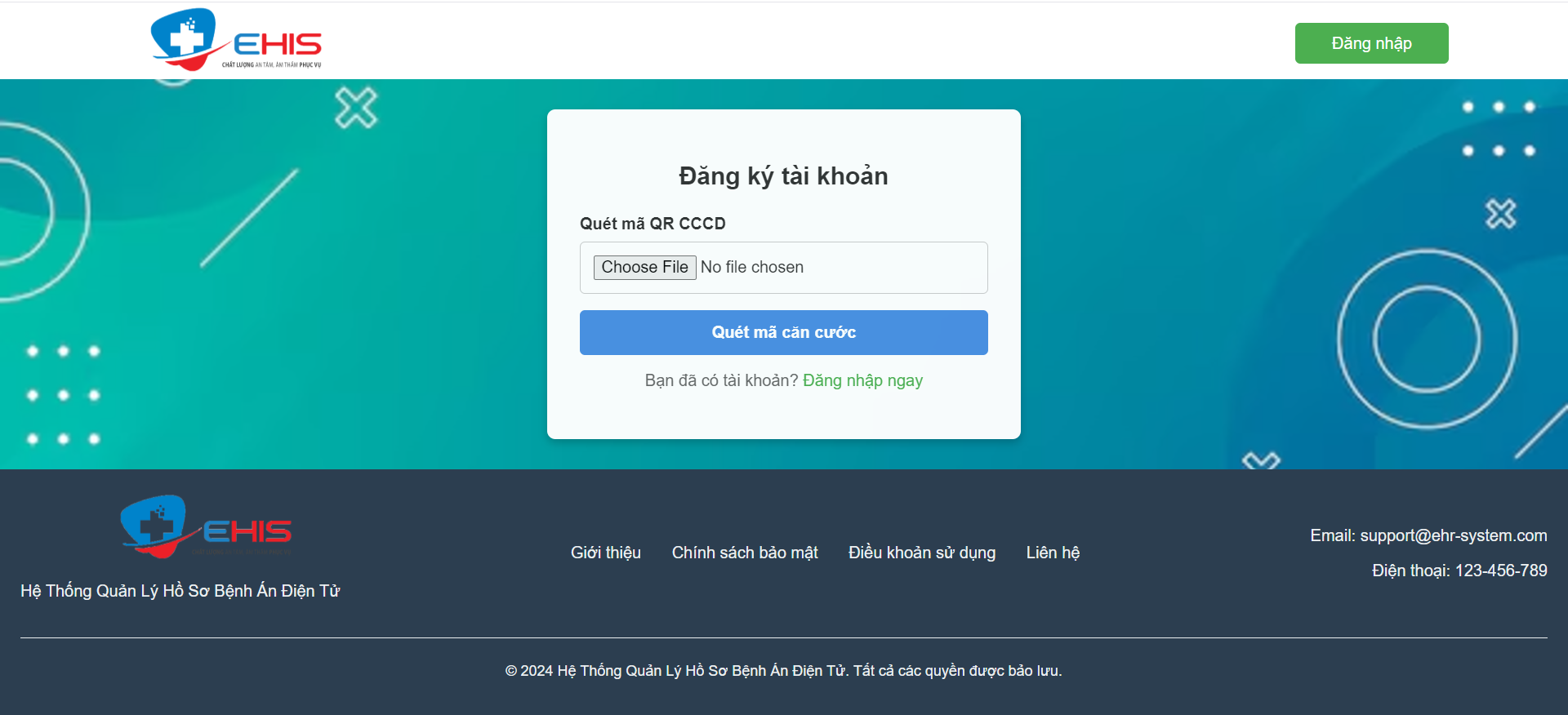
Người dùng chọn vào chức năng đăng nhập. Giao diện trang đăng nhập như Hình 3.16



# *Hình 3.16. Giao diện trang đăng nhập*

#### **3.5.1.2 Giao diện trang đăng ký xác thực bằng mã QR căn cước**

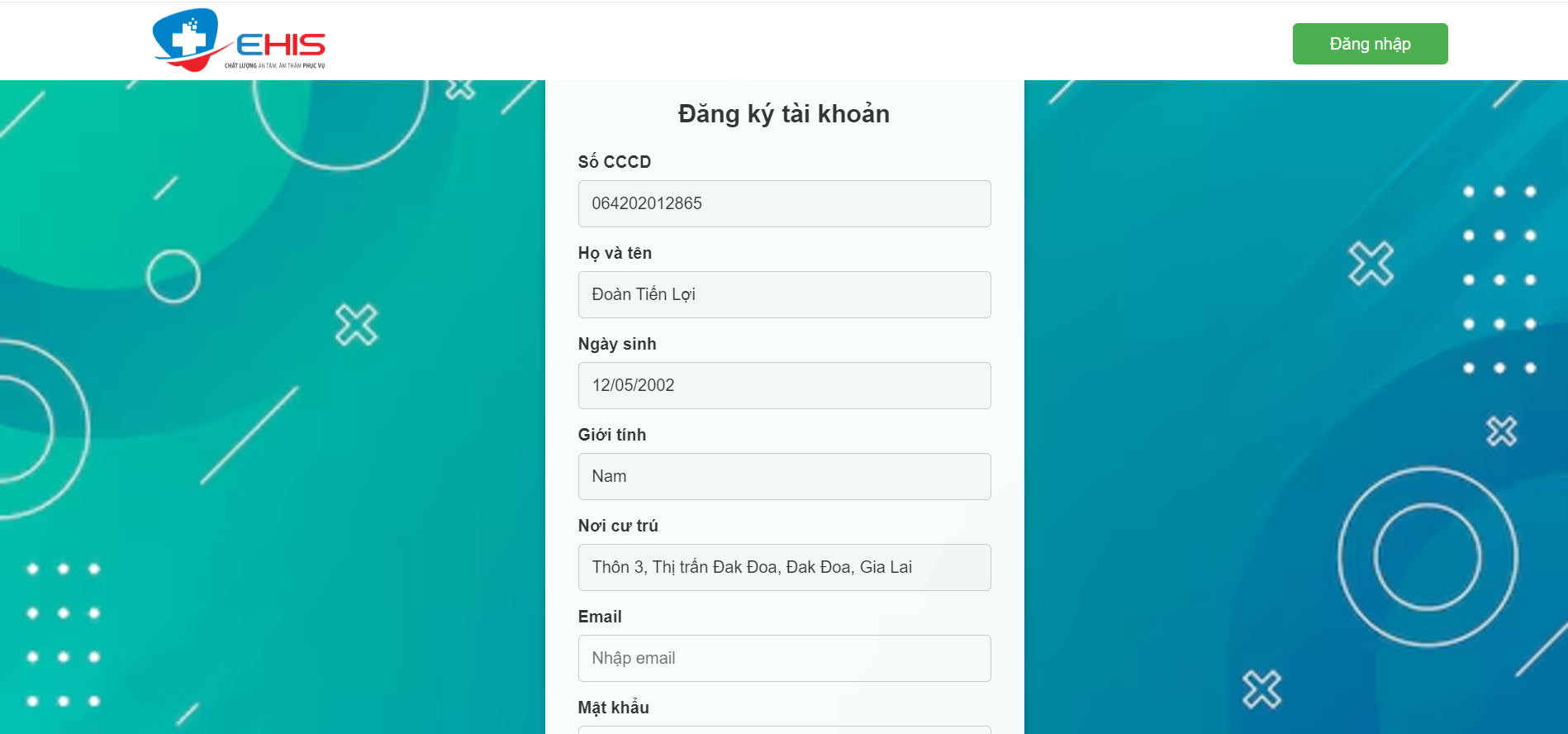
Người dùng chọn vào chức năng đăng ký. Giao diện trang đăng ký như Hình 3.17



# *Hình 3.17. Giao diện trang đăng ký*

#### **3.5.1.3 Giao diện trang đăng ký xác thực bằng mã QR căn cước**

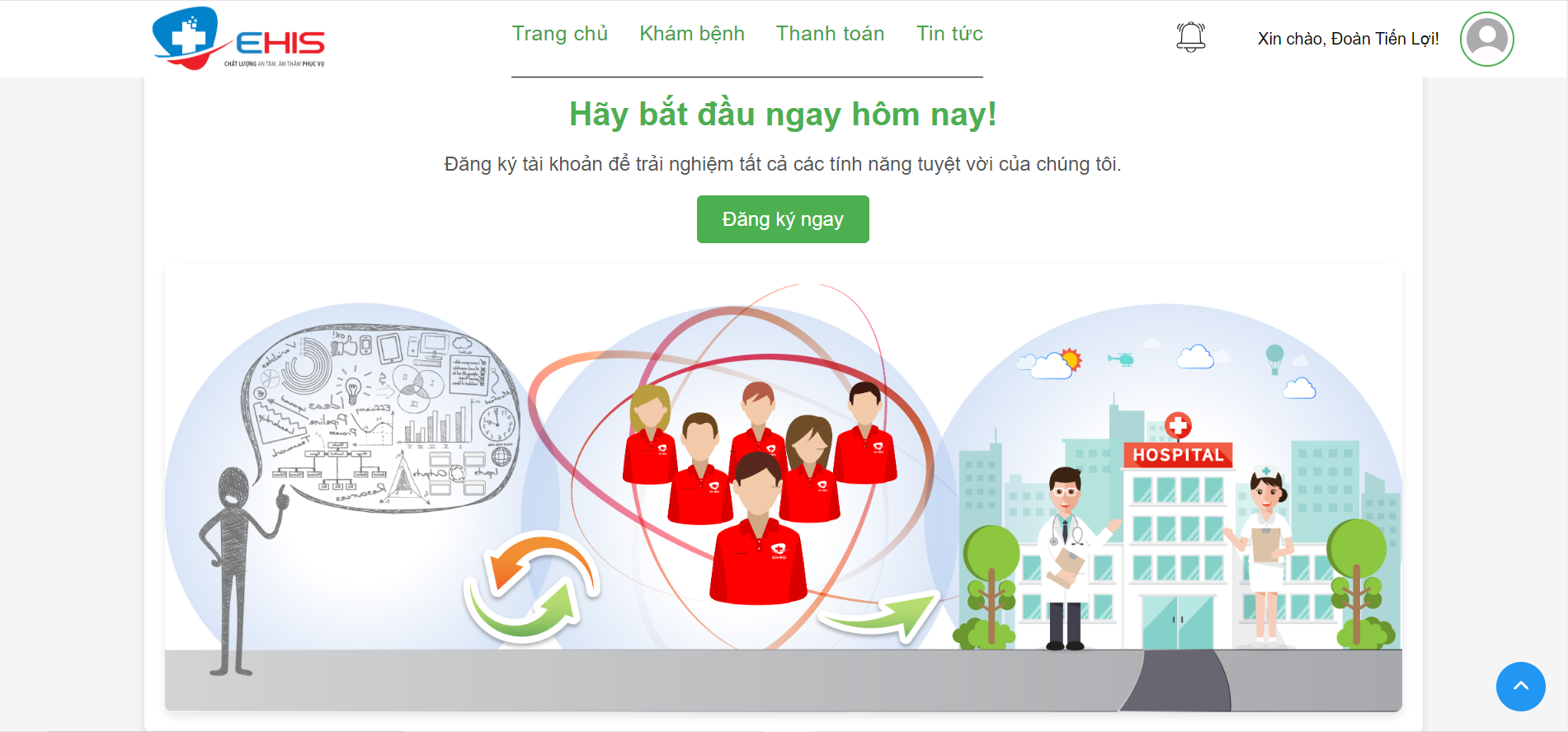
Sau khi người dùng chọn ảnh căn cước để quét. Giao diện trang đăng ký như Hình 3.18



# *Hình 3.18. Giao diện trang đăng ký*

#### **3.5.1.4 Giao diện trang chủ**

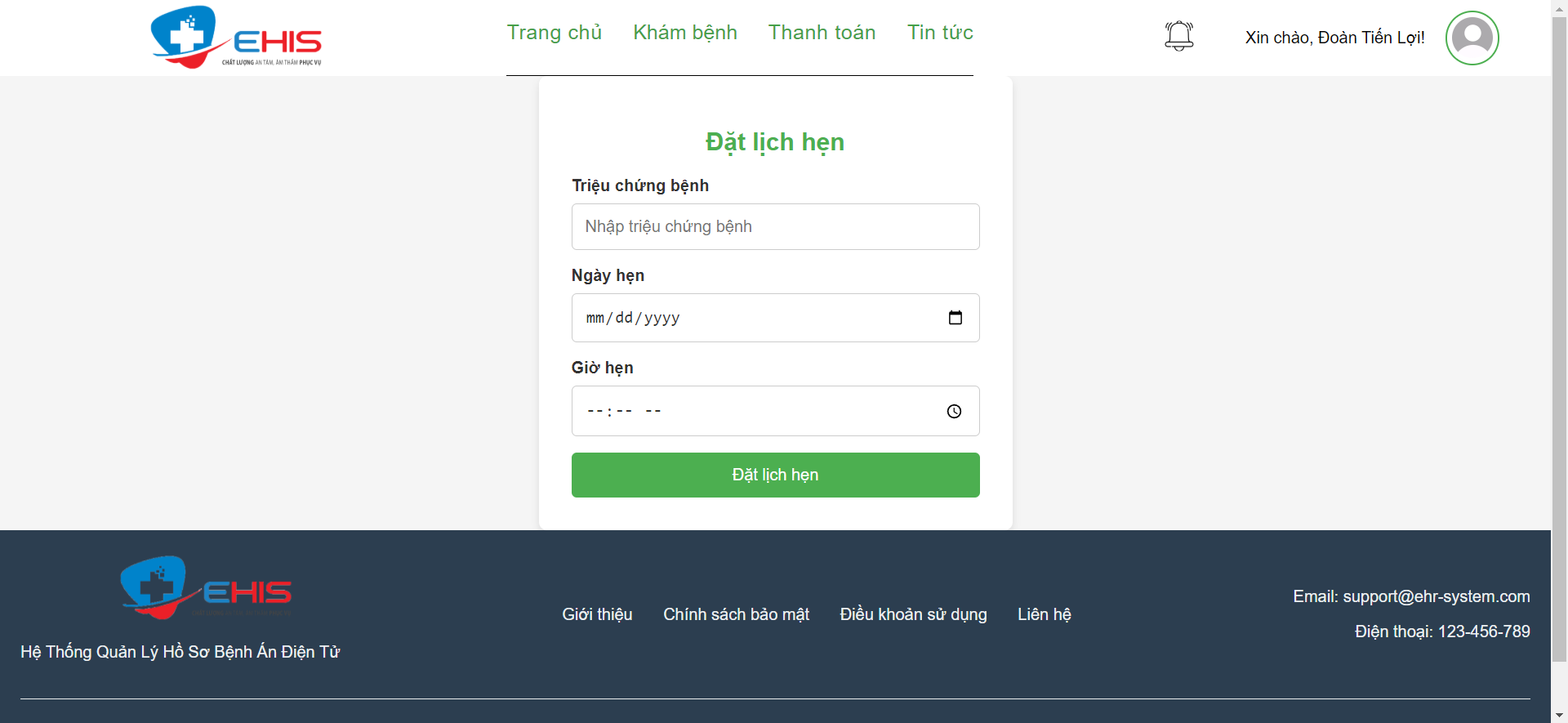
Sau khi người dùng đăng nhập thành công. Giao diện trang chủ bệnh nhân như Hình 3.19



# *Hình 3.19. Giao diện trang chủ bệnh nhân*

#### **3.5.1.5 Giao diện trang đặt lịch**

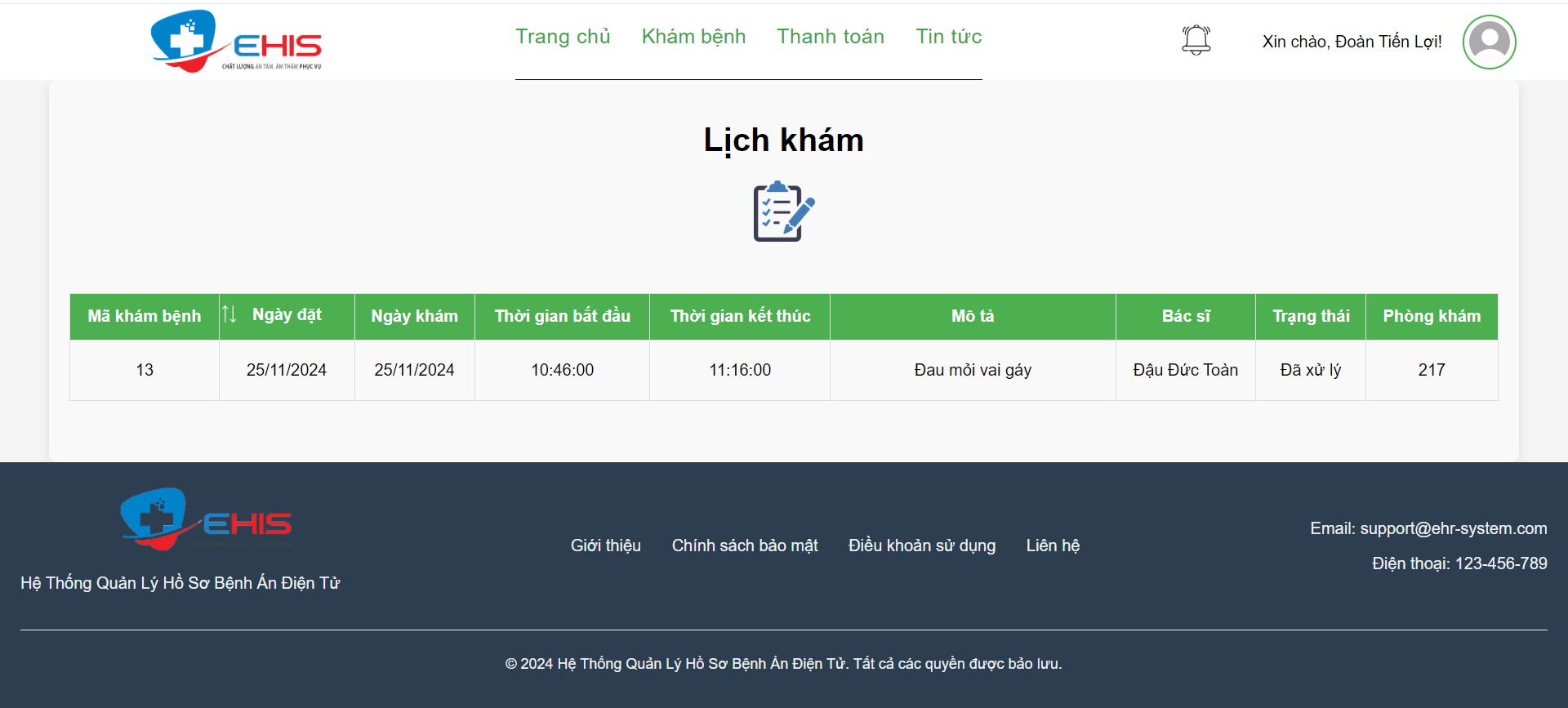
Người dùng chọn vào chức năng đặt lịch khám. Giao diện trang đặt lịch khám bệnh nhân như Hình 3.20



# *Hình 3.20. Giao diện trang đặt lịch khám bệnh nhân*

#### **3.5.1.6 Giao diện trang xem lịch khám**

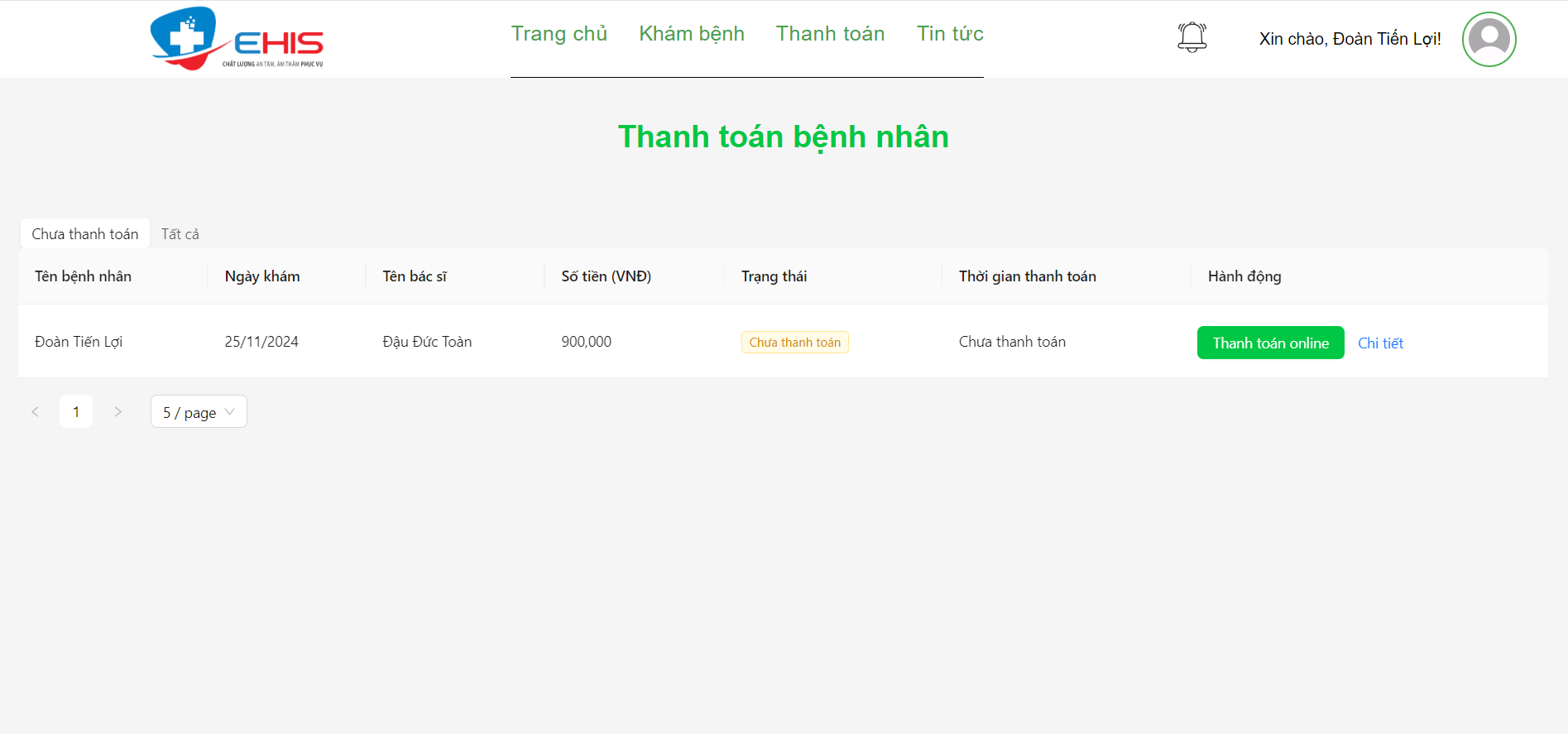
Người dùng chọn vào chức năng xem lịch khám. Giao diện trang xem lịch khám bệnh nhân như Hình 3.21



# *Hình 3.21. Giao diện trang xem lịch khám bệnh nhân*

#### **3.5.1.7 Giao diện trang thanh toán**

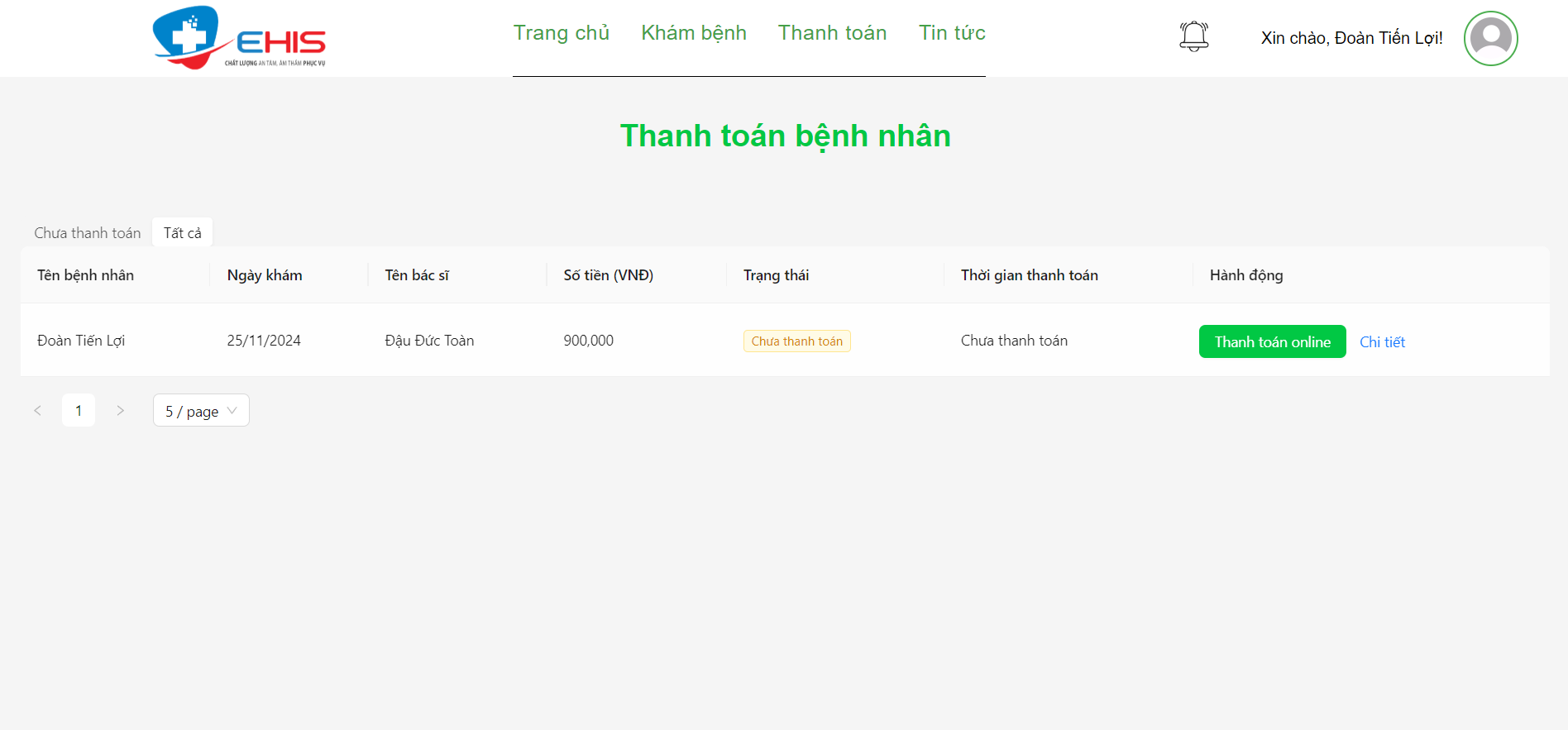
Người dùng chọn vào chức năng thanh toán bệnh nhân. Giao diện trang thanh toán bệnh nhân như Hình 3.22



# *Hình 3.22. Giao diện trang thanh toán bệnh nhân*

#### **3.5.1.8 Giao diện trang tất cả thanh toán**

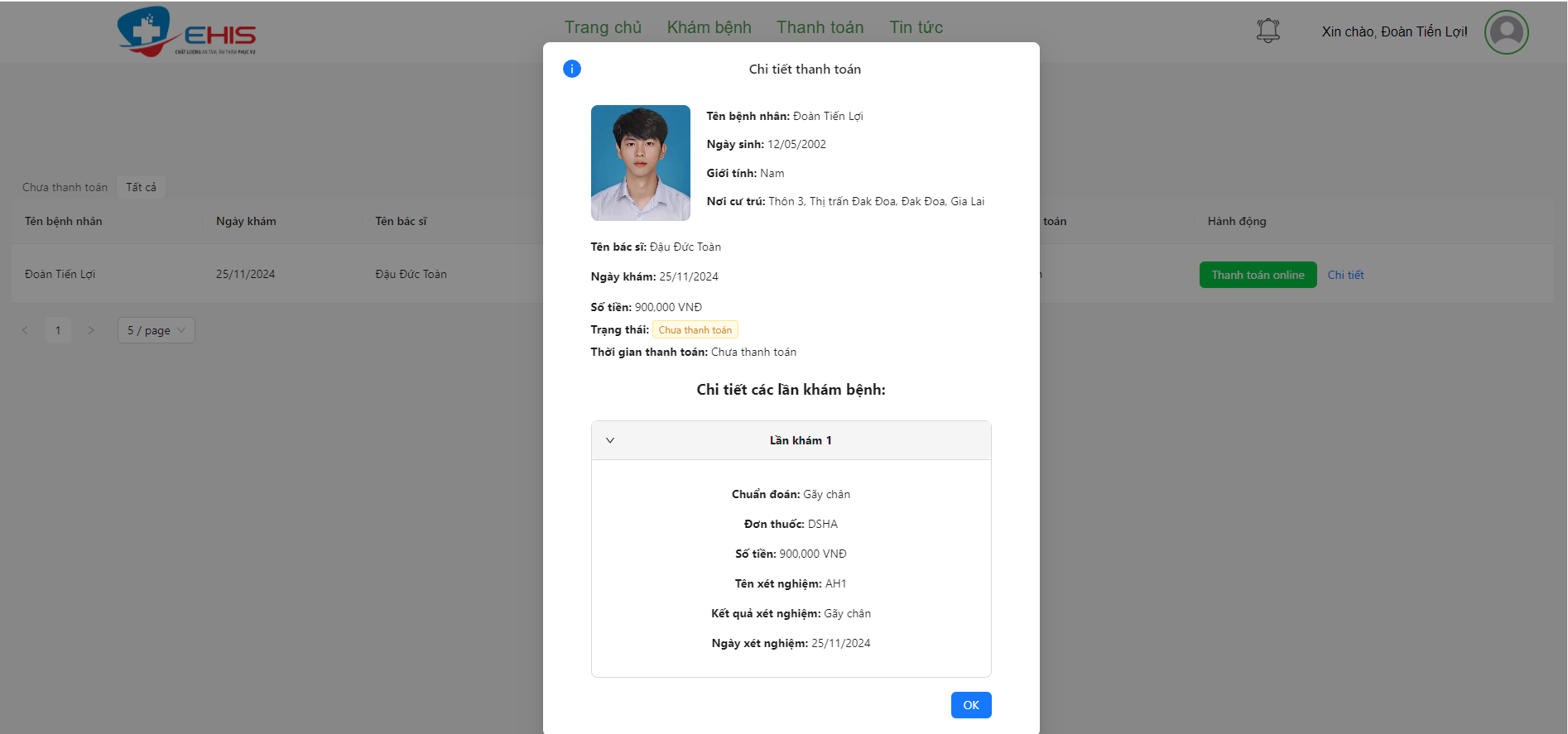
Người dùng chọn vào chức năng tất cả thanh toán bệnh nhân. Giao diện trang thanh toán bệnh nhân như Hình 3.23



# *Hình 3.23. Giao diện trang tất cả thanh toán bệnh nhân*

#### **3.5.1.5 Giao diện trang chi tiết thanh toán bệnh nhân**

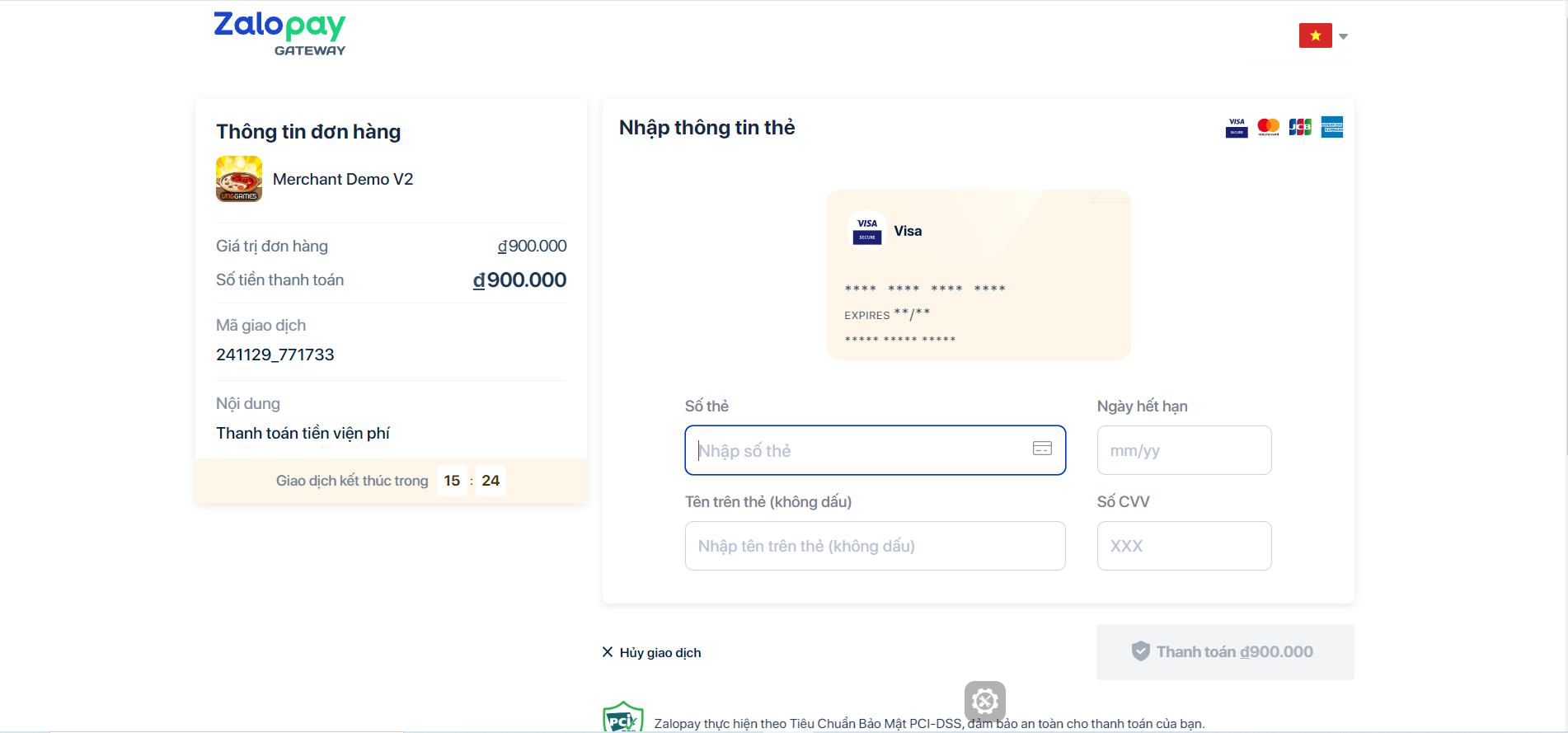
Người dùng chọn vào chức năng chi tiết thanh toán. Giao diện trang chi tiết thanh toán bệnh nhân như Hình 3.24



# *Hình 3.24. Giao diện trang tất cả thanh toán bệnh nhân*

#### **3.5.1.9 Giao diện trang nhập thông tin zalo pay**

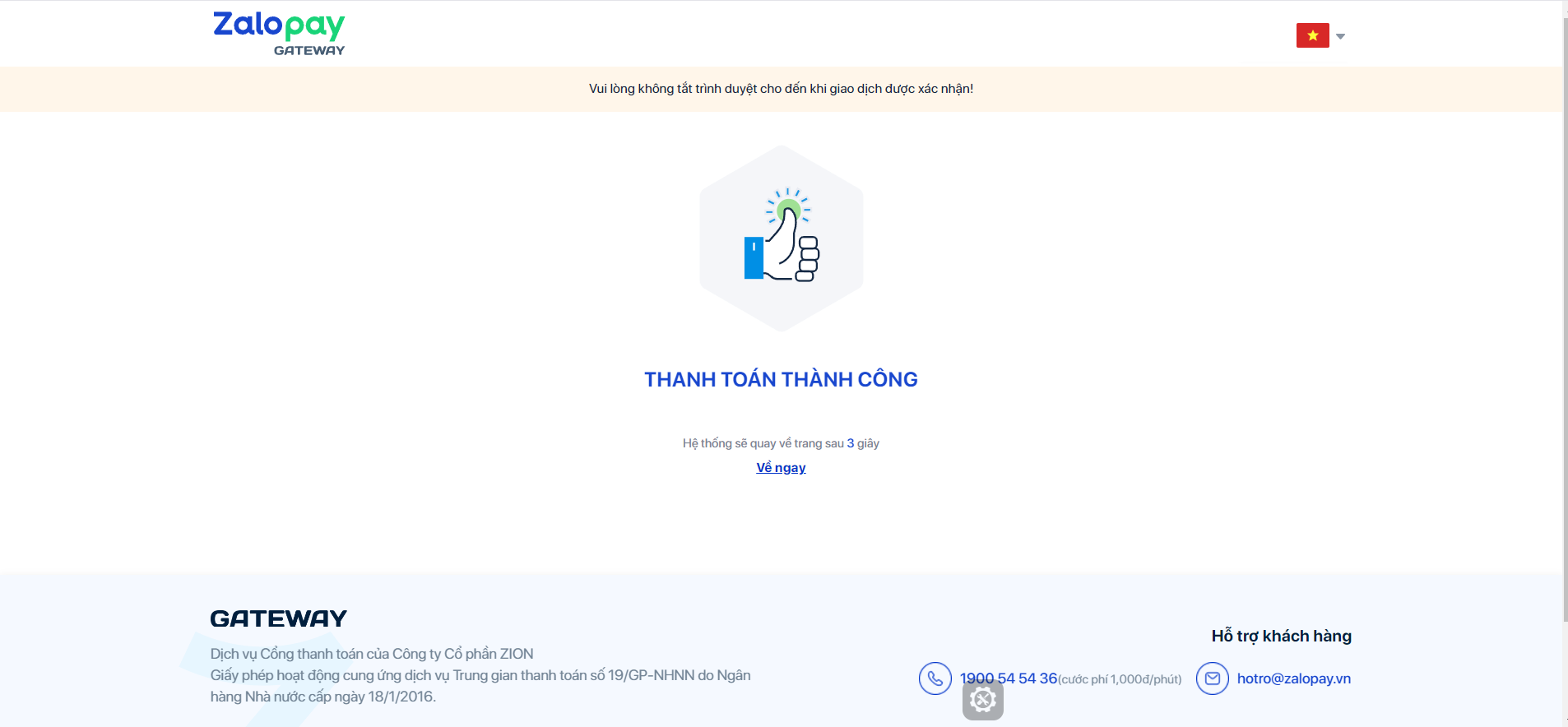
Người dùng chọn vào chức năng thanh toán. Giao diện trang thanh toán zalo pay như Hình 3.25



# *Hình 3.25. Giao diện trang nhập thông tin zalo pay*

#### **3.5.1.10 Giao diện trang nhập thông tin zalo pay**

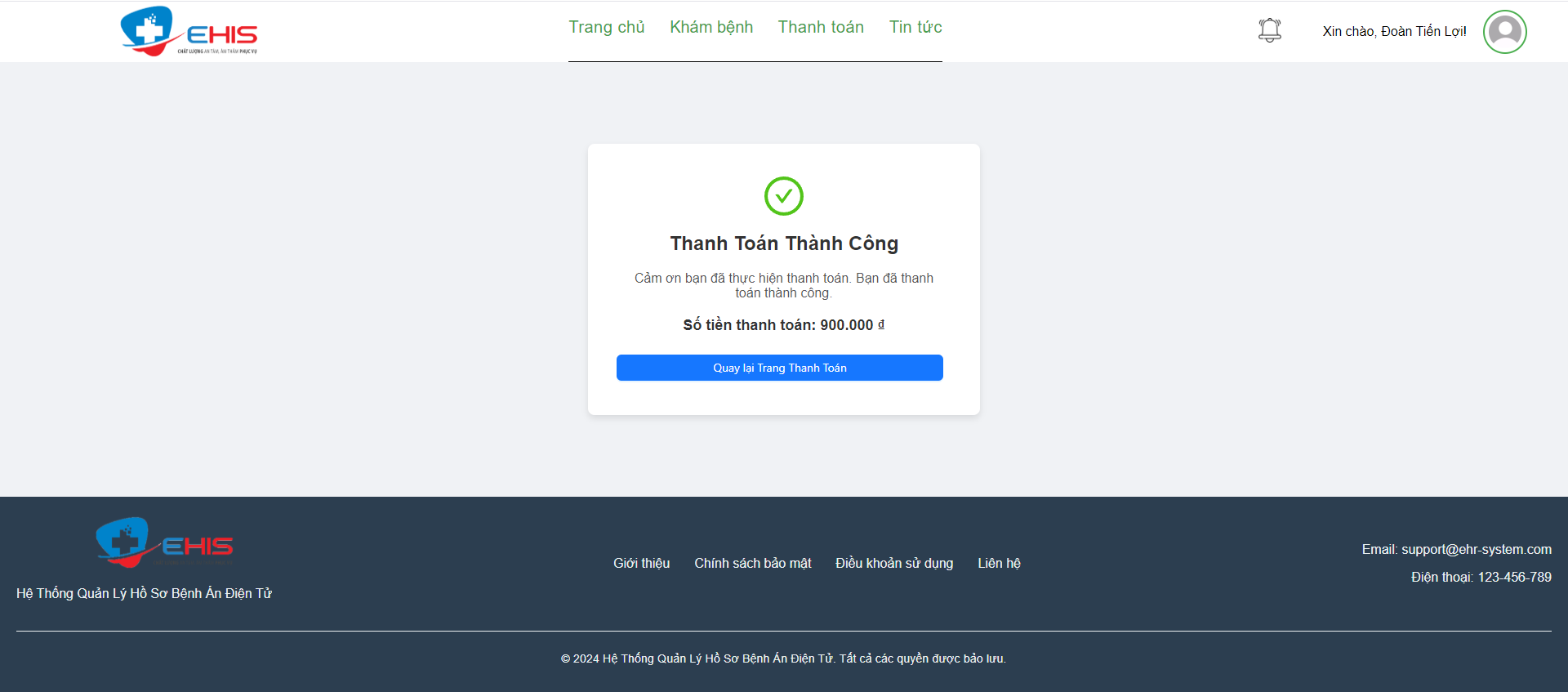
Người dùng chọn vào chức năng thanh toán. Giao diện trang thanh toán thành công zalo pay như Hình 3.26



# *Hình 3.26. Giao diện thanh toán zalo pay thành công*

#### **3.5.1.11 Giao diện trang thanh toán thành công**

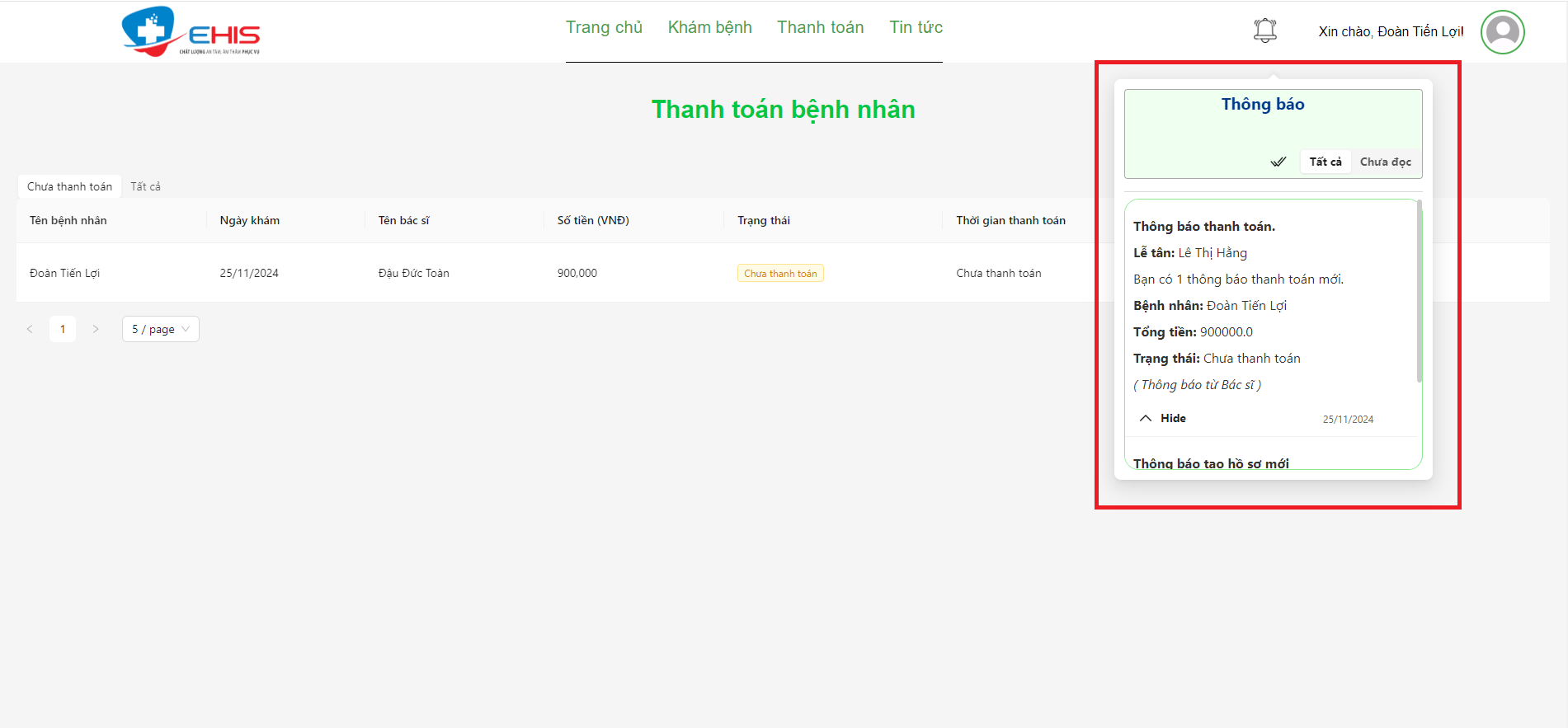
Sau khi người dùng xác nhận thanh toán zalop pay thành công. Giao diện trang thanh toán thành công như Hình 3.27



# *Hình 3.27. Giao diện thanh toán thành công*

#### **3.5.1.12 Giao diện chức năng thông báo**

Sau khi người dùng nhấn biểu tượng chuông. Giao diện trang thông báo chuông như Hình 3.28

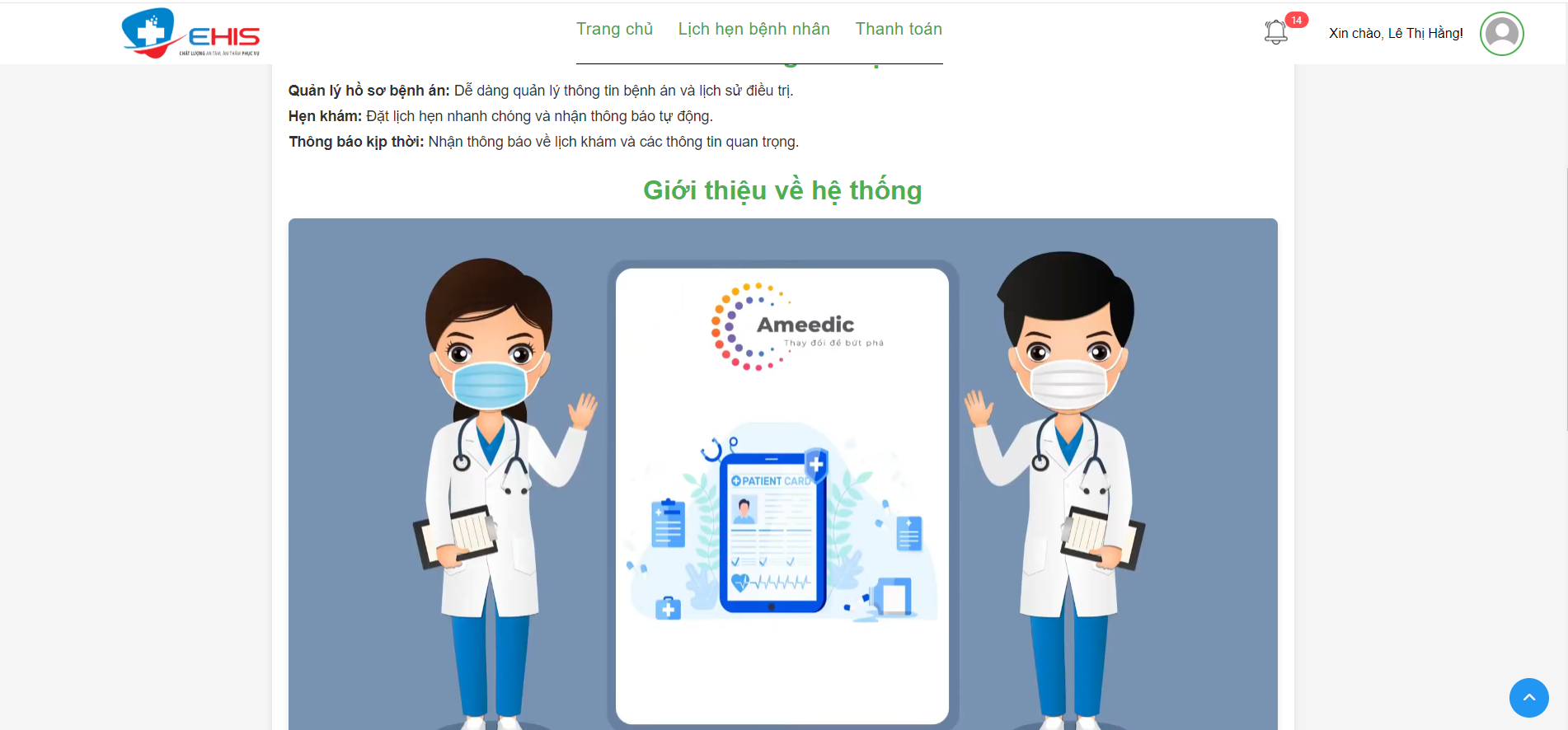


# *Hình 3.28. Giao diện chức năng thông báo*

### **Giao diện lễ tân**

#### **3.5.2.1 Giao diện trang chủ lễ tân**

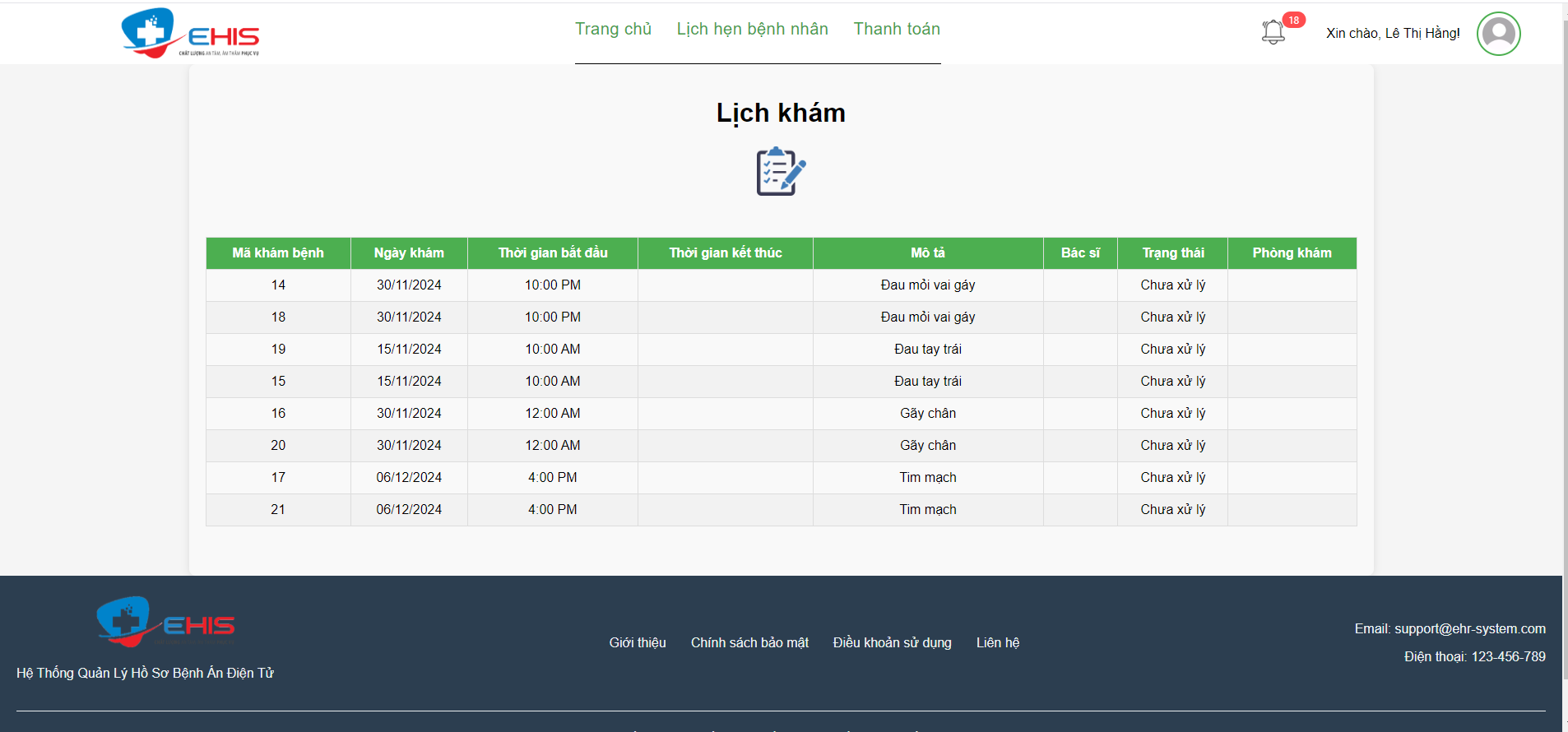
Sau khi Lễ Tân đăng nhập thành công. Giao diện trang chủ lễ tân như Hình 3.29



# *Hình 3.29. Giao diện chủ lễ tân*

#### **3.5.2.2 Giao diện lịch hẹn bệnh nhân**

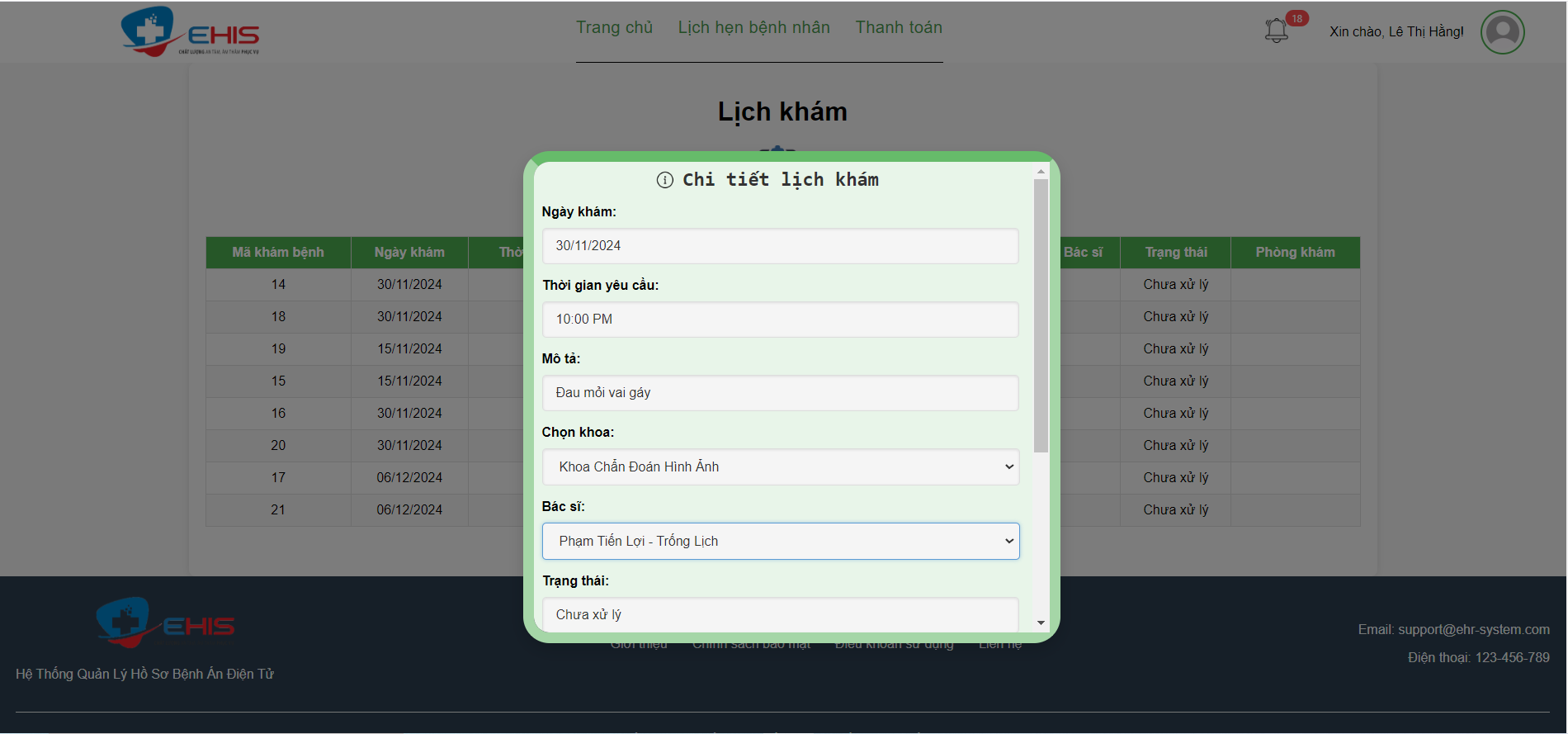
Sau khi Lễ Tân nhấn nút lịch hẹn bệnh nhân. Giao diện lịch hẹn bệnh nhân lễ tân như Hình 3.30



# *Hình 3.30. Giao diện lịch hẹn Lễ Tân*

#### **3.5.2.3 Giao diện chi tiết lịch khám**

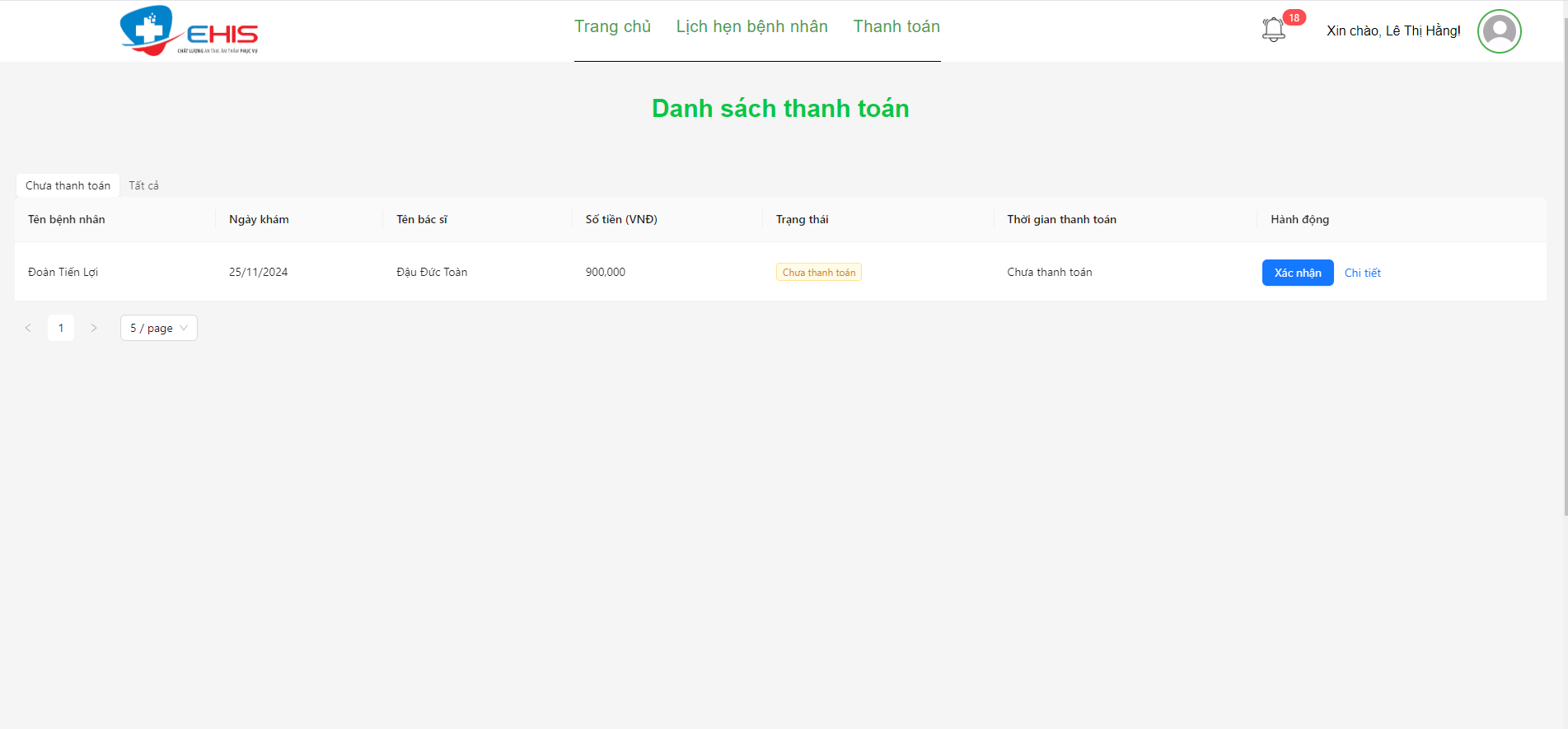
Sau khi Lễ Tân nhấn vào lịch hẹn bệnh nhân. Giao diện chi tiết lịch khám như Hình 3.31



# *Hình 3.31. Giao diện chi tiết lịch khám*

#### **3.5.2.4 Giao diện thanh toán lễ tân**

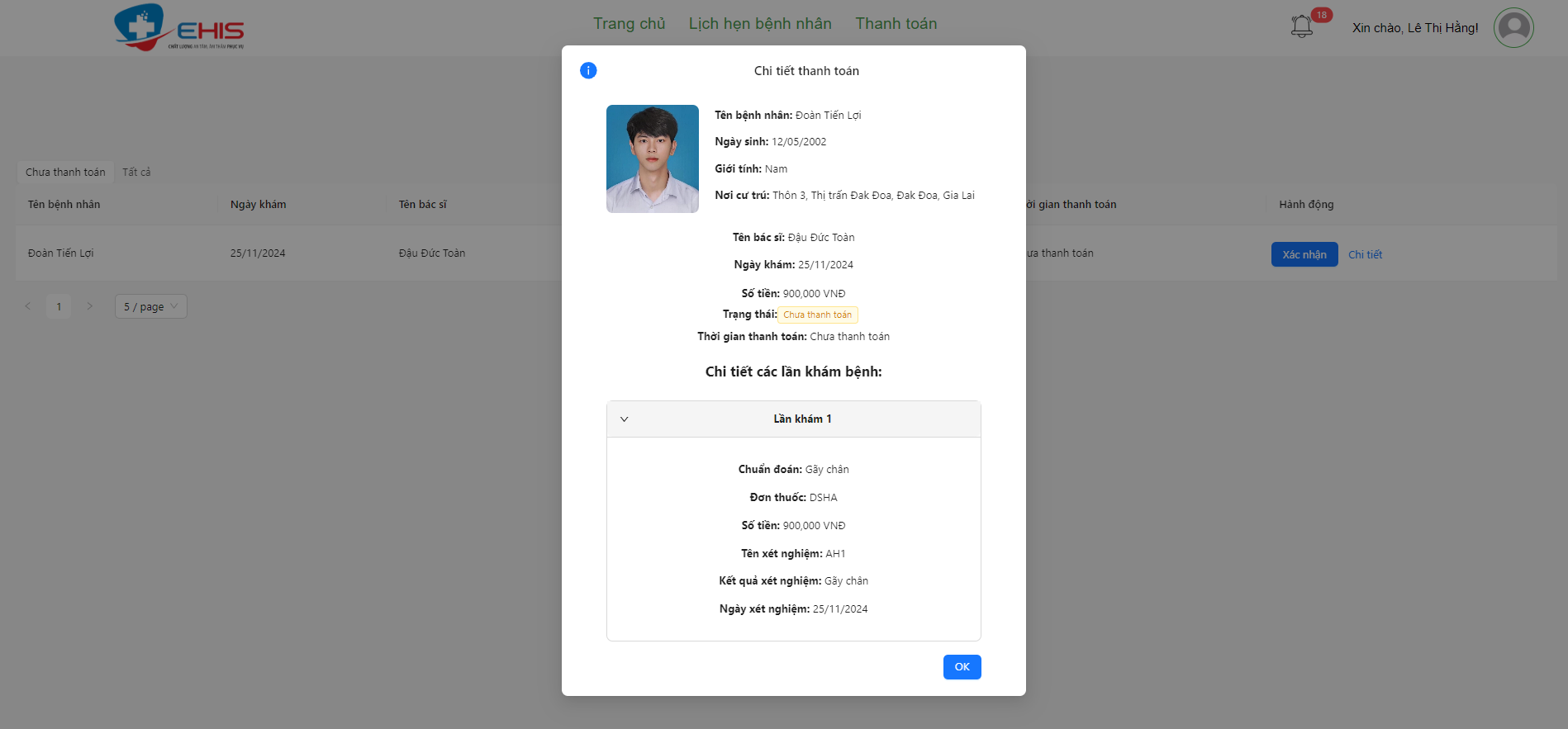
Sau khi Lễ Tân nhấn vào thanh toán. Giao diện thanh toán như Hình 3.32



# *Hình 3.32. Giao diện thanh toán lễ tân*

#### **3.5.2.5 Giao diện chi tiết thanh toán lễ tân**

Sau khi Lễ Tân nhấn vào chi tiết thanh toán. Giao diện chi tiết thanh toán như Hình 3.33



# *Hình 3.33. Giao diện chi tiết thanh toán lễ tân*

### **Giao diện bác sĩ**

#### **3.5.3.1 Giao diện trang chủ bác sĩ**

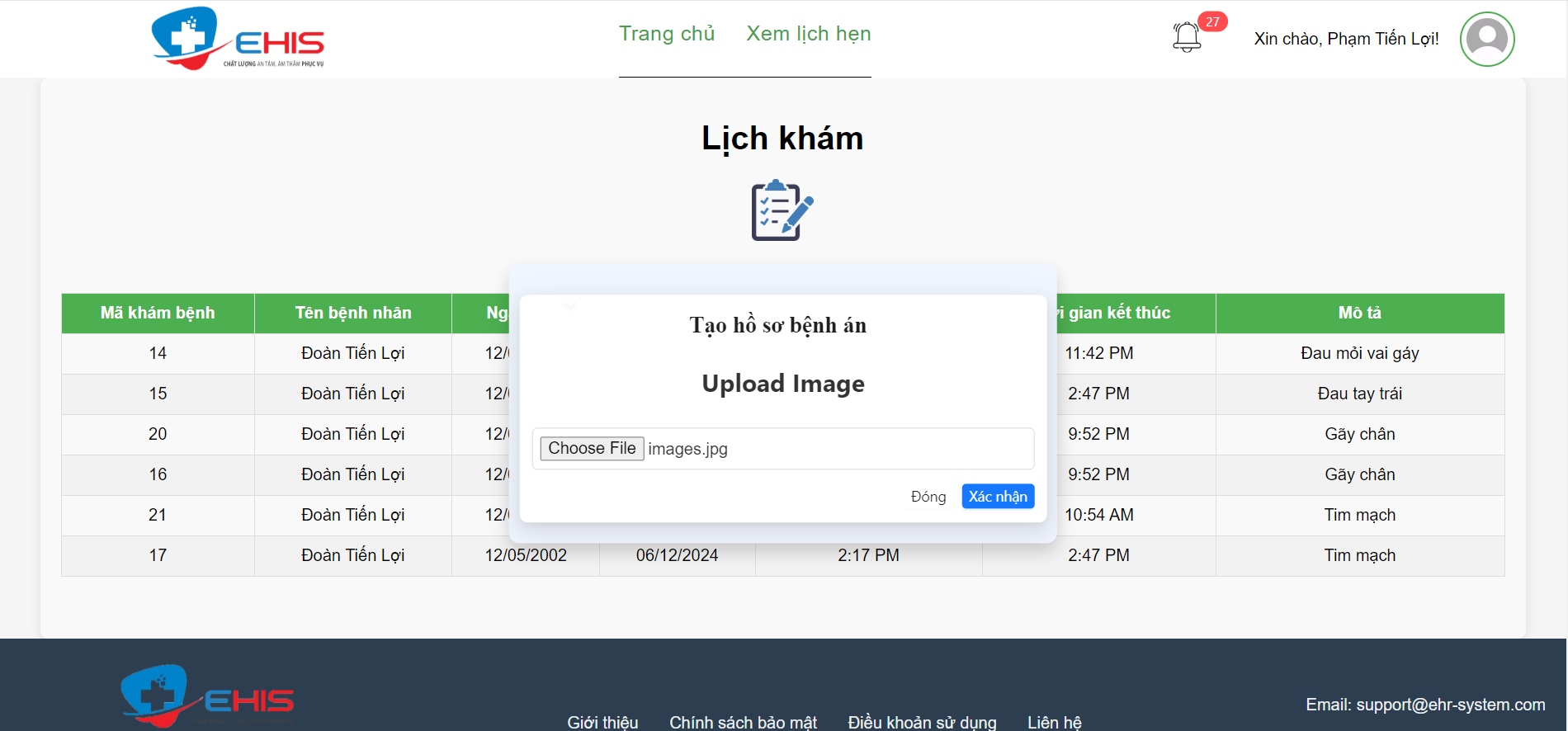
Sau khi Bác sĩ đăng nhập thành công. Giao diện trang chủ bác sĩ như Hình 3.34



# *Hình 3.34. Giao diện trang chủ bác sĩ*

#### **3.5.3.2 Giao diện xem lịch hẹn bác sĩ**

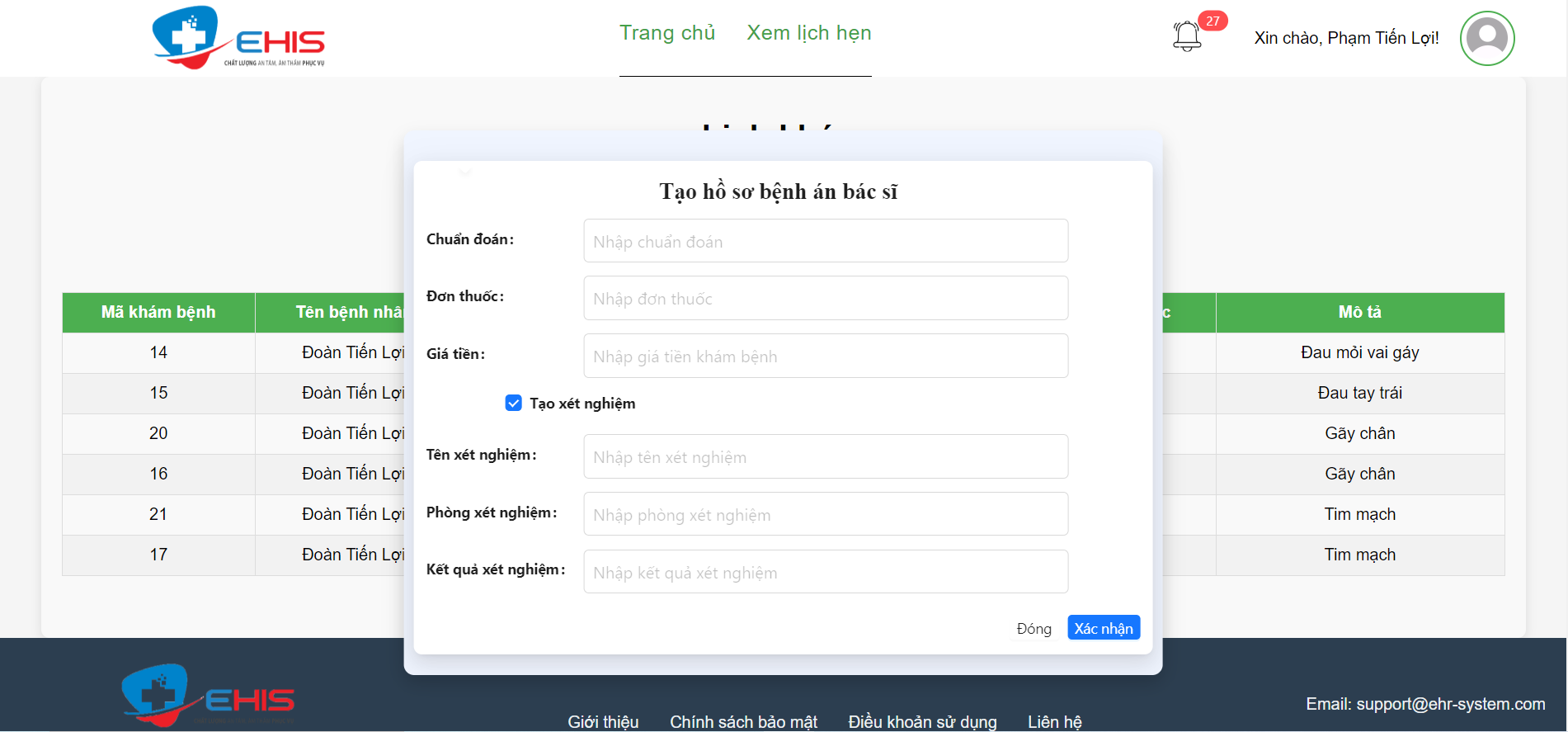
Sau khi Bác sĩ nhấn xem lịch hẹn. Giao diện xem lịch hẹn bác sĩ như Hình 3.35



# *Hình 3.35. Giao diện trang xem lịch hẹn bác sĩ*

#### **3.5.3.3 Giao diện trang tạo hồ sơ bệnh án bác sĩ**

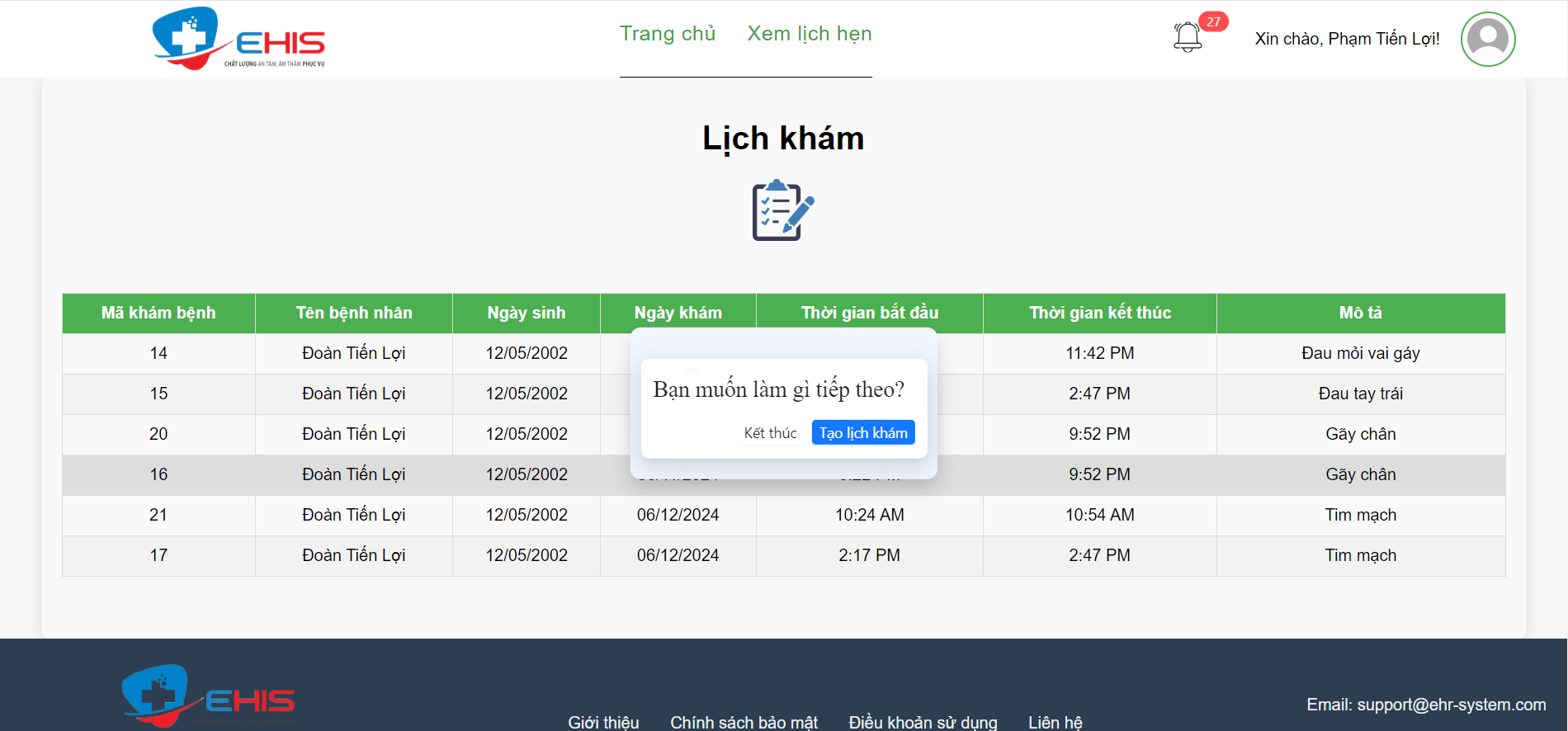
Sau khi Bác sĩ nhấn tạo hồ sơ bệnh án bác sĩ. Giao diện tạo hồ sơ bệnh án bác sĩ như Hình 3.36



# *Hình 3.36. Giao diện trang tạo hồ sơ bệnh án bác sĩ*

#### **3.5.3.4 Giao diện trang tạo lịch khám tiếp theo**

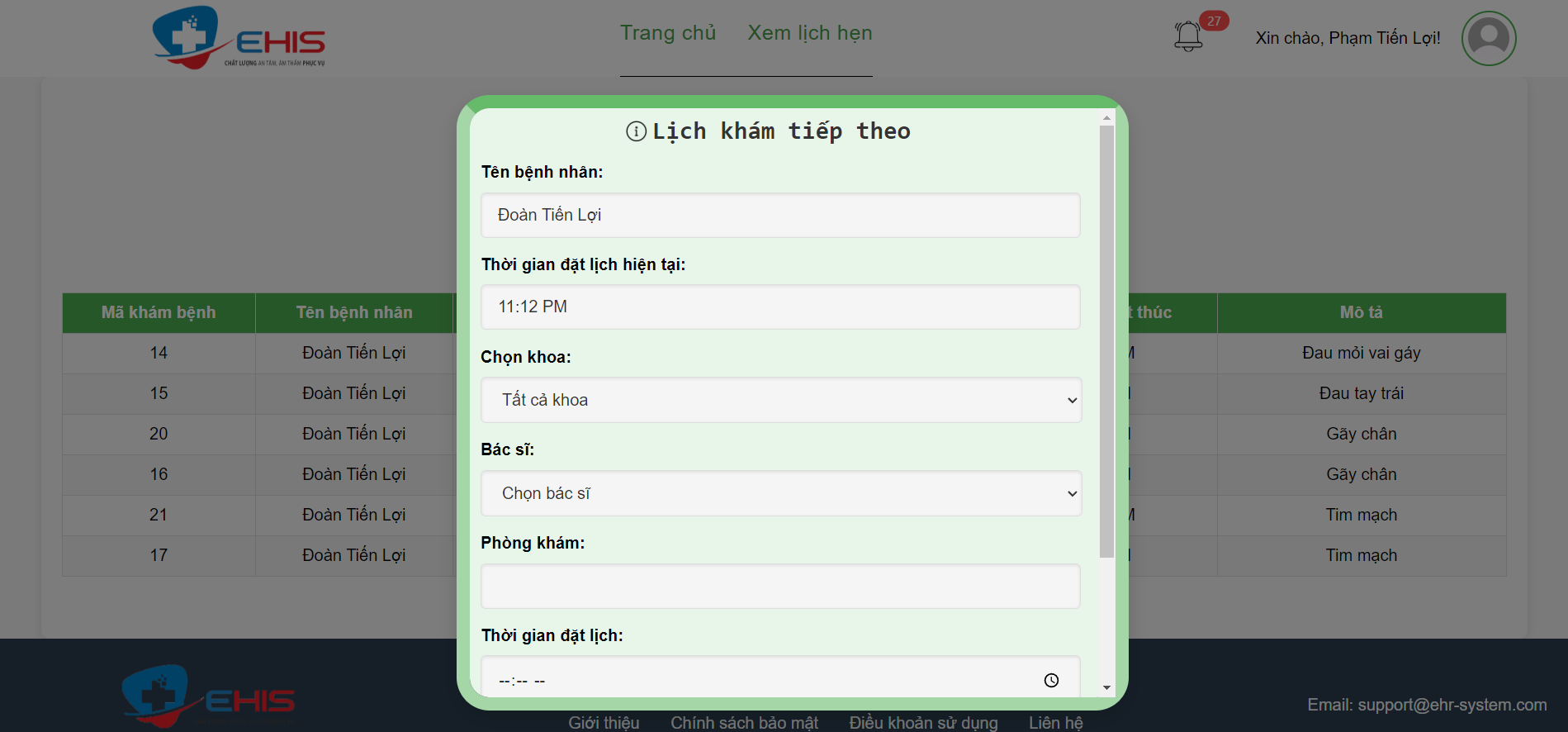
Sau khi Bác sĩ nhấn tạo lịch khám Giao diện tạo lịch khám như Hình 3.37



# *Hình 3.37. Giao diện trang tạo lịch khám tiếp theo*

#### **3.5.3.5 Giao diện trang tạo lịch khám tiếp theo**

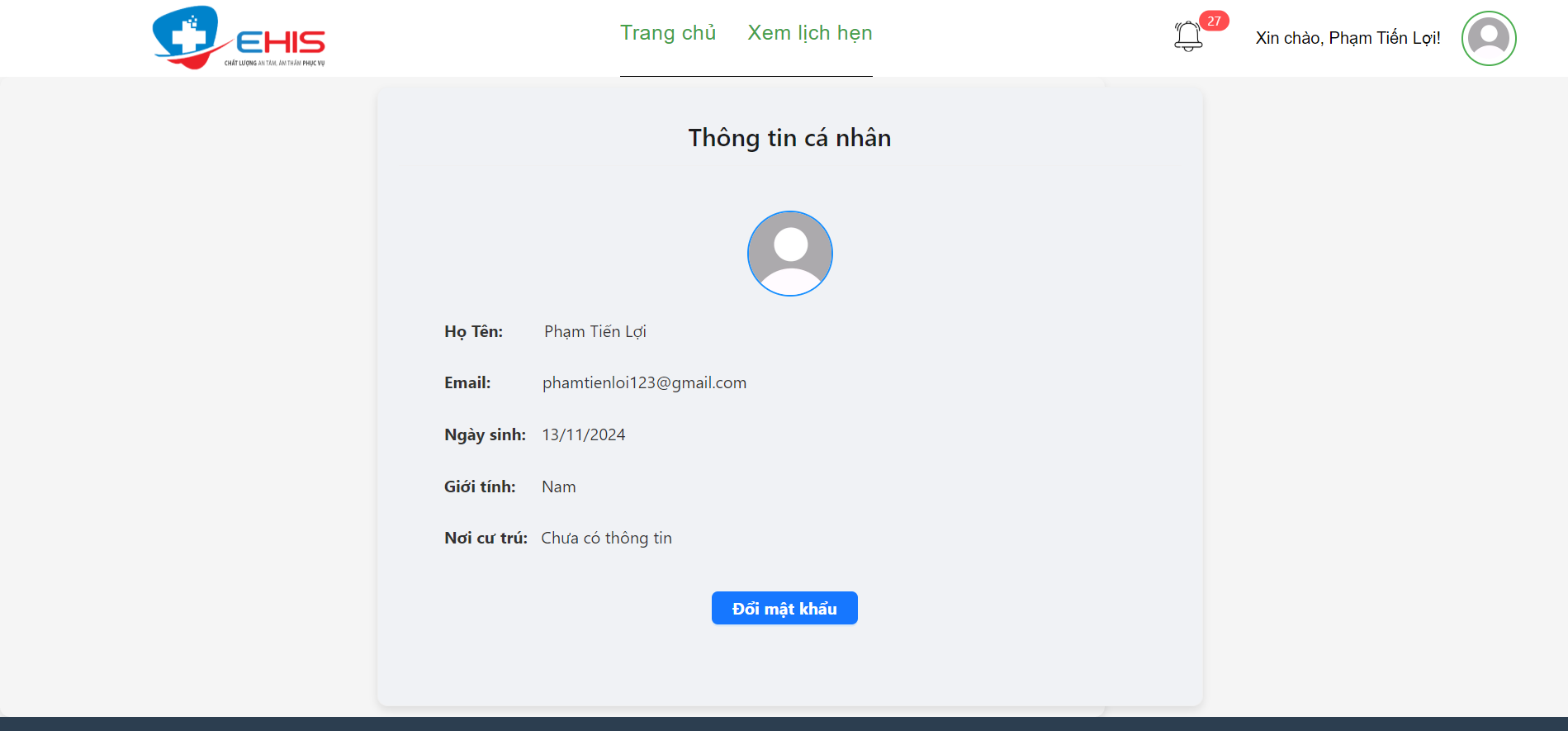
Sau khi Bác sĩ nhấn tạo lịch khám Giao diện lịch khám tiếp theo như Hình 3.38



# *Hình 3.38. Giao diện trang lịch khám tiếp theo*

#### **3.5.3.6 Giao diện trang thông tin cá nhân**

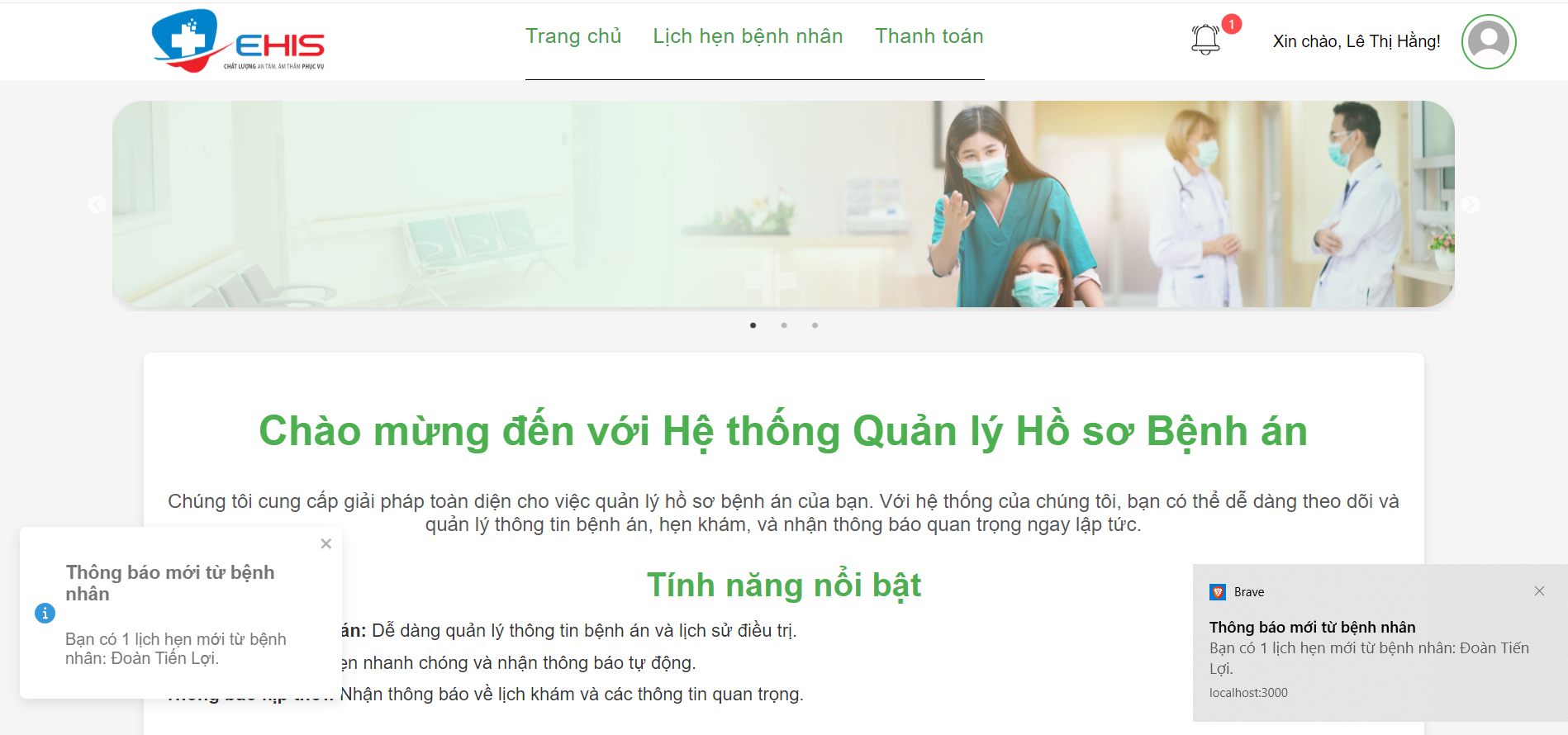
Sau khi Bác sĩ nhấn thông tin cá nhân Giao diện thông tin cá nhân như Hình 3.39



# *Hình 3.39. Giao diện trang thông tin cá nhân*

#### **3.5.3.7 Giao diện trang thông tin cá nhân**

Sau khi Bác sĩ nhấn thông tin cá nhân Giao diện thông tin cá nhân như Hình 3.40



# *Hình 3.40. Giao diện trang thông báo thời gian thực*

# **Chương 4:** **XÂY DỰNG WEBSITE QUẢN LÝ HỒ SƠ BỆNH ÁN**

## **4.1. Môi trường phát triển và công cụ lập trình**

### ***4.1.1. Cài đặt python***

- Mô tả: Python là ngôn ngữ lập trình được sử dụng để xây dựng phần backend của hệ thống. Với các framework mạnh mẽ như Django hoặc Flask, Python giúp dễ dàng tạo API, xử lý logic nghiệp vụ và kết nối cơ sở dữ liệu.

- Yêu cầu: Đảm bảo đang cài đặt phiên bản Python 3.8 hoặc cao hơn để tương thích với các thư viện và công nghệ mới.

- Hướng dẫn chi tiết:

* Tải xuống Python:
* Truy cập trang chủ Python.
* Chọn phiên bản phù hợp với hệ điều hành của bạn (Windows, macOS, Linux).
* Cài đặt Python:
* Chạy file cài đặt.
* Đánh dấu tùy chọn Add Python to PATH để dễ dàng sử dụng lệnh Python trong terminal.
* Kiểm tra cài đặt:
* Mở terminal và gõ lệnh:
* bash
* python --version
* Cài đặt pip:
* Pip là trình quản lý thư viện của Python.
* Nếu chưa có, cài đặt bằng lệnh:
* bash
* python -m ensurepip

### ***4.1.2. Cài đặt Pycharm***

- Mô tả: PyCharm là môi trường phát triển tích hợp (IDE) hỗ trợ tốt nhất cho việc viết code Python, với nhiều tính năng mạnh mẽ như kiểm tra lỗi, gợi ý cú pháp và tích hợp quản lý thư viện.

- Yêu cầu: Phiên bản Community (miễn phí) hoặc Professional (có phí).

- Hướng dẫn chi tiết:

* Tải xuống PyCharm:
* Truy cập JetBrains.
* Chọn phiên bản phù hợp và tải về.
* Cài đặt:
* Chạy file cài đặt và làm theo hướng dẫn trên màn hình.
* Liên kết với môi trường Python đã cài đặt bằng cách chọn interpreter trong Settings > Project > Python Interpreter.
* Cấu hình cơ bản:
* Tạo project mới và thêm các thư viện cần thiết thông qua terminal của PyCharm bằng lệnh pip install.

### ***4.1.3. Cài đặt ReactJs***

- Mô tả: React.Js là thư viện JavaScript phổ biến được dùng để xây dựng giao diện người dùng (frontend) với hiệu năng cao và khả năng tái sử dụng các thành phần (components).

- Yêu cầu: Cài đặt Node.js và npm (Node Package Manager).

- Hướng dẫn chi tiết:

* Cài đặt Node.js và npm:
* Truy cập Node.js và tải phiên bản LTS (Long Term Support).
* Kiểm tra cài đặt bằng lệnh:
* bash
* node --version
* npm --version
* Tạo project React:
* Mở terminal và gõ lệnh:
* bash
* npx create-react-app my-app
* cd my-app
* npm start
* Ứng dụng mẫu sẽ được khởi chạy trên trình duyệt.
* Cài đặt thư viện bổ sung:
* Các thư viện như React Router hoặc Axios có thể được cài đặt bằng lệnh:
* bash
* npm install react-router-dom axios

### ***4.1.4. Cài đặt PostgreSQL***

- Mô tả: PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ, hỗ trợ lưu trữ và truy xuất dữ liệu một cách hiệu quả.

- Hướng dẫn chi tiết:

* Tải xuống PostgreSQL:
* Truy cập PostgreSQL và chọn phiên bản phù hợp với hệ điều hành.
* Cài đặt:
* Chạy file cài đặt và thiết lập mật khẩu cho tài khoản postgres (superuser mặc định).
* Quản lý cơ sở dữ liệu:
* Sử dụng pgAdmin (công cụ GUI) hoặc terminal để tạo và quản lý cơ sở dữ liệu.
* Tạo cơ sở dữ liệu mới với lệnh:
* sql
* CREATE DATABASE project\_db;
* Kiểm tra kết nối:
* Dùng lệnh sau để kiểm tra:
* bash
* psql -U postgres -d project\_db

## **4.2. Xây dựng API cho hệ thống**

***4.2.1 Cấu trúc tổ chức thư mục của dự án***

* + Chức năng xác thực căn cước bằng mã QR
    - Vấn đề
      * Yêu cầu xác định danh tính người dùng trong các tình huống như đăng nhập, kiểm tra thông tin, hoặc làm thủ tục hành chính.
      * Giảm thiểu sự phụ thuộc vào các giấy tờ vật lý hoặc quy trình xác minh thủ công.
    - Giải pháp
      * Sử dụng các thư viện như OpenCV hoặc pyzbar để quét mã QR từ ảnh hoặc camera.
      * Giải mã dữ liệu và trích xuất thông tin dưới dạng các trường dữ liệu cần thiết (CCCD, họ tên, ngày sinh, v.v.).
      * Xác thực thông tin:
      * Đối chiếu dữ liệu giải mã được với cơ sở dữ liệu trung tâm (CSDL quốc gia, hoặc CSDL doanh nghiệp).
      * Nếu mã QR không hợp lệ, trả về thông báo cụ thể để người dùng có thể thử lại
    - Code minh họa:

import io  
import logging  
from typing import Any  
  
import cv2  
import numpy as np  
from PIL import Image  
from fastapi import APIRouter  
from fastapi import UploadFile, File  
from fastapi.responses import JSONResponse  
from pyzbar import pyzbar  
  
router = APIRouter()  
logger = logging.getLogger(\_\_name\_\_)  
  
  
@router.post("/scan-qrcode/")  
async def scan\_qrcode(file: UploadFile = File(...)):  
 contents = await file.read()  
 image = Image.open(io.BytesIO(contents))  
 image\_np = np.array(image)  
 sizes = [6500, 4500, 3200]  
 count = 0  
 info = None  
 while count < len(sizes) - 1:  
 origin\_img = cv2.cvtColor(image\_np, cv2.COLOR\_BGR2RGB)  
 origin\_img = resize\_image(image=origin\_img, new\_width=sizes[count])  
 qr\_detector = cv2.QRCodeDetector()  
 data, bbox, \_ = qr\_detector.detectAndDecode(origin\_img)  
 if bbox is not None:  
 qr\_code\_img = split\_qr\_code\_area(origin\_img=origin\_img, bbox=bbox)  
 decoded\_objects = pyzbar.decode(qr\_code\_img)  
 for obj in decoded\_objects:  
 data = obj.data.decode('utf-8')  
 info = extract\_qr\_info(data=data)  
 if info:  
 break  
 count += 1  
 if not info:  
 return JSONResponse(status\_code=400,  
 content={"message": "Please upload a photo with QRcode"})  
 return JSONResponse(status\_code=200, content={"data": info})  
  
def resize\_image(image: np.ndarray, new\_width: int) -> np.ndarray:  
 ratio = new\_width / image.shape[1]  
 new\_height = int(image.shape[0] \* ratio)  
 new\_size = (new\_width, new\_height)  
 new\_image = cv2.resize(image, new\_size, interpolation=cv2.INTER\_LINEAR)  
 return new\_image

def split\_qr\_code\_area(origin\_img: np.ndarray, bbox: Any) -> np.ndarray:  
 x, y, w, h = cv2.boundingRect(bbox)  
 image = origin\_img[y:y + h, x:x + w]  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(30, 30))  
 image = clahe.apply(gray)  
 return image  
  
def extract\_qr\_info(data: str):  
 if not data:  
 return None  
 data\_parts = data.split('|')  
 if len(data\_parts) == 7:  
 return {  
 "cccd\_id": data\_parts[0],  
 "cmnd\_id": data\_parts[1],  
 "full\_name": data\_parts[2],  
 "dob": format\_date\_str(data\_parts[3]),  
 "gender": data\_parts[4],  
 "residence": data\_parts[5],  
 "issuance": format\_date\_str(data\_parts[6]),  
 }  
 else:  
 return {  
 "cccd\_id": data\_parts[0],  
 "cmnd\_id": None,  
 "full\_name": data\_parts[1],  
 "dob": format\_date\_str(data\_parts[2]),  
 "gender": data\_parts[3],  
 "residence": data\_parts[4],  
 "issuance": format\_date\_str(data\_parts[5]),  
 }  
def format\_date\_str(date\_str: str):  
 return f"{date\_str[0:2]}/{date\_str[2:4]}/{date\_str[4:]}"

* + Chức năng gửi thông báo thời gian thực
    - Vấn đề
      * Yêu cầu thông báo đến người dùng trong thời gian ngắn nhất để đảm bảo tính thời gian thực, ví dụ: tin nhắn mới, cập nhật đơn hàng.
      * Hàng nghìn người dùng có thể đồng thời nhận thông báo, dẫn đến tải hệ thống cao và cần tối ưu để không bị nghẽn.
      * Thông báo phải được gửi đúng đối tượng mà không nhầm lẫn hoặc gửi tràn lan.
      * Đảm bảo kết nối thời gian thực an toàn, thông tin không bị lộ lọt hoặc giả mạo.
    - Giải pháp:
      * Pusher Channels là giải pháp thời gian thực giúp duy trì kết nối giữa client và server thông qua WebSocket, hỗ trợ gửi thông báo ngay lập tức.
      * Dễ tích hợp với FastAPI và các framework front-end như React, Vue, Angular.
      * Server sử dụng FastAPI để xử lý yêu cầu từ client và sử dụng thư viện pusher-http-python để gửi thông báo qua Pusher.
      * Client kết nối WebSocket với Pusher Channels để nhận thông báo.
      * Pusher quản lý tải kết nối, giảm gánh nặng cho server ứng dụng.
      * Hệ thống sử dụng hàng đợi như Redis hoặc RabbitMQ để xử lý thông báo theo hàng đợi và gửi đi lần lượt.
      * Sử dụng Token Authentication hoặc JWT để xác thực user trước khi kết nối với kênh Pusher.
      * Kênh riêng tư (private channels) để đảm bảo chỉ những user hợp lệ mới nhận được thông báo.
      * Mỗi người dùng kết nối vào một private channel riêng, giúp gửi thông báo cụ thể cho từng người mà không ảnh hưởng đến người khác.
      * Pusher đảm bảo kết nối không bị gián đoạn nhờ cơ chế tự động phục hồi khi mất kết nối.
    - Code minh họa
* @router.put("/{id}")  
  async def update(appointment\_data: AppointmentUpdateRequest,  
   id: int, session: AsyncSession = Depends(get\_async\_session),  
   user: User = Depends(get\_current\_active\_user),  
   ):  
   appointment\_service = AppointmentService(session=session)  
   appointment\_data = await appointment\_service.update(appointment\_data=appointment\_data, id=id)  
   await appointment\_service.update\_notification(data=appointment\_data, user=user)  
   return make\_response\_object(data=appointment\_data.dict())

async def update\_notification(self, data: Appointment, user):

doctor\_id = data.doctor\_id

patient\_id = data.patient\_id

doctor\_name = data.doctor.name

clinic\_location = data.doctor.clinic\_location

start\_date = convert\_datetime\_to\_date\_str(data.start\_time)

start\_time = convert\_datetime\_to\_time\_str(data.start\_time)

notification\_data =

UpdateAppointmentNotification(to\_notify\_users=[patient\_id, doctor\_id],  
 seen\_users=[],  
 title="Thông báo lịch hẹn mới",  
 description=f"Bạn có 1 lịch hẹn mới. Lịch hẹn này được tạo tự động bởi lễ tân.",  
 doctor\_name=doctor\_name,  
 clinic\_location=clinic\_location,  
 start\_date=start\_date,  
 start\_time=start\_time,  
 created\_at=datetime.now(),  
 updated\_at=datetime.now(),  
 )  
 notification\_dict = notification\_data.dict()  
 user\_ids = [patient\_id, doctor\_id]  
 collection = db[COLLECTION\_NAME]  
 insert\_result = collection.insert\_one(notification\_dict)  
 notification\_id = str(insert\_result.inserted\_id)  
 notification\_dict["\_id"] = notification\_id

send\_notification\_batch.delay(channels=user\_ids, notification\_data=notification\_dict)

@celery.task  
def send\_notification\_batch(channels: List[Union[int, str]], notification\_data: dict):  
 batch = []  
 for channel in channels:  
 # Tạo tên kênh cho người dùng  
 user\_channel = f'{channel}' if isinstance(channel, int) else channel  
  
 # Tạo thông tin dữ liệu thông báo từ notification\_data  
 notification\_content = {  
 '\_id': notification\_data.get('\_id'),  
 'title': notification\_data.get('title', 'Thông báo'),  
 'description': notification\_data.get('description', ''),  
 'patient\_name': notification\_data.get('patient\_name', ''),  
 'total\_payment': notification\_data.get('total\_payment', ''),  
 'receptionist\_name': notification\_data.get('receptionist\_name', ''),  
 'status\_payment': notification\_data.get('status\_payment', ''),  
 'doctor': {  
 'name': notification\_data.get('doctor\_name', ''),  
 'clinic\_location': notification\_data.get('clinic\_location', '')  
 },  
 'start\_date':notification\_data.get('start\_date', ''),  
 'start\_time':notification\_data.get('start\_time', ''),  
 }  
  
 # Thêm vào batch để gửi thông báo  
 batch.append({  
 'channel': user\_channel,  
 'name': "notification",  
 'data': notification\_content  
 })  
  
 try:  
 # Gửi thông báo theo dạng batch qua `pusher\_client`  
 pusher\_client.trigger\_batch(batch, False)  
 logger.info(f"Thông báo đã được gửi cho người dùng qua các kênh: {[item['channel'] for item in batch]}")  
 except Exception as e:  
 # Log chi tiết nếu có lỗi  
 logger.error(  
 f"Lỗi khi gửi thông báo cho các kênh: {[item['channel'] for item in batch]}, lỗi: {str(e)}",  
 exc\_info=True  
 )

* + Chức năng thanh toán Zalo Pay
    - Vấn đề
      * Làm thế nào để tích hợp Zalo Pay một cách an toàn và hiệu quả vào hệ thống ứng dụng?
      * Đảm bảo API của Zalo Pay hoạt động ổn định trong cả môi trường thử nghiệm và triển khai thực tế.
      * Làm thế nào để thông tin thanh toán như số tiền, mã giao dịch không bị giả mạo hoặc tấn công?
      * Làm thế nào để theo dõi chính xác trạng thái giao dịch từ Zalo Pay, bao gồm thành công, thất bại hoặc đang chờ xử lý?
      * Quá trình thanh toán cần mượt mà, dễ sử dụng, không gây rối hoặc mất thời gian cho người dùng.
    - Giải pháp
      * Sử dụng bộ công cụ SDK và tài liệu do Zalo Pay cung cấp để tích hợp thanh toán.
      * Xây dựng một REST API trung gian trong ứng dụng của bạn để gửi yêu cầu đến Zalo Pay và nhận phản hồi.
      * Mọi yêu cầu gửi đến Zalo Pay cần được ký bằng HMAC SHA-256 dựa trên API Key và Secret Key do Zalo Pay cung cấp.
      * Kiểm tra chữ ký số (signature) trong phản hồi từ Zalo Pay để xác thực tính hợp lệ của dữ liệu.
      * Lưu trữ trạng thái giao dịch trong cơ sở dữ liệu của bạn để dễ dàng tra cứu và kiểm tra.
    - Code minh họa:
* @router.post("/zalopay/create-payment")  
  async def create\_zalopay\_payment(payment: PaymentRequest):  
   async with httpx.AsyncClient() as client:  
   try:  
   app\_id = 2553  
   key1 = "PcY4iZIKFCIdgZvA6ueMcMHHUbRLYjPL"  
   endpoint = "https://sb-openapi.zalopay.vn/v2/create"  
   transID = random.randrange(1000000)  
   order = {  
   "app\_id": app\_id,  
   "app\_trans\_id": "{:%y%m%d}\_{}".format(datetime.today(), transID),  
   "app\_user": f"{payment.payment\_id}",  
   "app\_time": int(round(time.time() \* 1000)),  
   "embed\_data": json.dumps({'redirecturl': 'http://localhost:3000/thanh-toan-thanh-cong'}),  
   "item": json.dumps([{}]),  
   "amount": int(payment.amount),  
   "description": "Thanh toán tiền viện phí",  
   "bank\_code": "CC",  
   "callback\_url": 'https://325d-2402-800-6205-55c9-c9e8-3579-4319-d446.ngrok-free.app/api/payments/callback/',  
   }  
   mac\_data = f"{order['app\_id']}|{order['app\_trans\_id']}|{order['app\_user']}|{order['amount']}|{order['app\_time']}|{order['embed\_data']}|{order['item']}"  
   order["mac"] = hmac.new(key1.encode(), mac\_data.encode(), hashlib.sha256).hexdigest()  
   try:  
   response = requests.post(endpoint, json=order)  
   response.raise\_for\_status()  
   response\_data = response.json()  
   if response\_data.get('order\_url'):  
   return {"zalopay\_url": response\_data['order\_url']}  
   else:  
   raise HTTPException(status\_code=500, detail="Không nhận được URL từ ZaloPay")  
   except requests.exceptions.RequestException as e:  
   raise HTTPException(status\_code=500, detail="Không thể kết nối đến API ZaloPay")  
   except Exception as e:  
   logger.exception(f"Exception occurred: {str(e)}")  
   raise HTTPException(status\_code=500, detail=str(e))
* @router.post("/callback")  
  async def callback\_zalopay(request: Request, session: AsyncSession = Depends(get\_async\_session)):  
   try:  
   body = await request.json()  
   logger.info(f"Received callback with body: {body}")  
    
   data = json.loads(body.get('data', '{}'))  
   logger.info(f"Parsed data: {data}")  
    
   amount = data.get("amount")  
   payment\_id = data.get("app\_user")  
   payment\_id = int(payment\_id)  
    
   logger.info(f"Payment success: amount={amount}, payment\_id={payment\_id}")  
    
   payment = await payment\_crud.get(session, Payment.id == payment\_id)  
   if not payment:  
   logger.error(f"Payment record not found for payment\_id: {payment\_id}")  
   raise HTTPException(status\_code=404, detail="Không tìm thấy thanh toán")  
    
   payment\_date = datetime.utcnow()  
    
   payment\_date\_vn = payment\_date + timedelta(hours=7)  
   payment = await payment\_crud.update(  
   session,  
   obj\_in=PaymentUpdate(status=StatusPaymentEnum.COMPLETED, payment\_date=payment\_date\_vn),  
   db\_obj=payment,  
   )  
   payment\_amount = payment.amount  
   medical\_record\_id = payment.medical\_record\_id  
   medical\_record = await medical\_record\_crud.get(session, MedicalRecord.id == medical\_record\_id )  
   patient\_name = medical\_record.patient.name  
    
   receptionist = await user\_crud.get(session, User.role == RoleEnum.RECEPTIONIST)  
   receptionist\_id = receptionist.id  
   notification\_data = UpdateAppointmentNotification(to\_notify\_users=[receptionist\_id],  
   seen\_users=[],  
   title="Thông báo thanh toán thành công",  
   description=f"Thanh toán thành công số tiền: {payment\_amount} từ bệnh nhân {patient\_name}.",  
   created\_at=datetime.now(),  
   updated\_at=datetime.now(),  
   )  
   notification\_dict = notification\_data.dict()  
   user\_ids = [receptionist\_id]  
   collection = db[COLLECTION\_NAME]  
   insert\_result = collection.insert\_one(notification\_dict)  
   notification\_id = str(insert\_result.inserted\_id)  
   notification\_dict["\_id"] = notification\_id # Thêm ID vào dữ liệu gửi đi  
   send\_notification\_batch.delay(channels=user\_ids, notification\_data=notification\_dict)  
   logger.info(f"Updated payment status to COMPLETED for payment\_id: {payment\_id}")  
   return {"status": "success", "message": f"Thanh toán {payment\_id} thành công: {amount}"}  
   except Exception as e:  
   logger.exception(f"Exception occurred while processing callback: {str(e)}")  
   raise HTTPException(status\_code=500, detail=f"Lỗi khi xử lý callback: {str(e)}")
  + Chức năng phân quyền OPA (Open Policy Agent)
    - Vấn đề
      * Việc quản lý quyền truy cập dựa trên vai trò hay yêu cầu cần trở nên linh hoạt và có thể thay đổi dễ dàng, đặc biệt trong các ứng dụng phức tạp.
      * Hệ thống phân quyền cần có khả năng mở rộng để thêm hoặc thay đổi các quy tắc phân quyền mà không gây ảnh hưởng đến các phần khác của ứng dụng.
      * Các chính sách phân quyền cần được định nghĩa và áp dụng một cách tập trung cho toàn bộ hệ thống, đặc biệt là khi ứng dụng có nhiều dịch vụ và nhiều môi trường khác nhau.
    - Giải pháp
      * OPA là một công cụ mã nguồn mở cho phép định nghĩa các chính sách phân quyền linh hoạt, có thể áp dụng cho nhiều loại ứng dụng và dịch vụ khác nhau.
      * Chính sách phân quyền được viết bằng ngôn ngữ Rego, giúp dễ dàng thay đổi, mở rộng mà không thay đổi mã nguồn của ứng dụng.
      * Dữ liệu về quyền truy cập và các yêu cầu người dùng sẽ được gửi đến OPA để kiểm tra tính hợp lệ trước khi quyết định cho phép hay từ chối.
      * OPA sẽ nhận đầu vào là các thông tin người dùng và yêu cầu (như URL, phương thức HTTP, vai trò người dùng, v.v.), sau đó quyết định xem yêu cầu có được phép thực hiện hay không.
    - Code minh họa
* import logging  
  from enum import Enum  
  from typing import Union, List  
    
  from fastapi import Request  
  from sqlalchemy.ext.asyncio import AsyncSession  
    
  from app.constant import BasePath  
  from app.models import User  
  from app.opa.permissions.base\_permission import OpenPolicyAgentPermission  
  from app.utils.decorator import add\_members\_from  
    
  logger = logging.getLogger(\_\_name\_\_)  
    
    
  class AppointmentPermission(OpenPolicyAgentPermission):  
   def \_\_init\_\_(self, \*\*kwargs):  
   super().\_\_init\_\_(\*\*kwargs)  
   self.opa\_url = self.opa\_url + '/appointment/result'  
    
   @add\_members\_from(OpenPolicyAgentPermission.Scopes)  
   class AppointmentScopes(Enum):  
   pass  
    
   @classmethod  
   async def create(cls, request: Request, session: AsyncSession, user: Union[User, None]) \  
   -> List[OpenPolicyAgentPermission]:  
   permissions = []  
   if request.url.path.startswith(f'{BasePath}/appointment') and request.method == "POST":  
   self = cls.create\_base\_perm(scope=cls.AppointmentScopes.CREATE.value, user\_id=user.id,  
   role=user.role)  
   await self.get\_resource(session=session)  
   permissions.append(self)  
   return permissions  
    
   async def get\_resource(self, session: AsyncSession, \*\*kwargs):  
   switch\_scope = {  
   self.AppointmentScopes.CREATE.value: self.handle\_create\_scope,  
   self.AppointmentScopes.LIST.value: self.handle\_list\_scope,  
   self.AppointmentScopes.READ.value: self.handle\_read\_scope,  
   self.AppointmentScopes.UPDATE.value: self.handle\_update\_scope,  
   self.AppointmentScopes.DELETE.value: self.handle\_delete\_scope,  
   }  
    
   handler = switch\_scope.get(self.scope)  
   await handler(session=session, \*\*kwargs)

package appointment  
  
import future.keywords.if  
import future.keywords.contains  
import data.utils  
  
default allow = false  
  
allow {  
 utils.is\_patient  
 input.scope == utils.CREATE  
}  
reasons contains "Tài nguyên được yêu cầu thuộc về bệnh nhân" if {  
 { utils.CREATE }[input.scope]  
 input.scope == utils.CREATE  
 not utils.is\_patient  
}  
  
result := {  
 "allow": allow,  
 "reasons": reasons  
}

* + Chức năng xử lý tác vụ bất đồng bộ celery worker
    - Vấn đề
      * Xử lý tác vụ dài hạn làm gián đoạn trải nghiệm người dùng
      * Khi hệ thống phải thực hiện các tác vụ lâu dài, chẳng hạn như gửi email, xử lý dữ liệu, tạo báo cáo, hoặc kết nối với các dịch vụ bên ngoài, chúng có thể làm giảm hiệu suất của ứng dụng
      * Các tác vụ có thể phức tạp, yêu cầu tái thực hiện (retry), chia nhỏ (split), hoặc được xâu chuỗi (chain) lại với nhau. Cần có một giải pháp để quản lý và thực thi hiệu quả các tác vụ này mà không làm tắc nghẽn hệ thống chính.
    - Giải pháp
      * Celery là một thư viện Python mạnh mẽ hỗ trợ xử lý các tác vụ bất đồng bộ. Celery giúp chuyển các tác vụ dài và tốn thời gian vào các worker riêng biệt để không làm gián đoạn quá trình xử lý chính của ứng dụng.
      * Celery có thể tích hợp với các message brokers như Redis hoặc RabbitMQ để truyền tải các tác vụ giữa các worker và ứng dụng chính.
      * Quản lý tác vụ: Celery hỗ trợ các tính năng như retry (thực hiện lại tác vụ khi thất bại), chaining (xâu chuỗi các tác vụ), và grouping (thực thi các tác vụ song song).
    - Code minh họa
* import logging  
  from typing import Union, List  
    
  from celery import Celery  
  from pusher import Pusher  
    
  from app.core import settings  
    
  logger = logging.getLogger(\_\_name\_\_)  
    
    
  def make\_celery():  
   celery = Celery(  
   \_\_name\_\_,  
   backend=settings.CELERY\_RESULT\_BACKEND,  
   broker=settings.CELERY\_BROKER\_URL  
   )  
   return celery  
    
    
  celery = make\_celery()  
    
  # Khởi tạo Pusher client  
  pusher\_client = Pusher(  
   app\_id=settings.PUSHER\_APP\_ID,  
   key=settings.PUSHER\_KEY,  
   secret=settings.PUSHER\_SECRET,  
   cluster=settings.PUSHER\_CLUSTER,  
   ssl=True  
  )

# **KẾT LUẬN**

1. KẾT QUẢ ĐẠT ĐƯỢC

1. Phía Bệnh nhân:

**- Đặt lịch khám bệnh:** Bệnh nhân có thể dễ dàng chọn ngày, giờ khám phù hợp.

**- Xem lịch hẹn:** Cung cấp giao diện hiển thị chi tiết lịch hẹn.

**- Thanh toán online:** Bệnh nhân có thể thanh toán hóa đơn liên quan đến khám bệnh trực tuyến.

- Thông tin hồ sơ bệnh án và xét nghiệm

- Hiển thị chi tiết lịch sử bệnh án và các kết quả xét nghiệm.

- Thông báo thời gian thực qua **Pusher** và lưu trữ thông báo trong **MongoDB**.

- Các thông báo được lưu lại và hiển thị trong giao diện **Notification**, ngay cả khi Pusher không hoạt động.

**- Đăng ký bằng quét mã căn cước công dân:** Hỗ trợ đăng ký tài khoản bệnh nhân nhanh chóng mà không cần nhập liệu thủ công.

1. Phía Lễ tân:

- Đặt lịch cho bệnh nhân với sự ưu tiên Bác sĩ có lịch rảnh nếu không, chọn bác sĩ có thời gian rảnh sớm nhất.

- Hiển thị danh sách thanh toán đã hoàn thành, chưa thanh toán, và toàn bộ thanh toán.

- Thanh toán cho bệnh nhân trực tiếp tại quầy.

- Nhận thông báo từ bác sĩ hoặc bệnh nhân để xử lý kịp thời các yêu cầu.

1. Phía Bác sĩ:

- Tạo hồ sơ bệnh án mới, thêm thông tin về chẩn đoán và xét nghiệm.

- Đặt lịch cho bệnh nhân với bác sĩ khác nếu bệnh nhân cần theo dõi thêm.

- Gửi thông báo đến lễ tân để yêu cầu thanh toán.

1. Công nghệ và kỹ thuật:

**- Thông báo thời gian thực:** Sử dụng **Pusher** và **Celery Worker** để gửi thông báo bất đồng bộ.

**- Quản lý phân quyền:** Sử dụng **OPA (Open Policy Agent)** để kiểm tra quyền truy cập linh hoạt.

**- Xử lý dữ liệu bất đồng bộ:** Sử dụng async/await để tương tác với cơ sở dữ liệu, đảm bảo hiệu năng cao.

1. HẠN CHẾ

1. Chức năng chưa hoàn thiện:

- Một số tính năng quản trị chưa được phát triển, như dashboard tổng quan cho admin.

- Thiếu chức năng phân tích dữ liệu chuyên sâu (ví dụ: thống kê lịch sử khám bệnh, doanh thu, v.v.).

2. Hiệu năng:

- Hệ thống có thể gặp khó khăn khi xử lý lượng dữ liệu lớn (chưa tối ưu hóa cho dữ liệu thời gian thực).

1. Bảo mật:

- Quản lý đăng ký bằng quét mã căn cước cần đảm bảo tính bảo mật và chống giả mạo.

- Hệ thống thông báo qua Pusher phụ thuộc nhiều vào dịch vụ bên ngoài.

1. Trải nghiệm người dùng (UI/UX):

- Giao diện chưa thân thiện hoàn toàn với mọi đối tượng người dùng (ví dụ: người lớn tuổi).

1. Thanh toán online có thể gặp hạn chế khi kết nối với các cổng thanh toán khác nhau.
2. HƯỚNG PHÁT TRIỂN

1. Phát triển tính năng mới:

- Thêm tính năng dashboard tổng quan cho quản trị viên.

- Phát triển module báo cáo và thống kê cho các vai trò: **Bệnh nhân:** Thống kê lịch sử khám, **Lễ tân:** Thống kê lịch hẹn và thanh toán, **Bác sĩ:** Thống kê bệnh nhân theo khoa, lịch sử chẩn đoán

- Hỗ trợ thêm ngôn ngữ hoặc tính năng trợ lý ảo để hướng dẫn bệnh nhân.

2. Tối ưu hóa hiệu năng:

- Áp dụng caching với **Redis** để lưu trữ tạm thời thông tin lịch hẹn và thông báo.

- Sử dụng **WebSocket** thay thế **Pusher** để giảm phụ thuộc vào bên thứ ba.

3. Nâng cao bảo mật:

- Tăng cường bảo mật dữ liệu bằng mã hóa (AES/RSA) cho thông tin nhạy cảm như mã căn cước và thanh toán.

- Tích hợp xác thực hai yếu tố (2FA) cho tài khoản bệnh nhân.

4. Cải thiện trải nghiệm người dùng:

- Thiết kế giao diện thân thiện với mọi độ tuổi và tối ưu cho di động.

- Thêm thông báo qua email hoặc SMS để mở rộng kênh thông tin.

5. Tích hợp AI và Machine Learning:

- Gợi ý bác sĩ hoặc xét nghiệm phù hợp dựa trên bệnh sử.

- Phân tích lịch sử bệnh án để cảnh báo sớm nguy cơ sức khỏe.

6. Mở rộng tính năng thanh toán:

- Hỗ trợ nhiều cổng thanh toán quốc tế như PayPal, Stripe.

- Tích hợp thanh toán QR code cho bệnh nhân thanh toán trực tiếp tại quầy.

# **TÀI LIỆU THAM KHẢO**

[1] Kuhlman, Dave. “A Python Book: Beginning Python, Advanced Python, and Python Exercises”. Section 1.1. Bản gốc (PDF) lưu trữ ngày 23 tháng 6 năm 2012.

[2] "FastAPI". fastapi.tiangolo.com. Truy cập ngày 2024-04-10.

[3] Lubanovic, Bill (2019-11-06). Introducing Python: Modern

[4] “React - A JavaScript library for building user interfaces”. reactjs.org (bằng tiếng Anh). Truy cập ngày 7 tháng 4 năm 2018.

[5] Krill, Paul (15 tháng 5 năm 2014). “React: Making faster, smoother UIs for data-driven Web apps”. InfoWorld. Truy cập ngày 23 tháng 2 năm 2021.

[6] Hemel, Zef (3 tháng 6 năm 2013). “Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews”. infoq.com (bằng tiếng Anh). Truy cập ngày 11 tháng 1 năm 2022.

[7] Dawson, Chris (25 tháng 7 năm 2014). “JavaScript's History and How it Led To ReactJS”. The New Stack (bằng tiếng Anh). Truy cập ngày 19 tháng 7 năm 2020.

[8] Dere, Mohan (19 tháng 2 năm 2018). “How to integrate create-react-app with all the libraries you need to make a great app”. freeCodeCamp (bằng tiếng Anh). Truy cập ngày 14 tháng 6 năm 2018.

[9] Panchal, Krunal (19 tháng 2 năm 2018). “Angular vs React Detailed Comparison”. Groovy Web (bằng tiếng Anh). Truy cập ngày 25 tháng 4 năm 2022.

[10] ["Restarting 'uvicorn' Workers with the 'SIGHUP' Signal"](https://bugfactory.io/articles/restarting-uvicorn-workers-with-the-sighup-signal/). *bugfactory.io*. Truy cập ngày 2024-06-17.

[11] O'Gara, Maureen (July 26, 2013). "Ben Golub, Who Sold Gluster to Red Hat, Now Running dotCloud". SYS-CON Media. Archived from the original on September 13, 2019.

[12] "What is a Container?". docker.com. Docker, Inc. Retrieved May 13, 2019.

[13] Ratan, Vivek (February 8, 2017). "Docker: A Favourite in the DevOps World". Open Source For U. Retrieved June 14, 2017.

[14] “GitHub Press Info”. github.com. Github. Truy cập ngày 15 tháng 11 năm 2015.

[15] "Celery: Distributed Task Queue". Archived from the original on 2019-06-13. Retrieved 2016-01-14.

[16] "Eventlet Networking Library". eventlet.net. Retrieved 2022-09-21.

[17] "What is gevent? — gevent 21.12.1.dev0 documentation". gevent.org. Retrieved 2022-09-21.

[18] Wikipedia. "JSON Web Token." Truy cập ngày 2024-11-26