

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

XÂY DỰNG WEBSITE QUẢN LÝ BỆNH NHÂN CHO TRUNG TÂM Y TẾ MEDICALHOPE TÍCH HỢP CHATBOT AI

Người hướng dẫn: TS. LÝ QUỲNH TRẦN

Sinh viên thực hiện: VÕ TRỌNG QUYỀN

Mã số sinh viên: 2151220090

Lớp: 21CT2

Đà Nẵng 12/2025

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**XÂY DỰNG WEBSITE QUẢN LÝ BỆNH
NHÂN CHO TRUNG TÂM Y TẾ
MEDICALHOPE TÍCH HỢP CHATBOT AI**

Người hướng dẫn: **TS. LÝ QUỲNH TRÂN**

Sinh viên thực hiện: **VÕ TRỌNG QUYỀN**

Mã số sinh viên: **2151220090**

Lớp: **21CT2**

Đà Nẵng 12/2025

TÓM TẮT

Tên đề tài: Xây dựng website quản lý bệnh nhân cho trung tâm y tế MEDICALHOPE tích hợp chatbot AI.

Sinh viên thực hiện: Võ Trọng Quyền, MSSV: 2151220090.

MEDICALHOPE là một hệ thống hỗ trợ khám chữa bệnh thiện nguyện, đóng vai trò như cầu nối giữa bệnh nhân có hoàn cảnh khó khăn, bác sĩ tình nguyện và các nhà hảo tâm. Nền tảng cho phép bệnh nhân đăng ký hồ sơ bệnh và hoàn cảnh để được duyệt và nhận hỗ trợ y tế miễn phí từ đội ngũ bác sĩ phù hợp. Nhà hảo tâm có thể dễ dàng xem danh sách bệnh nhân hoặc các dự án cần giúp đỡ, thực hiện quyên góp tiền hoặc hiện vật và theo dõi quá trình hỗ trợ một cách minh bạch. Đối với bác sĩ, hệ thống cung cấp không gian để đăng ký tham gia khám chữa bệnh thiện nguyện và tiếp cận hồ sơ bệnh phù hợp với chuyên môn của mình.

Bên cạnh đó, MEDICALHOPE còn tích hợp cơ chế quản lý cho admin nhằm kiểm duyệt hồ sơ, quản lý thông tin người dùng, gói khám, chương trình từ thiện, các yêu cầu hỗ trợ và lịch hẹn, báo cáo thống kê. Với mục tiêu xây dựng một môi trường nhân đạo, minh bạch và dễ truy cập, MEDICALHOPE giúp việc kết nối – hỗ trợ – điều trị trở nên thuận tiện, nhanh chóng và đáng tin cậy cho cộng đồng.

LỜI NÓI ĐẦU

Trong suốt thời gian học tập và rèn luyện tại Khoa Công nghệ Thông tin – Trường Đại học Kiến trúc Đà Nẵng, em đã có cơ hội tiếp thu nhiều kiến thức quý báu từ quý thầy cô, những người luôn tận tâm giảng dạy và truyền cảm hứng cho chúng em trên con đường học vấn. Đó chính là nền tảng quan trọng giúp em có được khả năng nghiên cứu, tư duy và hoàn thành đồ án tốt nghiệp này.

Em xin bày tỏ lòng biết ơn sâu sắc đến TS. Lý Quỳnh Trân, giảng viên hướng dẫn, người đã đồng hành, định hướng và hỗ trợ em trong suốt quá trình thực hiện đề tài. Những chỉ dẫn chi tiết, những góp ý chân thành và sự tận tụy của cô là nguồn động lực lớn giúp em hoàn thiện bài làm một cách tốt nhất.

Mặc dù đã nỗ lực rất nhiều nhưng với kinh nghiệm thực tiễn còn hạn chế, bài đồ án chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được sự góp ý của quý thầy cô để em có thể rút kinh nghiệm và hoàn thiện bản thân hơn trong tương lai.

Em xin trân trọng cảm ơn!

LỜI CAM ĐOAN

Em xin cam đoan rằng toàn bộ kết quả và nội dung trình bày trong đồ án tốt nghiệp này là do chính em thực hiện dưới sự hướng dẫn của TS. Lý Quỳnh Trân, không sao chép hay sử dụng trái phép bất kỳ công trình nào của người khác. Những nội dung được trích dẫn hoặc tham khảo đều được ghi rõ nguồn và tuân thủ đầy đủ các quy định về trích dẫn học thuật.

Mọi dữ liệu, phân tích và kết luận trong báo cáo đều phản ánh đúng quá trình nghiên cứu và làm việc của bản thân em. Em xin chịu hoàn toàn trách nhiệm trước nhà trường về tính trung thực và nguyên bản của nội dung đồ án này.

Sinh viên thực hiện

Võ Trọng Quyền

MỤC LỤC

LỜI NÓI ĐẦU	i
LỜI CAM ĐOAN	ii
MỤC LỤC	iii
DANH SÁCH BẢNG BIỂU	v
DANH MỤC HÌNH ẢNH	vi
DANH MỤC CHỮ VIẾT TẮT	vi
MỞ ĐẦU	1
Chương 1: GIỚI THIỆU TỔNG QUAN	2
1.1. Lý do chọn đề tài.....	2
1.2. Lịch sử nghiên cứu	2
1.3. Mục tiêu đề tài.....	3
1.4. Đối tượng, phạm vi nghiên cứu	4
Chương 2: CƠ SỞ LÝ THUYẾT	5
2.1. Giới thiệu về JavaScript.....	5
2.2. Giới thiệu về TypeScript	7
2.3. Giới thiệu về ReactJS	9
2.4. Giới thiệu về hệ quản trị cơ sở dữ liệu MongoDB.....	12
2.5. Giới thiệu về ExpressJS.....	15
2.6. Giới thiệu về NodeJS.....	17
Chương 3: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG.....	21
3.1. Xác định yêu cầu hệ thống.....	21
3.1.1. Yêu cầu chức năng.....	21
3.1.2. Yêu cầu phi chức năng.....	22
3.2. Phân tích yêu cầu hệ thống	23
3.2.1. Chức năng đăng nhập.....	23
3.2.2. Chức năng quản lý sự kiện và hoạt động	23
3.2.3. Chức năng quản lý gói khám.....	23
3.2.4. Chức năng quản lý người dùng	23
3.2.5. Chức năng quản lý hỗ trợ.....	23
3.2.6. Chức năng quản lý lịch hẹn.....	24
3.2.7. Chức năng quản lý tổ chức.....	24
3.2.8. Chức năng quản lý đánh giá	24

3.2.9. Chức năng quản lý bệnh nhân	24
3.2.10. Chức năng quản lý thông tin cá nhân.....	24
3.2.11. Chức năng quản lý bác sĩ	24
3.2.12. Chức năng Chatbot AI	25
3.3. Mô hình hóa yêu cầu	25
3.3.1. Chức năng đăng nhập.....	26
3.3.2. Chức năng quản lý sự kiện và hoạt động	28
3.3.3. Chức năng quản lý gói khám.....	31
3.3.4. Chức năng quản lý người dùng	34
3.3.5. Chức năng quản lý hỗ trợ.....	36
3.3.6. Chức năng quản lý lịch hẹn.....	40
3.3.7. Chức năng quản lý tổ chức.....	43
3.3.8. Chức năng quản lý đánh giá	46
3.3.9. Chức năng quản lý bệnh nhân	49
3.3.10. Chức năng quản lý thông tin cá nhân.....	52
3.3.11. Chức năng quản lý bác sĩ	54
3.3.12. Chức năng Chatbot AI	57
3.4. Thiết kế cơ sở dữ liệu	59
3.5. Danh mục các bảng	59
3.6. Thiết kế giao diện.....	66
3.6.1. Giao diện Admin.....	66
3.6.2. Giao diện Bệnh nhân.....	72
3.6.3. Giao diện Bác sĩ.....	74
3.6.4. Giao diện Người dùng.....	77
Chương 4: CÀI ĐẶT VÀ PHÁT TRIỂN WEBSITE	82
4.1. Cấu trúc thư mục code.....	82
4.1.1. Cấu trúc thư mục code Back End	82
4.1.2. Cấu trúc thư mục code Front End.....	82
KẾT LUẬN	84
I. KẾT QUẢ ĐẠT ĐƯỢC.....	84
II. HẠN CHẾ	84
III. HƯỚNG PHÁT TRIỂN	85
TÀI LIỆU THAM KHẢO	86

DANH SÁCH BẢNG BIỂU

Bảng 3.1 Mô tả Use Case đăng nhập	26
Bảng 3.2 Mô tả Use Case quản lý sự kiện và hoạt động	28
Bảng 3.3 Mô tả Use Case Quản lý gói khám	31
Bảng 3.4 Mô tả Use Case Quản lý người dùng	34
Bảng 3.5 Mô tả Use Case Quản lý hỗ trợ	36
Bảng 3.6 Mô tả Use Case Quản lý lịch hẹn	40
Bảng 3.7 Mô tả Use Case Quản lý tổ chức	43
Bảng 3.8 Mô tả Use Case Quản lý đánh giá	46
Bảng 3.9 Mô tả Use Case Quản lý bệnh nhân	49
Bảng 3.10 Mô tả Use Case Quản lý thông tin cá nhân	52
Bảng 3.11 Mô tả Use Case Quản lý bác sĩ	54
Bảng 3.12 Mô tả Use Case ChatbotAI	57
Bảng 3.13 Mô tả các bảng cơ sở dữ liệu	59
Bảng 3.14 Mô tả bảng User	60
Bảng 3.15 Mô tả bảng Profile	61
Bảng 3.16 Mô tả bảng Patient	61
Bảng 3.17 Mô tả bảng Doctor	62
Bảng 3.18 Mô tả bảng Package Registration	62
Bảng 3.19 Mô tả bảng Donation	63
Bảng 3.20 Mô tả bảng Event	63
Bảng 3.21 Mô tả bảng Event Registration	64
Bảng 3.22 Mô tả bảng Package	64
Bảng 3.23 Mô tả bảng Appointment	64
Bảng 3.24 Mô tả bảng Testimonial	65
Bảng 3.25 Mô tả bảng Partner	65
Bảng 3.26 Mô tả bảng Patient Assistance	65

DANH MỤC HÌNH ẢNH

Hình 3.1 Biểu đồ Use Case tổng quát.....	25
Hình 3.2 Biểu đồ Use Case đăng nhập.....	26
Hình 3.3 Biểu đồ tuần tự đăng nhập.....	27
Hình 3.4 Biểu đồ Use Case quản lý sự kiện và hoạt động.....	28
Hình 3.5 Biểu đồ tuần tự quản lý sự kiện cho admin	29
Hình 3.6 Biểu đồ tuần tự đăng ký sự kiện cho user, xem và duyệt admin ..	30
Hình 3.7 Biểu đồ Use Case quản lý gói khám	31
Hình 3.8 Biểu đồ tuần tự quản lý gói khám.....	32
Hình 3.9 Biểu đồ tuần tự đăng ký gói khám cho user, xem và duyệt cho admin	33
Hình 3.10 Biểu đồ Use Case quản lý người dùng	34
Hình 3.11 Biểu đồ tuần tự quản lý người dùng.....	35
Hình 3.12 Biểu đồ Use Case quản lý hỗ trợ	36
Hình 3.13 Biểu đồ tuần tự quản lý hỗ trợ(gửi yêu cầu hỗ trợ)	38
Hình 3.14 Biểu đồ tuần tự quản lý hỗ trợ(Tạo quyền góp), xem danh sách quyền góp.....	39
Hình 3.15 Biểu đồ Use Case quản lý lịch hẹn	40
Hình 3.16 Biểu đồ quản lý lịch hẹn cho bệnh nhân.....	41
Hình 3.17 Biểu đồ tuần tự quản lý lịch hẹn cho bác sĩ.....	42
Hình 3.18 Biểu đồ Use Case quản lý tổ chức	43
Hình 3.19 Biểu đồ tuần tự quản lý đối tác(xem danh sách) cho người dùng	44
Hình 3.20 Biểu đồ tuần tự quản lý đối tác cho admin	45
Hình 3.21 Biểu đồ Use Case quản lý đánh giá.....	46
Hình 3.22 Biểu đồ tuần tự quản lý đánh giá cho admin.....	47
Hình 3.23 Biểu đồ tuần tự quản lý đánh giá cho người dùng	48
Hình 3.24 Biểu đồ Use Case quản lý bệnh nhân	49
Hình 3.25 Biểu đồ tuần tự quản lý bệnh nhân cho admin	50
Hình 3.26 Biểu đồ tuần tự quản lý bệnh nhân cho bác sĩ.....	51
Hình 3.27 Biểu đồ Use Case quản lý thông tin cá nhân.....	52
Hình 3.28 Biểu đồ tuần tự quản lý hồ sơ cá nhân	53
Hình 3.29 Biểu đồ Use Case quản lý bác sĩ.....	54
Hình 3.30 Biểu đồ tuần tự quản lý bác sĩ cho bệnh nhân	55

Hình 3.31 Biểu đồ tuần tự quản lý bác sĩ cho admin	56
Hình 3.32 Biểu đồ Use Case ChatbotAI.....	57
Hình 3.33 Biểu đồ tuần tự ChatbotAI.....	58
Hình 3.34 Mô hình quan hệ	59
Hình 3.35 Giao diện đăng nhập	66
Hình 3.36 Giao diện chính Admin	66
Hình 3.37 Giao diện quản lý người dùng	67
Hình 3.38 Giao diện quản lý bệnh nhân.....	67
Hình 3.39 Giao diện thêm bệnh nhân	67
Hình 3.40 Giao diện quản lý bác sĩ	68
Hình 3.41 Giao diện quản lý hỗ trợ.....	68
Hình 3.42 Giao diện quản lý đối tác.....	68
Hình 3.43 Giao diện thêm đối tác	69
Hình 3.44 Giao diện quản lý gói khám	69
Hình 3.45 Giao diện tạo gói khám	69
Hình 3.46 Giao diện đơn đăng ký gói khám	70
Hình 3.47 Giao diện quản lý sự kiện.....	70
Hình 3.48 Giao diện thêm sự kiện.....	70
Hình 3.49 Giao diện danh sách đơn đăng ký tham gia sự kiện	71
Hình 3.50 Giao diện quản lý lời yêu thương	71
Hình 3.51 Giao diện Thống kê báo cáo.....	71
Hình 3.52 Giao diện chính của bệnh nhân	72
Hình 3.53 Giao diện quản lý thông tin cá nhân	72
Hình 3.54 Giao diện cập nhật thông tin cá nhân	72
Hình 3.55 Giao diện danh sách lịch hẹn	73
Hình 3.56 Giao diện đặt lịch khám	73
Hình 3.57 Giao diện danh sách bác sĩ	73
Hình 3.58 Giao diện danh sách yêu cầu hỗ trợ và ủng hộ	74
Hình 3.59 Giao diện tạo yêu cầu hỗ trợ	74
Hình 3.60 Giao diện chính cho bác sĩ.....	74
Hình 3.61 Giao diện hồ sơ bác sĩ.....	75
Hình 3.62 Giao diện cập nhật hồ sơ	75
Hình 3.63 Giao diện quản lý lịch khám	75
Hình 3.64 Giao diện cập nhật lịch rảnh bác sĩ.....	76
Hình 3.65 Giao diện danh sách bệnh nhân	76

Hình 3.66 Giao diện thông tin chi tiết bệnh nhân	76
Hình 3.67 Giao diện đăng ký.....	77
Hình 3.68 Giao diện chính	77
Hình 3.69 Giao diện các hoạt động thiện nguyện đang diễn ra	77
Hình 3.70 Giao diện chi tiết hoạt động.....	78
Hình 3.71 Giao diện đăng ký tham gia hoạt động.....	78
Hình 3.72 Giao diện các gói khám miễn phí	78
Hình 3.73 Giao diện đăng ký gói khám miễn phí.....	79
Hình 3.74 Giao diện các yêu cầu hỗ trợ cần ủng hộ.....	79
Hình 3.75 Giao diện thanh toán ủng hộ	79
Hình 3.76 Giao diện danh sách chia sẻ lời yêu thương.....	80
Hình 3.77 Giao diện tạo lời yêu thương.....	80
Hình 3.78 Giao diện thông tin các chuyến xe 0 đồng.....	80
Hình 3.79 Giao diện thông tin các bữa cơm 0 đồng	81
Hình 3.80 Giao diện các tổ chức.....	81
Hình 3.81 Giao diện ChatbotAI	81

DANH MỤC CHỮ VIẾT TẮT

STT	Chữ viết tắt	Tên đầy đủ
1	AI	Artificial Intelligence
2	API	Application Programming Interface
3	CRUD	Create – Read – Update – Delete
4	DB	Database
5	DOM	Document Object Model
6	EHR	Electronic Health Record
7	HTTP	HyperText Transfer Protocol
8	HTTPS	HyperText Transfer Protocol Secure
9	JWT	Json Web Token
10	MERN	MongoDB – Express.js – React.js – Node.js
11	MongoDB	Mongo Database
12	NoSQL	Not only SQL
13	PWA	Progressive Web App
14	REST/RESTful	Representational State Transfer
15	SPA	Single Page Application
16	UI	User Interface
17	UX	User Experience
18	UML	Unified Modeling Language
19	2FA	Two-Factor Authentication

MỞ ĐẦU

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ tại Việt Nam, lĩnh vực y tế từ thiện và hỗ trợ bệnh nhân có hoàn cảnh khó khăn đang rất cần những nền tảng công nghệ hiện đại để nâng cao hiệu quả kết nối, quản lý và minh bạch hóa các hoạt động hỗ trợ. Hiện nay, nhiều bệnh nhân nghèo, đặc biệt ở vùng sâu vùng xa, vẫn gặp rất nhiều trở ngại khi tiếp cận dịch vụ khám chữa bệnh miễn phí hoặc các chương trình thiện nguyện do thiếu thông tin, quy trình phức tạp và sự phối hợp chưa đồng bộ giữa bệnh nhân – bác sĩ tình nguyện – nhà hảo tâm – tổ chức từ thiện.

Xuất phát từ thực tế đó, MedicalHope ra đời với sứ mệnh trở thành cầu nối nhân ái, giúp bệnh nhân khó khăn dễ dàng đăng ký hồ sơ, được duyệt và nhận hỗ trợ y tế kịp thời; đồng thời tạo điều kiện để các bác sĩ tình nguyện, nhà hảo tâm và tổ chức từ thiện tham gia một cách minh bạch, nhanh chóng và hiệu quả.

Đồ án tốt nghiệp này hướng đến việc nghiên cứu và xây dựng một website quản lý bệnh nhân toàn diện dành riêng cho trung tâm y tế MedicalHope, sử dụng các công nghệ hiện đại: ReactJS + TypeScript + TailwindCSS ở phía client, Node.js + ExpressJS + MongoDB ở phía server, tích hợp Chatbot AI thông minh (Google Gemini) và cổng thanh toán quốc tế PayPal. Hệ thống cung cấp đầy đủ các chức năng chính:

- Quản lý đa vai trò (Admin – Bác sĩ – Bệnh nhân – Nhà hảo tâm)
- Đặt lịch khám trực tuyến với bác sĩ tình nguyện
- Quản lý hồ sơ bệnh nhân, gói khám miễn phí, sự kiện thiện nguyện
- Tiếp nhận và xử lý yêu cầu hỗ trợ kèm quyền góp minh bạch
- Chatbot AI hỗ trợ 24/7 bằng tiếng Việt
- Giao diện thân thiện, responsive, dễ sử dụng với mọi đối tượng

Đồ án không chỉ là sản phẩm học thuật mà còn mang ý nghĩa nhân văn sâu sắc, góp phần nhỏ bé vào việc ứng dụng công nghệ thông tin để lan tỏa yêu thương, hỗ trợ những hoàn cảnh kém may mắn trong xã hội.

Em hy vọng sản phẩm sẽ là một giải pháp thực tiễn, có thể triển khai thực tế tại MedicalHope và các tổ chức từ thiện khác trong tương lai.

Chương 1: GIỚI THIỆU TỔNG QUAN

1.1. Lý do chọn đề tài

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, việc ứng dụng các giải pháp số vào lĩnh vực y tế đang trở thành xu hướng nhằm nâng cao hiệu quả quản lý, cải thiện chất lượng dịch vụ khám chữa bệnh và tạo thuận lợi cho người dân trong quá trình tiếp cận chăm sóc sức khỏe. Đặc biệt với các mô hình khám chữa bệnh thiện nguyện hay hỗ trợ bệnh nhân có hoàn cảnh khó khăn như MedicalHope, nhu cầu về một hệ thống quản lý chuyên nghiệp, minh bạch và vận hành hiệu quả là vô cùng quan trọng, không chỉ lưu trữ – quản lý hồ sơ bệnh nhân mà còn kết nối đồng bộ bác sĩ, nhà hảo tâm, tổ chức từ thiện và người bệnh.

Công nghệ web hiện đại với ReactJS ở phía giao diện và NodeJS ở phía máy chủ được đánh giá cao nhờ khả năng xử lý nhanh, tính linh hoạt và dễ mở rộng. Việc lựa chọn hai công nghệ này giúp mang lại trải nghiệm thân thiện cho người dùng, đảm bảo hiệu năng và khả năng nâng cấp trong tương lai, đồng thời giúp sinh viên củng cố kiến thức chuyên môn và rèn luyện kỹ năng thực tiễn.

Bên cạnh đó, sự phát triển mạnh mẽ của trí tuệ nhân tạo mở ra cơ hội tích hợp chatbot AI vào hệ thống, hỗ trợ bệnh nhân tra cứu thông tin, giải đáp thắc mắc cơ bản về sức khỏe và giảm tải một phần công việc cho đội ngũ y tế. Điều này góp phần nâng cao chất lượng dịch vụ và mang lại trải nghiệm tiện lợi hơn cho người dùng.

Từ những nhu cầu thực tiễn và giá trị ý nghĩa đó, em quyết định chọn đề tài "Nghiên cứu ReactJS, NodeJS xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI". Đề tài không chỉ giúp em vận dụng các kiến thức đã học vào một dự án cụ thể, mà còn mang tính nhân văn cao, hỗ trợ cộng đồng và đóng góp vào sự phát triển của các nền tảng chăm sóc sức khỏe thiện nguyện.

Với mong muốn tạo ra một sản phẩm có ý nghĩa, hữu ích và đáp ứng nhu cầu thực tế, em lựa chọn đề tài này như một bước khởi đầu cho con đường phát triển nghề nghiệp trong tương lai.

1.2. Lịch sử nghiên cứu

Trong nước:

- Nhiều hệ thống Electronic Health Record (EHR) đã được triển khai, giúp số hóa hồ sơ bệnh nhân, quản lý lịch khám và theo dõi quá trình điều trị.
- Ứng dụng ReactJS và NodeJS trong phát triển web y tế cho phép xây dựng giao diện linh hoạt, phản hồi nhanh và backend mạnh mẽ.

- Chatbot AI được tích hợp vào hệ thống y tế để tư vấn sức khỏe cơ bản, trả lời thắc mắc, hướng dẫn bệnh nhân và giảm tải công việc cho nhân viên y tế.

Ngoài nước:

- Các hệ thống quản lý bệnh nhân và nền tảng y tế điện tử chủ yếu triển khai tại các bệnh viện lớn và trung tâm y tế.
- Chưa có nhiều nền tảng hỗ trợ bệnh nhân có hoàn cảnh khó khăn kết hợp yếu tố thiện nguyện.
- Việc xây dựng website quản lý bệnh nhân tích hợp chatbot AI cho các trung tâm như MedicalHope vẫn là hướng nghiên cứu mới, đáp ứng nhu cầu thực tiễn và ứng dụng công nghệ hiện đại.

1.3. Mục tiêu đề tài

Mục tiêu của đề tài là phát triển một website quản lý bệnh nhân và hỗ trợ y tế thiện nguyện cho trung tâm y tế MedicalHope sử dụng ReactJS và NodeJS. Hệ thống sẽ cung cấp các chức năng quan trọng như Quản lý thông tin bệnh nhân, Quản lý thông tin bác sĩ tình nguyện, Đặt lịch khám, Quản lý các hoạt động từ thiện, Giới thiệu gói khám miễn phí, Các chuyến xe 0 đồng, Điểm phát cơm từ thiện, Danh sách các tổ chức từ thiện trong và ngoài nước, cùng các yêu cầu cần được hỗ trợ. Hệ thống được thiết kế với tính năng bảo mật cao, giao diện thân thiện với người dùng và khả năng mở rộng trong tương lai.

Mục tiêu cụ thể:

- Xây dựng giao diện và trải nghiệm người dùng: Thiết kế giao diện trực quan, thân thiện và dễ sử dụng, đảm bảo tính tương tác cao cho bệnh nhân, bác sĩ, nhà hảo tâm và các tổ chức từ thiện.
- Phát triển tính năng Chatbot AI hỗ trợ giải đáp: Tích hợp chatbot AI giúp tư vấn sức khỏe cơ bản, hướng dẫn sử dụng hệ thống, giải đáp thắc mắc nhanh chóng và nâng cao trải nghiệm người dùng.
- Quản lý thông tin bác sĩ và bệnh nhân: Xây dựng chức năng quản lý hồ sơ bệnh nhân, thông tin bác sĩ tình nguyện, lịch khám và theo dõi quá trình điều trị, đảm bảo tính minh bạch và lịch trình khám hợp lý.
- Phát triển chức năng đặt lịch khám: Bệnh nhân có thể đặt lịch trực tuyến dễ dàng, chọn thời gian và bác sĩ mong muốn. Hệ thống sẽ tự động gửi thông báo xác nhận để bệnh nhân chuẩn bị cho buổi khám.
- Quản lý thông tin các hoạt động thiện nguyện: Hệ thống lưu trữ và quản lý các gói khám miễn phí, các chuyến xe 0 đồng, điểm phát cơm từ thiện, danh

sách tổ chức từ thiện trong và ngoài nước, cũng như các yêu cầu hỗ trợ từ bệnh nhân hoặc cộng đồng.

- Cải tiến liên tục: Theo dõi hiệu suất của hệ thống để cải tiến liên tục, đảm bảo đáp ứng nhu cầu người dùng và nâng cao hiệu quả quản lý.

1.4. Đối tượng, phạm vi nghiên cứu

1.4.1. Đối tượng nghiên cứu

- Nội dung: ReactJS, NodeJS, ExpressJS, MongoDB, Chatbot AI.
- Đối tượng sử dụng: Bệnh nhân, bác sĩ tình nguyện, nhà hảo tâm và các tổ chức từ thiện tham gia vào các hoạt động y tế thiện nguyện do trung tâm MedicalHope triển khai.

1.4.2. Phạm vi nghiên cứu

- Phạm vi nội dung: Xây dựng website quản lý bệnh nhân và hỗ trợ y tế thiện nguyện cho trung tâm MedicalHope, bao gồm các chức năng đặt lịch khám miễn phí, quản lý hồ sơ bệnh nhân, giới thiệu gói khám miễn phí, các hoạt động từ thiện, chuyển xe 0 đồng, điểm phát cơm từ thiện, danh sách tổ chức từ thiện trong và ngoài nước, các yêu cầu cần hỗ trợ và tích hợp chatbot AI.

Hệ thống hướng đến phục vụ các đối tượng tham gia vào các dịch vụ khám chữa bệnh thiện nguyện, các hoạt động từ thiện, và hỗ trợ bệnh nhân có hoàn cảnh khó khăn trong phạm vi lãnh thổ Việt Nam.

Chương 2: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về JavaScript

JavaScript là một ngôn ngữ lập trình hướng kịch bản (scripting language) được phát triển để tạo sự tương tác và sinh động cho các trang web. Nó cho phép lập trình viên sử dụng các đối tượng có sẵn hoặc tự định nghĩa để xử lý dữ liệu, quản lý giao diện và tương tác người dùng một cách linh hoạt. Nhờ khả năng ứng dụng rộng rãi, tốc độ xử lý nhanh và dễ dàng kết hợp với HTML, CSS, JavaScript đã trở thành một trong những công nghệ cốt lõi trong phát triển website hiện đại.

Một trong những điểm mạnh nổi bật của JavaScript là khả năng hoạt động trên nhiều trình duyệt khác nhau, bao gồm Chrome, Firefox, Safari, Edge, cũng như các trình duyệt trên thiết bị di động. Điều này giúp các ứng dụng web chạy ổn định, mượt mà và đồng nhất trải nghiệm người dùng trên nhiều nền tảng khác nhau. JavaScript có thể được nhúng trực tiếp vào các thẻ HTML hoặc lưu trong các file riêng biệt để sử dụng khi cần thiết, mang lại sự tiện lợi và khả năng mở rộng trong phát triển dự án.

JavaScript thực hiện phần lớn các thao tác trực tiếp trên phía client (trình duyệt của người dùng), điều này giúp giảm tải cho máy chủ, tiết kiệm băng thông và tăng tốc độ phản hồi của website. Khi người dùng tương tác với trang web, các thao tác như xử lý sự kiện, hiển thị thông báo, cập nhật nội dung hay thao tác với các phần tử giao diện được thực hiện ngay lập tức, mang lại trải nghiệm mượt mà và trực quan.

Một số ưu điểm chính của JavaScript:

- Xử lý trực tiếp phía client: Mã JavaScript được thực hiện trên trình duyệt của người dùng, giảm áp lực xử lý cho server, đồng thời nâng cao tốc độ phản hồi cho các thao tác tương tác.
- Dễ học, dễ sử dụng: Cú pháp của JavaScript đơn giản, gần gũi với tiếng Anh và có khả năng thao tác trực tiếp với DOM (Document Object Model), giúp lập trình viên nhanh chóng tạo ra các tính năng cần thiết mà không gặp khó khăn.
- Khả năng xử lý nhanh chóng: Các thao tác được thực thi ngay lập tức trên trình duyệt, giúp người dùng nhận kết quả gần như ngay lập tức, đặc biệt hữu ích cho các ứng dụng yêu cầu phản hồi thời gian thực.

- Đa nền tảng và đa thiết bị: JavaScript tương thích với hầu hết các trình duyệt máy tính và thiết bị di động, đảm bảo các website có thể hoạt động ổn định trên nhiều môi trường khác nhau.
- Khả năng tích hợp cao: JavaScript hoạt động liền mạch với HTML và CSS, cho phép tạo giao diện người dùng trực quan, sinh động và dễ tùy chỉnh.

Ví dụ minh họa cơ bản về JavaScript:

```
// Hàm hiển thị thông báo chào mừng người dùng
function greetUser(name) {
    alert(`Xin chào, ${name}! Chào mừng bạn.`);
}

// Gọi hàm
greetUser('Quyen');
```

Lợi ích của JavaScript trong phát triển web:

- Tạo tính tương tác cao (Interactivity): Giúp các trang web phản hồi nhanh các thao tác của người dùng, xử lý sự kiện như click, hover, nhập dữ liệu, và cập nhật nội dung mà không cần tải lại toàn bộ trang.
- Khả năng đa nền tảng: Chạy được trên mọi trình duyệt phổ biến, từ desktop đến mobile, mà không cần phần mềm bổ sung.
- Hỗ trợ thời gian thực (Real-time): Phù hợp cho các ứng dụng cần cập nhật trực tiếp, như chat, thông báo, hay các dịch vụ live-streaming.
- Hỗ trợ thư viện và framework mạnh mẽ: Các công cụ phổ biến như ReactJS, Angular, VueJS, jQuery giúp lập trình viên phát triển các ứng dụng web nhanh chóng, dễ bảo trì và mở rộng.
- Chạy cả phía client và server: Với Node.js, JavaScript không chỉ được sử dụng phía client mà còn có thể phát triển các ứng dụng phía server mạnh mẽ, hỗ trợ RESTful API, xử lý cơ sở dữ liệu và các logic nghiệp vụ phức tạp.
- Tối ưu hóa trải nghiệm người dùng: Cho phép tạo các giao diện mượt mà, trực quan, dễ thao tác và phù hợp với nhu cầu của nhiều đối tượng khác nhau.

Nhờ những ưu điểm này, JavaScript không chỉ là một công cụ lập trình thông thường mà còn là nền tảng quan trọng cho các ứng dụng web hiện đại, từ những website đơn giản đến các hệ thống phức tạp, yêu cầu tính tương tác và xử lý dữ liệu thời gian

thực. Việc nắm vững và áp dụng JavaScript là một lợi thế lớn cho các lập trình viên trong việc phát triển các sản phẩm công nghệ web hiện nay.

2.2. Giới thiệu về TypeScript

TypeScript là một ngôn ngữ lập trình được phát triển bởi Microsoft, được xem là siêu tập (superset) của JavaScript, nghĩa là mọi mã JavaScript hợp lệ cũng là mã TypeScript hợp lệ. Tuy nhiên, TypeScript bổ sung nhiều tính năng quan trọng mà JavaScript thuần túy không có, đặc biệt là hệ thống kiểu tĩnh (static typing system), giúp phát hiện lỗi khi biên dịch thay vì khi chạy chương trình. Điều này mang lại sự ổn định cao hơn, đặc biệt hữu ích trong các dự án lớn, phức tạp hoặc có nhiều lập trình viên cùng tham gia.

Một trong những đặc điểm nổi bật của TypeScript là hỗ trợ kiểu dữ liệu tĩnh. Khi lập trình viên khai báo kiểu dữ liệu cho biến, hàm, đối tượng hay mảng, TypeScript sẽ kiểm tra tính hợp lệ của các thao tác trên các kiểu dữ liệu này ngay từ lúc biên dịch. Điều này giúp giảm thiểu lỗi runtime (lỗi khi chương trình chạy) và nâng cao độ tin cậy của ứng dụng. Ví dụ, nếu bạn khai báo một biến là kiểu số nhưng lại gán giá trị chuỗi, TypeScript sẽ báo lỗi ngay, giúp lập trình viên điều chỉnh trước khi đưa vào chạy thực tế.

TypeScript cũng hỗ trợ đầy đủ lập trình hướng đối tượng (OOP), cung cấp các tính năng như class, interface, abstract class, inheritance, encapsulation, và polymorphism, giúp xây dựng các hệ thống phần mềm quy mô lớn một cách rõ ràng, dễ quản lý và mở rộng. Khả năng này đặc biệt hữu ích trong việc phát triển các ứng dụng phức tạp như hệ thống quản lý bệnh nhân, website y tế hoặc các nền tảng thương mại điện tử, nơi có nhiều module và chức năng cần phối hợp chặt chẽ.

Một điểm mạnh nữa của TypeScript là tích hợp hoàn toàn với JavaScript. Lập trình viên có thể sử dụng tất cả các thư viện, framework và công cụ hiện có của JavaScript mà không gặp trở ngại. Đồng thời, mã TypeScript sẽ được biên dịch (transpile) sang JavaScript chuẩn, đảm bảo có thể chạy trên mọi trình duyệt và môi trường hỗ trợ JavaScript như Node.js, ReactJS hay Angular. Đây là một lợi thế lớn khi muốn nâng cấp dự án từ JavaScript sang TypeScript mà không cần viết lại toàn bộ mã nguồn.

Ưu điểm nổi bật của TypeScript:

- Hệ thống kiểu tĩnh: Giúp phát hiện lỗi khi biên dịch, tăng tính ổn định và đáng tin cậy cho ứng dụng.

- Hỗ trợ OOP đầy đủ: Dễ dàng xây dựng các cấu trúc phức tạp, quản lý module, mở rộng và bảo trì dự án lâu dài.
- Tích hợp với JavaScript: Có thể dùng các thư viện, framework JS sẵn có mà không gặp xung đột.
- Tương thích đa nền tảng: Sau khi biên dịch, TypeScript chạy được trên mọi trình duyệt, môi trường máy tính và thiết bị di động.
- Hỗ trợ IDE mạnh mẽ: Các công cụ như Visual Studio Code cung cấp tính năng tự động hoàn thiện mã, kiểm tra lỗi thời gian thực, giúp lập trình viên phát triển nhanh và ít lỗi hơn.

Ứng dụng thực tiễn của TypeScript:

TypeScript được sử dụng rộng rãi trong các dự án web và ứng dụng hiện đại, đặc biệt trong những dự án quy mô lớn và cần bảo trì lâu dài. Các framework phổ biến như Angular được phát triển hoàn toàn bằng TypeScript, trong khi ReactJS và Node.js cũng thường kết hợp TypeScript để tăng tính ổn định, bảo trì dễ dàng và đảm bảo trải nghiệm người dùng mượt mà. Trong các hệ thống quản lý bệnh nhân, website y tế, nền tảng thương mại điện tử hay các ứng dụng doanh nghiệp, TypeScript giúp xây dựng các module rõ ràng, quản lý dữ liệu và nghiệp vụ phức tạp, đồng thời giảm thiểu lỗi runtime và nâng cao hiệu suất phát triển.

Ví dụ cơ bản về TypeScript:

```
// Khai báo biến với kiểu dữ liệu
let userName: string = "Tiến";

// Hàm tính tổng với kiểu dữ liệu rõ ràng
function sum(a: number, b: number): number {
    return a + b;
}

// Sử dụng interface để định nghĩa cấu trúc đối tượng
interface Patient {
    name: string;
    age: number;
    hasInsurance: boolean;
}
```

```
const patient1: Patient = {  
  name: "Nguyễn Văn A",  
  age: 30,  
  hasInsurance: true  
};  
  
console.log(`Tên bệnh nhân: ${patient1.name}, Tuổi: ${patient1.age}`);
```

Lợi ích khi sử dụng TypeScript trong dự án:

- Phát hiện lỗi sớm: Các lỗi về kiểu dữ liệu hoặc sử dụng không hợp lệ được phát hiện ngay từ lúc biên dịch, giảm thiểu lỗi runtime.
- Dễ bảo trì và mở rộng: Các dự án lớn, nhiều lập trình viên cùng tham gia sẽ dễ quản lý, thêm mới module hoặc tính năng mà không ảnh hưởng đến phần khác của dự án.
- Tăng năng suất lập trình: Hỗ trợ gợi ý cú pháp, kiểm tra lỗi, tự động hoàn thiện mã, giúp lập trình viên viết mã nhanh và chính xác hơn.
- Ứng dụng trong các framework hiện đại: TypeScript kết hợp với React, Angular, Vue giúp phát triển các ứng dụng SPA (Single Page Application) nhanh chóng, trực quan và hiệu quả.
- Đảm bảo chất lượng dự án dài hạn: Giúp các dự án web phức tạp hoặc hệ thống y tế trực tuyến duy trì hiệu năng ổn định, ít lỗi và dễ bảo trì.

Nhờ những ưu điểm vượt trội này, TypeScript ngày càng trở thành ngôn ngữ lựa chọn hàng đầu cho các dự án web hiện đại, đặc biệt là những hệ thống phức tạp, yêu cầu quản lý dữ liệu lớn và bảo trì lâu dài. Việc kết hợp TypeScript với ReactJS và NodeJS trong phát triển các website quản lý bệnh nhân, nền tảng y tế trực tuyến hay các ứng dụng từ thiện như MedicalHope sẽ mang lại tính ổn định, khả năng mở rộng và trải nghiệm người dùng tối ưu, đồng thời giúp sinh viên và lập trình viên nâng cao kỹ năng chuyên môn và khả năng phát triển ứng dụng thực tiễn.

2.3. Giới thiệu về ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, chuyên dùng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. Điểm mạnh nổi bật của ReactJS là khả năng tạo ra các giao diện web tương tác cao, linh hoạt và hiệu suất vượt trội, nhờ vào kiến trúc component-based (dựa trên thành phần). Kiến trúc này

cho phép các nhà phát triển tách ứng dụng thành nhiều thành phần nhỏ, độc lập, dễ dàng quản lý, bảo trì và tái sử dụng trong nhiều phần của dự án.

Một trong những ưu điểm nổi bật của ReactJS là khả năng tối ưu hóa hiệu suất thông qua Virtual DOM. Thay vì cập nhật toàn bộ trang khi có thay đổi dữ liệu, React chỉ render lại những phần cần thiết, giúp ứng dụng chạy nhanh, mượt mà và giảm tải tài nguyên cho trình duyệt. Điều này đặc biệt quan trọng trong các ứng dụng web phức tạp hoặc các nền tảng thương mại điện tử, nơi người dùng thường xuyên tương tác với trang, như tìm kiếm sản phẩm, thêm sản phẩm vào giỏ hàng hay thanh toán trực tuyến.

ReactJS còn cung cấp khả năng tích hợp dễ dàng với các công nghệ khác, như Redux để quản lý trạng thái, React Router cho định tuyến SPA (Single Page Application), hoặc Next.js để hỗ trợ server-side rendering (SSR), từ đó cải thiện SEO và khả năng hiển thị nội dung trên công cụ tìm kiếm. Với việc hỗ trợ ES6+, React giúp viết mã ngắn gọn, rõ ràng và dễ bảo trì hơn, đồng thời tối ưu hóa trải nghiệm người dùng thông qua các cơ chế cập nhật giao diện gần như ngay lập tức khi dữ liệu thay đổi.

Lý do ReactJS được sử dụng phổ biến trong các ứng dụng web và thương mại điện tử:

- **Hiệu suất cao và tối ưu trải nghiệm người dùng:** ReactJS sử dụng Virtual DOM giúp giảm thiểu việc thao tác trực tiếp trên DOM thực, từ đó tăng tốc độ hiển thị và cập nhật giao diện khi có thay đổi dữ liệu. Điều này giúp các ứng dụng chạy mượt mà, giảm tải tài nguyên cho trình duyệt và tối ưu trải nghiệm người dùng. Bên cạnh đó, React còn hỗ trợ tính năng Lazy Loading, cho phép tải nội dung chỉ khi cần thiết, giúp cải thiện tốc độ tải trang, giảm thời gian chờ và mang lại trải nghiệm tương tác liền mạch cho người dùng.
- **Tính linh hoạt và khả năng tái sử dụng:** React được xây dựng theo kiến trúc component-based, cho phép các phần của ứng dụng như danh sách sản phẩm, thanh điều hướng hay biểu mẫu nhập liệu được tách riêng thành các thành phần độc lập, dễ dàng tái sử dụng nhiều lần trong toàn bộ dự án. Cấu trúc mô-đun này cũng hỗ trợ phát triển đồng thời bởi nhiều lập trình viên trên các phần khác nhau của ứng dụng, từ đó tăng hiệu quả, rút ngắn thời gian triển khai và giúp dự án dễ dàng mở rộng, bảo trì lâu dài.
- **Hỗ trợ SEO tốt:** Mặc dù React là một thư viện JavaScript chạy phía client, nhưng khi kết hợp với các công cụ như SSR (Server-side Rendering) hoặc prerendering, ví dụ như Next.js, React có thể cải thiện khả năng lập chỉ mục và xếp hạng của trang web trên các công cụ tìm kiếm. Điều này đặc biệt quan

trọng với các website thương mại điện tử hoặc dịch vụ trực tuyến, nơi khả năng hiển thị và tối ưu hóa SEO có ảnh hưởng trực tiếp đến lưu lượng truy cập và doanh thu.

- Cá nhân hóa trải nghiệm người dùng: React hỗ trợ việc tạo ra các tính năng cá nhân hóa trải nghiệm người dùng dựa trên hành vi và sở thích, chẳng hạn như đề xuất sản phẩm, danh sách yêu thích hoặc các bộ lọc tìm kiếm thông minh. Việc cá nhân hóa này giúp nâng cao sự tương tác, tăng tỷ lệ chuyển đổi và cải thiện trải nghiệm sử dụng dịch vụ hoặc mua sắm trực tuyến, từ đó mang lại giá trị cao hơn cho cả người dùng và doanh nghiệp.

Ví dụ cơ bản về ReactJS:

```
import React from 'react';
import ReactDOM from 'react-dom';

// Tạo một component React
function Welcome(props) {
  return <h1>Xin chào, {props.name}! Chào mừng bạn đến với
  ReactJS.</h1>;
}

// Hiển thị component lên giao diện
ReactDOM.render(
  <Welcome name="Tiến" />,
  document.getElementById('root')
);
```

Lợi ích khi sử dụng ReactJS:

- Tái sử dụng thành phần (Reusable Components): Giúp tiết kiệm thời gian phát triển và giảm lỗi khi bảo trì.
- Hiệu suất cao (High Performance): Virtual DOM chỉ cập nhật những phần giao diện thay đổi, giúp ứng dụng mượt mà.
- Quản lý dữ liệu đơn hướng (Unidirectional Data Flow): Dễ dàng kiểm soát trạng thái và luồng dữ liệu trong ứng dụng.
- Dễ bảo trì: Cấu trúc thành phần rõ ràng, đặc biệt hữu ích với các dự án lớn.

- SEO thân thiện: Có thể kết hợp với SSR để tăng khả năng hiển thị trên các công cụ tìm kiếm.
- Cộng đồng mạnh mẽ: Hỗ trợ bởi một cộng đồng lớn, nhiều thư viện và công cụ hỗ trợ như Redux, React Router, Next.js.

Khi nào nên sử dụng ReactJS:

- Khi cần phát triển giao diện người dùng phức tạp và tương tác cao.
- Khi muốn xây dựng ứng dụng SPA mượt mà với khả năng tái sử dụng mã nguồn.
- Khi muốn phát triển các ứng dụng web có khả năng mở rộng, bảo trì lâu dài và tối ưu hóa hiệu suất.

Với những ưu điểm vượt trội này, ReactJS đang trở thành lựa chọn hàng đầu cho việc phát triển giao diện web hiện đại, đặc biệt trong các dự án quản lý bệnh nhân, nền tảng y tế trực tuyến, thương mại điện tử và các ứng dụng cần trải nghiệm người dùng tối ưu.

2.4. Giới thiệu về hệ quản trị cơ sở dữ liệu MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc loại NoSQL, được thiết kế để quản lý dữ liệu phi cấu trúc hoặc bán cấu trúc với khả năng mở rộng linh hoạt và hiệu suất cao. Khác với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống (RDBMS), MongoDB lưu trữ dữ liệu dưới dạng document thay vì bảng, với mỗi document được biểu diễn theo chuẩn BSON (Binary JSON). Điều này giúp dữ liệu trở nên linh hoạt, dễ thay đổi và phù hợp với các mô hình dữ liệu phức tạp.

Kiến trúc dữ liệu phi quan hệ

- MongoDB sử dụng mô hình document-oriented, nghĩa là dữ liệu được lưu trữ trong các collection thay vì bảng truyền thống. Mỗi document là một đối tượng dữ liệu độc lập, có thể chứa nhiều trường khác nhau với cấu trúc đa dạng. Mô hình này cho phép các nhà phát triển thêm, xóa hoặc thay đổi dữ liệu mà không ảnh hưởng đến các document khác, giảm thiểu công việc quản lý cơ sở dữ liệu khi ứng dụng phát triển hoặc mở rộng.
- Document trong MongoDB có thể lưu trữ thông tin phức tạp, bao gồm các mảng, object lồng nhau, hoặc các kiểu dữ liệu đa dạng khác nhau, giúp xử lý dữ liệu linh hoạt mà không cần định nghĩa trước cấu trúc cứng nhắc.

Hiệu suất và khả năng mở rộng

- MongoDB nổi bật với khả năng mở rộng linh hoạt theo cả chiều ngang (horizontal scaling) và chiều dọc (vertical scaling). Hệ thống hỗ trợ sharding, phân tán dữ liệu trên nhiều máy chủ để xử lý lượng dữ liệu lớn mà vẫn đảm bảo tốc độ truy xuất nhanh. Đồng thời, replication thông qua replica set giúp sao lưu dữ liệu và đảm bảo tính sẵn sàng cao, bảo vệ dữ liệu khi có sự cố phần cứng hoặc mạng.
- MongoDB được tối ưu hóa cho truy vấn nhanh và xử lý dữ liệu lớn, cho phép thực hiện các truy vấn phức tạp mà không làm giảm hiệu suất. Với các tính năng như indexing, MongoDB tối ưu hóa tốc độ tìm kiếm, lọc và sắp xếp dữ liệu.

Truy vấn và thao tác dữ liệu

- MongoDB sử dụng MongoDB Query Language (MQL) với cú pháp gần giống JSON, cho phép tìm kiếm, lọc, sắp xếp, cập nhật, xóa và tổng hợp dữ liệu dễ dàng. Ngoài ra, aggregation pipeline cung cấp khả năng xử lý dữ liệu theo từng bước logic, thực hiện các phép tính và thống kê trực tiếp trên server, giảm tải cho ứng dụng phía client.

Ví dụ:

```
// Thêm một đối tượng vào collection 'KhachHang'  
db.KhachHang.insertOne({  
  ten: "Nguyen Van A",  
  tuoi: 30,  
  diaChi: "Ha Noi",  
  lichSuMuaSam: [  
    { ngay: "2025-11-22", sanPham: "Laptop" }  
  ]  
});
```

Bảo mật và toàn vẹn dữ liệu

MongoDB cung cấp nhiều lớp bảo mật:

- Quản lý quyền truy cập theo người dùng và nhóm.

- Xác thực bằng mật khẩu, hỗ trợ các cơ chế phức tạp như LDAP, Kerberos.
- Mã hóa dữ liệu khi lưu trữ và truyền tải.

MongoDB cũng hỗ trợ transaction theo chuẩn ACID trên các document, đảm bảo tính toàn vẹn dữ liệu trong các thao tác đồng thời, tránh mất mát thông tin trong môi trường nhiều người dùng.

Ứng dụng và tích hợp

MongoDB thường được sử dụng trong các ứng dụng web hiện đại kết hợp với NodeJS, ExpressJS, ReactJS hay các framework khác. MongoDB đặc biệt phù hợp với các hệ thống:

- Quản lý dữ liệu phi cấu trúc hoặc bán cấu trúc.
- Lưu trữ dữ liệu lớn, dữ liệu liên tục thay đổi hoặc dữ liệu phức tạp.
- Hệ thống yêu cầu khả năng mở rộng linh hoạt và hiệu suất truy xuất cao.

Lợi ích nổi bật

- Linh hoạt: Không cần cấu trúc cố định, dễ dàng thay đổi và mở rộng.
- Hiệu suất cao: Truy xuất và xử lý dữ liệu nhanh chóng.
- Dễ tích hợp: Kết hợp tốt với NodeJS, Python, Java, C# và các framework web hiện đại.
- Bảo mật và sẵn sàng cao: Hỗ trợ replication, backup, mã hóa dữ liệu.
- Quản lý dữ liệu đa dạng: Hỗ trợ dữ liệu phi cấu trúc, mảng, object lồng nhau.
- Khả năng mở rộng: Phù hợp với ứng dụng có lưu lượng cao hoặc dữ liệu liên tục tăng trưởng.

Ứng dụng thực tế

- Hệ thống quản lý khách hàng, sản phẩm trong thương mại điện tử.
- Lưu trữ dữ liệu người dùng cho mạng xã hội hoặc ứng dụng trực tuyến.
- Hệ thống quản lý doanh nghiệp, nhân sự, và các báo cáo thống kê.
- Ứng dụng kết hợp với các công cụ thời gian thực như chat, thông báo, dashboard.

MongoDB được đánh giá là lựa chọn lý tưởng cho các ứng dụng hiện đại nhờ khả năng linh hoạt, hiệu suất cao, dễ mở rộng, bảo mật tốt và tích hợp dễ dàng với các ngôn ngữ lập trình và framework phổ biến hiện nay.

2.5. Giới thiệu về ExpressJS

Express.js là một framework web nhẹ, linh hoạt và mạnh mẽ cho Node.js, được thiết kế để hỗ trợ xây dựng các ứng dụng web và API một cách nhanh chóng, hiệu quả và dễ bảo trì. Express.js giúp đơn giản hóa quá trình phát triển phía server bằng cách cung cấp một bộ công cụ toàn diện, cho phép xử lý yêu cầu HTTP, định tuyến, quản lý session, middleware, và nhiều tính năng quan trọng khác mà không cần viết quá nhiều mã nguồn.

Dựa trên Node.js

Express.js được phát triển dựa trên nền tảng Node.js, một môi trường JavaScript phía server. Điều này có nghĩa là các nhà phát triển có thể sử dụng cùng một ngôn ngữ là JavaScript cho cả phía client và server, giúp giảm thời gian học tập, dễ dàng đồng bộ logic giữa frontend và backend, đồng thời tạo ra một hệ sinh thái phát triển đồng nhất. Nhờ Node.js xử lý sự kiện bất đồng bộ (asynchronous event-driven), Express.js có khả năng xử lý hàng ngàn kết nối đồng thời mà vẫn duy trì hiệu suất cao, phù hợp cho các ứng dụng web có lượng truy cập lớn.

Kiến trúc nhẹ, tối ưu hiệu suất

Express.js nổi bật nhờ kiến trúc nhẹ, không kèm theo các tính năng thừa thãi, giúp tối ưu hóa tốc độ xử lý và khả năng mở rộng của ứng dụng. Kiến trúc này đặc biệt hữu ích cho các ứng dụng yêu cầu hiệu suất cao như thương mại điện tử, dịch vụ trực tuyến, ứng dụng mạng xã hội, hoặc các hệ thống thời gian thực. Với Express.js, việc thiết lập server cơ bản, định tuyến, và xử lý HTTP request trở nên nhanh chóng và trực quan, giúp tiết kiệm thời gian triển khai dự án.

Middleware – Lớp xử lý mạnh mẽ

Một trong những điểm nổi bật nhất của Express.js là hệ thống middleware. Middleware là các hàm được thực thi tuần tự giữa quá trình nhận yêu cầu từ client và trả phản hồi, có thể dùng để xử lý xác thực người dùng, ghi log, phân tích dữ liệu request, quản lý session, hoặc thực hiện các tác vụ tùy chỉnh khác. Middleware cho phép mở rộng ứng dụng một cách linh hoạt mà không làm ảnh hưởng đến cấu trúc chính, giúp phát triển các ứng dụng phức tạp trở nên dễ dàng hơn.

Hỗ trợ xây dựng RESTful API

Express.js là framework lý tưởng để phát triển API RESTful. Nó cung cấp các công cụ để tạo endpoint, xử lý HTTP request (GET, POST, PUT, DELETE...), và trả về phản hồi theo chuẩn REST. Điều này giúp xây dựng các dịch vụ web hoặc ứng dụng

di động hiệu quả, đồng thời dễ dàng tích hợp với frontend sử dụng ReactJS, Angular, Vue.js hoặc các công nghệ khác.

Hệ sinh thái và cộng đồng

Express.js sở hữu một cộng đồng phát triển lớn và một hệ sinh thái module phong phú, cho phép tích hợp nhanh các tính năng mở rộng như bảo mật, xác thực, logging, cache, xử lý file, và quản lý session. Sự phổ biến và hỗ trợ rộng rãi của cộng đồng giúp các nhà phát triển dễ dàng tìm kiếm tài liệu, ví dụ và giải pháp cho các vấn đề phát sinh trong quá trình phát triển ứng dụng.

Hỗ trợ view engine

Express.js hỗ trợ nhiều view engine để tạo ra các tệp HTML động, chẳng hạn như Pug (trước đây là Jade), EJS, Handlebars và các engine khác. Điều này giúp render dữ liệu từ server ra giao diện người dùng một cách linh hoạt, từ việc hiển thị danh sách sản phẩm, form nhập liệu, cho đến các bảng báo cáo động.

Bảo mật và quản lý

Express.js cung cấp các cơ chế để quản lý bảo mật và session, cho phép kiểm soát quyền truy cập, xác thực người dùng và bảo vệ dữ liệu. Khi kết hợp với các middleware và module phổ biến như Helmet (bảo vệ HTTP header), Passport (xác thực người dùng), hoặc express-session (quản lý session), ứng dụng có thể đảm bảo an toàn trước các tấn công web phổ biến như XSS, CSRF, hoặc SQL injection.

Ví dụ cơ bản:

```
const express = require('express');

const app = express();

// Định nghĩa một route cơ bản
app.get('/', (req, res) => {
  res.send('Hello, Express!');
});

// Khởi chạy server tại cổng 3000
app.listen(3000, () => {
  console.log('Server đang chạy tại http://localhost:3000');});
```

Lợi ích của Express.js

- Dễ học và sử dụng: Cú pháp đơn giản, logic rõ ràng, phù hợp cả với người mới học lập trình.
- Xử lý nhanh và hiệu quả: Kết hợp Node.js để xử lý hàng ngàn kết nối đồng thời.
- Tùy chỉnh linh hoạt: Cho phép thêm middleware và module tùy chỉnh theo nhu cầu dự án.
- Phát triển nhanh: Hệ sinh thái module phong phú giúp triển khai các tính năng nhanh chóng.
- Hỗ trợ API hiện đại: Là lựa chọn hàng đầu để xây dựng RESTful API cho web hoặc mobile.
- Bảo mật và quản lý dễ dàng: Hỗ trợ quản lý session, xác thực, và các module bảo vệ server.

Khi nào nên sử dụng Express.js

- Khi muốn xây dựng API hoặc backend nhanh chóng, hiệu quả.
- Khi phát triển ứng dụng web có cấu trúc linh hoạt, mở rộng dễ dàng.
- Khi làm việc với Node.js và muốn tận dụng hệ sinh thái JavaScript thống nhất.
- Khi cần xử lý dữ liệu thời gian thực hoặc lượng truy cập lớn, yêu cầu hiệu suất cao.

2.6. Giới thiệu về NodeJS

Node.js là một nền tảng chạy JavaScript phía server, được thiết kế để xây dựng các ứng dụng mạng nhanh chóng, hiệu quả và có khả năng mở rộng cao. Node.js được phát triển dựa trên V8 JavaScript Engine của Google, vốn là trình thông dịch JavaScript được sử dụng trong trình duyệt Chrome, nhờ đó có khả năng thực thi mã JavaScript với tốc độ cao. Điểm nổi bật của Node.js là kiến trúc hướng sự kiện (event-driven) và mô hình nhập/xuất không đồng bộ (asynchronous I/O), giúp xử lý nhiều yêu cầu từ client đồng thời mà không làm tắc nghẽn hệ thống, mang lại hiệu suất vượt trội cho các ứng dụng web, dịch vụ thời gian thực, hoặc các hệ thống có lượng người dùng lớn.

JavaScript phía server

Node.js cho phép các nhà phát triển sử dụng JavaScript ở cả phía client lẫn server, tạo nên sự thống nhất trong ngôn ngữ lập trình. Điều này giúp giảm bớt sự phức tạp trong phát triển ứng dụng đa tầng, đồng thời tăng năng suất khi không cần chuyển đổi ngôn ngữ giữa frontend và backend. Ngoài ra, khả năng tích hợp liền mạch với các thư

viện và framework JavaScript hiện đại như React.js, Vue.js, hoặc Angular giúp Node.js trở thành nền tảng lý tưởng cho các ứng dụng web và mobile hiện đại.

Kiến trúc event-driven và non-blocking I/O

Node.js sử dụng mô hình event-driven kết hợp non-blocking I/O, nghĩa là các tác vụ nhập/xuất dữ liệu sẽ không chặn luồng chính của ứng dụng. Khi có yêu cầu mới, Node.js sẽ đưa vào event loop, xử lý song song các tác vụ mà không tạo ra luồng mới, giúp tối ưu hiệu suất và giảm thiểu chi phí tài nguyên. Kiến trúc này đặc biệt phù hợp với các ứng dụng thời gian thực, ứng dụng mạng lớn hoặc hệ thống xử lý dữ liệu liên tục.

Cross-platform và khả năng mở rộng

Node.js hoạt động trên nhiều hệ điều hành phổ biến như Windows, macOS, Linux, và ngay cả các thiết bị nhúng như Raspberry Pi, giúp nó trở thành nền tảng linh hoạt cho nhiều loại dự án khác nhau. Node.js hỗ trợ mô hình đơn luồng nhưng vẫn xử lý hàng ngàn kết nối đồng thời nhờ Event Loop, tránh các vấn đề thường gặp ở hệ thống đa luồng như cạnh tranh tài nguyên hay deadlock.

Hỗ trợ WebSocket và ứng dụng thời gian thực

Node.js tích hợp tốt với WebSocket, cho phép xây dựng các ứng dụng real-time như trò chuyện trực tuyến, game nhiều người chơi, hệ thống thông báo, dashboard trực tiếp hay các ứng dụng IoT. Khả năng này giúp Node.js trở thành lựa chọn ưu tiên cho các ứng dụng cần phản hồi ngay lập tức với số lượng lớn người dùng đồng thời.

Ví dụ cơ bản:

```
// Import module HTTP có sẵn trong Node.js
const http = require('http');

// Tạo một server
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, Node.js!');
});
```

```
// Khởi động server tại cổng 3000
server.listen(3000, () => {
  console.log('Server đang chạy tại
http://localhost:3000/');
});
```

Ứng dụng của Node.js

- API Backend: Xây dựng các API RESTful hoặc GraphQL với các framework như Express.js, Koa, Nest.js.
- Ứng dụng real-time: Node.js lý tưởng cho chat, hệ thống thông báo, game online hay các ứng dụng live-streaming.
- Xử lý file và dữ liệu: Dễ dàng đọc, ghi, xóa file hoặc thao tác trên dữ liệu lớn mà không làm chậm hệ thống.
- Microservices: Node.js nhẹ và dễ mở rộng, phù hợp cho kiến trúc microservices phân tán.
- Ứng dụng IoT: Xử lý dữ liệu cảm biến, gửi/nhận tín hiệu real-time từ các thiết bị IoT.

Một số thư viện phổ biến

- Express.js: Framework web nhẹ và linh hoạt để xây dựng web server và API.
- Socket.io: Hỗ trợ giao tiếp thời gian thực giữa client và server.
- Mongoose: Thư viện tiện ích khi làm việc với MongoDB.
- jsonwebtoken (JWT): Xác thực người dùng qua token.
- Lodash: Cung cấp các hàm tiện ích để xử lý dữ liệu, mảng, đối tượng.

Lợi ích của Node.js

- Đơn giản hóa ngăn xếp công nghệ: Sử dụng JavaScript cả frontend và backend, dễ quản lý và đồng bộ mã nguồn.
- Hiệu suất cao: Xử lý hàng ngàn kết nối đồng thời nhờ kiến trúc event-driven và non-blocking I/O.
- Cộng đồng lớn: Cộng đồng phát triển sôi động, hỗ trợ nhiều module, thư viện và giải pháp thực tiễn.
- Tiết kiệm chi phí: Hiệu suất cao giúp giảm yêu cầu về tài nguyên phần cứng, tiết kiệm chi phí vận hành.

- Linh hoạt và mở rộng: Phù hợp cho các ứng dụng cần khả năng mở rộng cao hoặc xử lý dữ liệu liên tục.

Khi nào nên sử dụng Node.js

- Khi cần xây dựng các ứng dụng real-time như chat, thông báo hoặc game trực tuyến.
- Khi muốn phát triển API backend nhanh chóng và hiệu quả.
- Khi xử lý các ứng dụng có nhiều kết nối đồng thời, IoT, hoặc dữ liệu liên tục thay đổi.
- Khi cần nền tảng nhẹ, dễ mở rộng, tiết kiệm tài nguyên nhưng vẫn đảm bảo hiệu suất cao.

Chương 3: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

3.1. Xác định yêu cầu hệ thống

Trong quá trình phát triển hệ thống MedicalHope – một nền tảng hỗ trợ y tế từ thiện kết nối bệnh nhân, bác sĩ tình nguyện, nhà hảo tâm và ban quản trị – việc xác định yêu cầu hệ thống là một bước quan trọng nhằm đảm bảo toàn bộ quá trình thiết kế và triển khai diễn ra chính xác, hiệu quả. Giai đoạn này giúp đội ngũ phát triển hiểu rõ hệ thống cần làm gì, phục vụ ai, và phải vận hành như thế nào để đáp ứng đúng mục tiêu, tránh sai lệch hoặc phải chỉnh sửa nhiều lần về sau.

Trước hết, quá trình thu thập yêu cầu được thực hiện thông qua khảo sát, phân tích nghiệp vụ và trao đổi với các nhóm người dùng khác nhau. Mỗi nhóm tác nhân trong hệ thống – bao gồm bệnh nhân (người dùng), bác sĩ tình nguyện, quản lý từ thiện và admin – đều có nhu cầu, quyền hạn và hành vi riêng khi tương tác với hệ thống. Việc phân tích kỹ những nhu cầu này cho phép xây dựng các chức năng phù hợp, đảm bảo trải nghiệm người dùng mượt mà và trực quan.

Các yêu cầu hệ thống sau khi được thu thập sẽ được phân chia thành hai nhóm chính: yêu cầu chức năng và yêu cầu phi chức năng. Nhóm yêu cầu chức năng mô tả những nhiệm vụ cụ thể mà hệ thống cần cung cấp, chẳng hạn như đăng ký – đăng nhập, tạo và quản lý hồ sơ bệnh nhân, bác sĩ tư vấn, tiếp nhận và xử lý yêu cầu hỗ trợ, quản lý chiến dịch từ thiện và thống kê báo cáo. Trong khi đó, yêu cầu phi chức năng lại liên quan đến các tiêu chí về hiệu năng, bảo mật, tốc độ phản hồi, tính ổn định, khả năng mở rộng và trải nghiệm giao diện người dùng.

Ngoài ra, việc xác định yêu cầu còn đóng vai trò cơ sở cho các giai đoạn tiếp theo như thiết kế kiến trúc, phân tích dữ liệu, xây dựng sơ đồ Use Case và lập trình chức năng. Nhờ có yêu cầu rõ ràng, hệ thống MedicalHope có thể được phát triển theo đúng định hướng, đảm bảo toàn bộ người dùng – đặc biệt là bệnh nhân cần hỗ trợ – đều có thể tiếp cận dịch vụ một cách nhanh chóng, minh bạch và thuận tiện.

3.1.1. Yêu cầu chức năng

Về yêu cầu chức năng, hệ thống trước hết cần đáp ứng khả năng quản lý người dùng. Người dùng bao gồm bệnh nhân, bác sĩ tình nguyện, nhà hảo tâm và quản trị viên phải có thể đăng ký tài khoản, đăng nhập, đăng xuất và chỉnh sửa thông tin cá nhân của mình. Mỗi đối tượng sẽ có những quyền hạn và phạm vi thao tác riêng tùy theo vai trò. Bệnh nhân có thể tạo và cập nhật hồ sơ bệnh, mô tả tình trạng sức khỏe, hoàn cảnh khó

khẩn và theo dõi quá trình hỗ trợ từ hệ thống. Những thông tin này cần được quản trị viên kiểm duyệt để đảm bảo tính chính xác và minh bạch.

Đối với bác sĩ tình nguyện, hệ thống yêu cầu cung cấp chức năng đăng ký hồ sơ nghề nghiệp, bao gồm chuyên môn, kinh nghiệm và chứng chỉ liên quan. Quản trị viên sẽ xem xét và duyệt hồ sơ trước khi cấp quyền tham gia khám bệnh. Sau khi được duyệt, bác sĩ có thể xem danh sách bệnh nhân phù hợp với chuyên môn của mình và tiếp nhận ca khám khi cần thiết. Song song đó, hệ thống cũng hỗ trợ bệnh nhân đặt lịch khám trực tuyến, lựa chọn bác sĩ mong muốn và thời gian phù hợp. Bác sĩ có thể xác nhận hoặc từ chối lịch hẹn, trong khi quản trị viên theo dõi và điều phối lịch khám để đảm bảo hoạt động trơn tru.

Bên cạnh chức năng y tế, MedicalHope còn tích hợp nhiều hoạt động từ thiện. Hệ thống cần có khả năng quản lý các gói khám miễn phí, chương trình hỗ trợ như “chuyến xe 0 đồng”, suất ăn miễn phí, cùng danh sách các tổ chức từ thiện đang tham gia. Bệnh nhân có thể gửi yêu cầu hỗ trợ về thuốc men, thiết bị y tế, tài chính hoặc vận chuyển. Nhà hảo tâm có thể xem thông tin các yêu cầu này và tự nguyện hỗ trợ. Tất cả yêu cầu đều được quản trị viên kiểm duyệt và cập nhật trạng thái để đảm bảo minh bạch và đúng quy trình.

Hệ thống cũng cần tích hợp chatbot AI để hỗ trợ người dùng. Chatbot có khả năng trả lời các câu hỏi thường gặp, hướng dẫn bệnh nhân đăng ký, đặt lịch khám hoặc truy cập hồ sơ sức khỏe. Đây là chức năng quan trọng nhằm giảm tải cho bộ phận hỗ trợ và tăng tính chủ động cho người dùng. Ngoài ra, quản trị viên có quyền quản lý toàn bộ hệ thống bao gồm người dùng, bác sĩ, bệnh nhân, các chương trình hỗ trợ, yêu cầu từ thiện cũng như xem thống kê số lượng lịch khám, lượt hỗ trợ và tình trạng hoạt động của hệ thống.

3.1.2. Yêu cầu phi chức năng

Bên cạnh các yêu cầu chức năng, hệ thống MedicalHope cũng cần đáp ứng những yêu cầu phi chức năng quan trọng để đảm bảo hiệu quả vận hành. Về hiệu năng, hệ thống phải đảm bảo tốc độ tải trang nhanh, ổn định; chatbot cần phản hồi gần như tức thì để tối ưu trải nghiệm người dùng. Về bảo mật, tất cả mật khẩu đều phải được mã hóa an toàn, sử dụng JWT cho xác thực, phân quyền rõ ràng theo từng vai trò và có cơ chế bảo vệ khỏi các cuộc tấn công phổ biến như XSS, CSRF hoặc Injection.

Hệ thống cũng cần có khả năng mở rộng linh hoạt để đáp ứng sự tăng trưởng người dùng và mở rộng dịch vụ trong tương lai. Giao diện phải được thiết kế thân thiện, trực quan, dễ sử dụng cho mọi đối tượng, kể cả người lớn tuổi. Dữ liệu cần được xử lý

và đồng bộ theo thời gian thực, tránh trùng lặp hồ sơ và đảm bảo tính nhất quán. Ngoài ra, hệ thống phải có cơ chế sao lưu tự động và phục hồi dữ liệu khi xảy ra sự cố nhằm đảm bảo hoạt động liên tục, an toàn và ổn định.

3.2. Phân tích yêu cầu hệ thống

3.2.1. Chức năng đăng nhập

Chức năng đăng nhập được áp dụng cho ba đối tượng gồm Người dùng, Bác sĩ tình nguyện và Admin. Người dùng đăng nhập để truy cập các tính năng như đặt lịch khám, quản lý thông tin cá nhân và theo dõi yêu cầu hỗ trợ. Bác sĩ tình nguyện đăng nhập nhằm quản lý lịch rảnh, xem bệnh nhân đã đặt lịch và cập nhật thông tin nghề nghiệp. Admin đăng nhập để thực hiện các chức năng quản trị toàn hệ thống. Hệ thống sử dụng JWT để xác thực và phân quyền nhằm đảm bảo đúng vai trò khi truy cập. 3.2.2. Chức năng quản lý sự kiện và hoạt động

3.2.2. Chức năng quản lý sự kiện và hoạt động

MedicalHope hỗ trợ quản lý các chương trình, sự kiện, hoạt động khám chữa bệnh miễn phí hoặc hoạt động nhân đạo do bệnh viện hoặc tổ chức liên kết thực hiện. Admin có thể tạo mới sự kiện, cập nhật thông tin, theo dõi tiến độ và ngừng sự kiện khi cần. Người dùng có thể xem danh sách sự kiện, đăng ký tham gia và theo dõi các thông tin liên quan.

3.2.3. Chức năng quản lý gói khám

Hệ thống cho phép Admin tạo và quản lý các gói khám miễn phí hoặc gói khám đặc biệt dành cho bệnh nhân khó khăn. Người dùng có thể xem thông tin chi tiết và đăng ký nhận gói khám nếu đủ điều kiện. Admin có thể giới hạn số lượng, thời gian áp dụng và theo dõi danh sách người đã đăng ký.

3.2.4. Chức năng quản lý người dùng

Admin có thể xem danh sách người dùng trong hệ thống, bao gồm bệnh nhân và bác sĩ. Quản trị viên có quyền chỉnh sửa thông tin cơ bản của người dùng, khóa tài khoản nếu vi phạm và kiểm duyệt những dữ liệu quan trọng. Mục tiêu là đảm bảo an toàn, đúng thông tin và tránh tài khoản giả mạo.

3.2.5. Chức năng quản lý hỗ trợ

Bệnh nhân có thể gửi yêu cầu hỗ trợ như hỗ trợ khám bệnh, hỗ trợ phương tiện di chuyển, hỗ trợ bệnh án hoặc tư vấn. Admin tiếp nhận, xem xét và cập nhật trạng thái

từng yêu cầu. Chức năng này giúp hệ thống giải quyết các nhu cầu hỗ trợ một cách minh bạch, rõ ràng và có kiểm soát.

3.2.6. Chức năng quản lý lịch hẹn

Người dùng có thể đặt lịch khám với bác sĩ tình nguyện dựa trên thời gian rảnh mà bác sĩ đã cung cấp. Bác sĩ có quyền xác nhận hoặc từ chối lịch hẹn tùy vào điều kiện thực tế. Admin có thể theo dõi, điều phối hoặc can thiệp xử lý các trường hợp bất thường. Hệ thống lưu trữ và cập nhật trạng thái lịch hẹn theo thời gian thực.

3.2.7. Chức năng quản lý tổ chức

Hệ thống cho phép Admin quản lý thông tin các tổ chức liên kết như bệnh viện, đội khám tình nguyện hoặc đơn vị hỗ trợ. Admin có thể thêm mới, chỉnh sửa hoặc xóa tổ chức. Người dùng có thể xem thông tin để đăng ký tham gia các hoạt động do tổ chức đó phụ trách.

3.2.8. Chức năng quản lý đánh giá

Sau khi tham gia khám chữa bệnh hoặc sự kiện, người dùng có thể đánh giá bác sĩ hoặc chương trình. Đánh giá được lưu lại để cải thiện chất lượng dịch vụ. Admin có thể kiểm duyệt đánh giá và loại bỏ những nội dung sai lệch hoặc vi phạm quy tắc.

3.2.9. Chức năng quản lý bệnh nhân

Chức năng này cho phép bác sĩ và admin xem thông tin bệnh nhân bao gồm bệnh án, lịch sử khám, yêu cầu hỗ trợ và tình trạng sức khỏe. Bác sĩ chỉ có quyền truy cập thông tin của những bệnh nhân có lịch khám liên quan đến mình. Admin có toàn quyền giám sát để đảm bảo tính minh bạch và đầy đủ dữ liệu.

3.2.10. Chức năng quản lý thông tin cá nhân

Người dùng và bác sĩ có thể chỉnh sửa thông tin cá nhân như họ tên, địa chỉ, số điện thoại, email và ảnh đại diện. Các thay đổi được cập nhật trong hệ thống nhằm đảm bảo dữ liệu luôn đúng và phục vụ cho quá trình đặt lịch, liên hệ hoặc hỗ trợ.

3.2.11. Chức năng quản lý bác sĩ

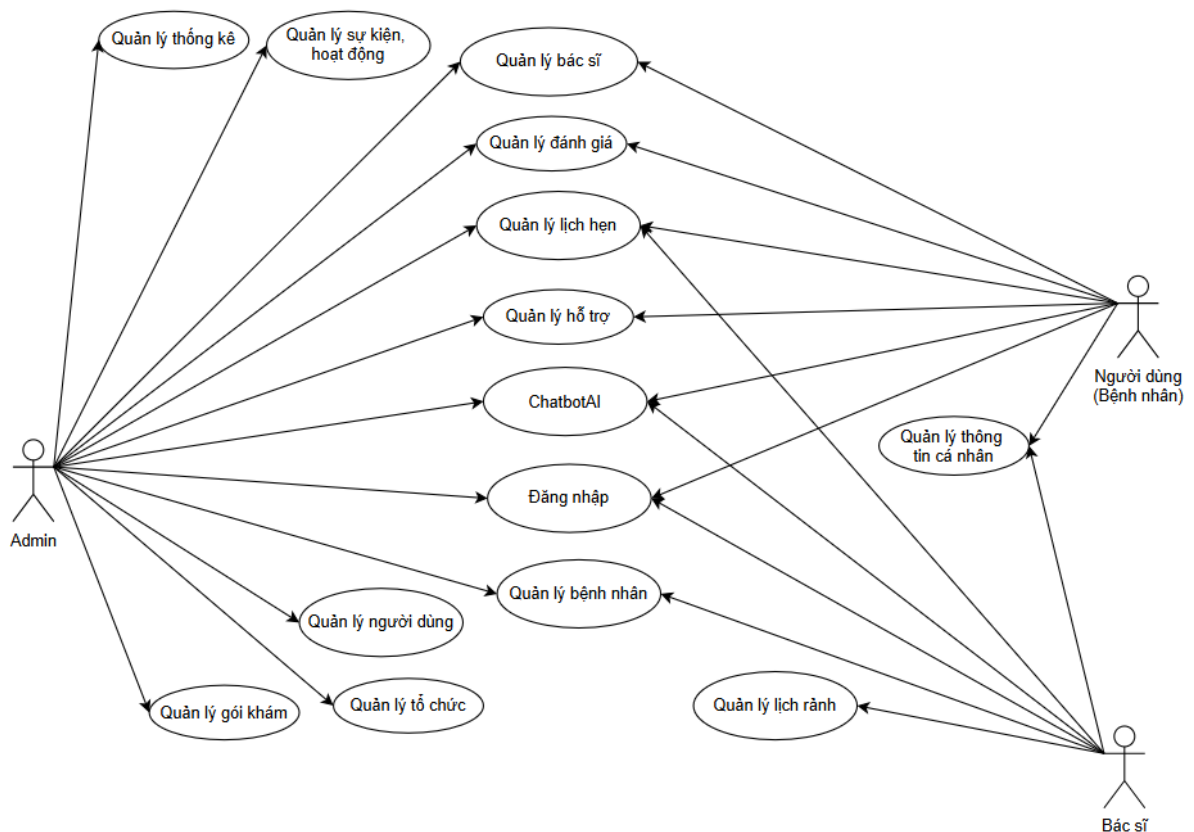
Admin có khả năng xem và kiểm duyệt hồ sơ bác sĩ tình nguyện, bao gồm chuyên môn, bằng cấp và kinh nghiệm. Bác sĩ có thể cập nhật thông tin nghề nghiệp và tình trạng hoạt động của mình. Admin có thể vô hiệu hóa hoặc kích hoạt lại tài khoản bác sĩ khi cần thiết.

3.2.12. Chức năng Chatbot AI

Chatbot AI được tích hợp để hỗ trợ người dùng trong việc đặt lịch khám, tìm kiếm thông tin, hướng dẫn sử dụng hệ thống hoặc trả lời câu hỏi thường gặp. Chatbot giúp giảm tải cho bộ phận hỗ trợ và tăng tính chủ động cho người dùng. Hệ thống đảm bảo chatbot phản hồi nhanh, chính xác và phù hợp ngữ cảnh.

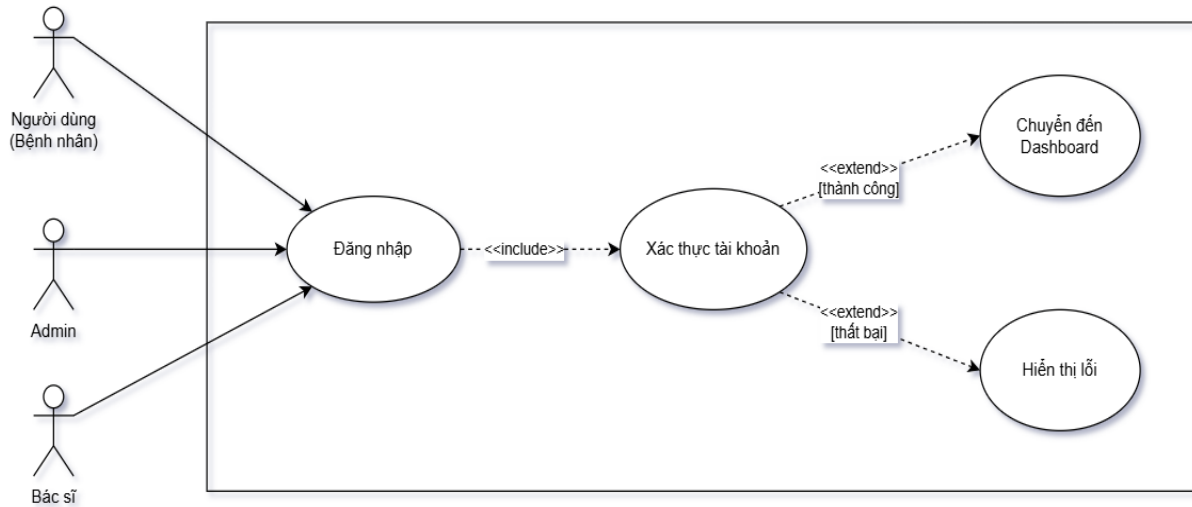
3.3. Mô hình hóa yêu cầu

- Use Case tổng quát



Hình 3.1 Biểu đồ Use Case tổng quát

3.3.1. Chức năng đăng nhập



Hình 3.2 Biểu đồ Use Case đăng nhập

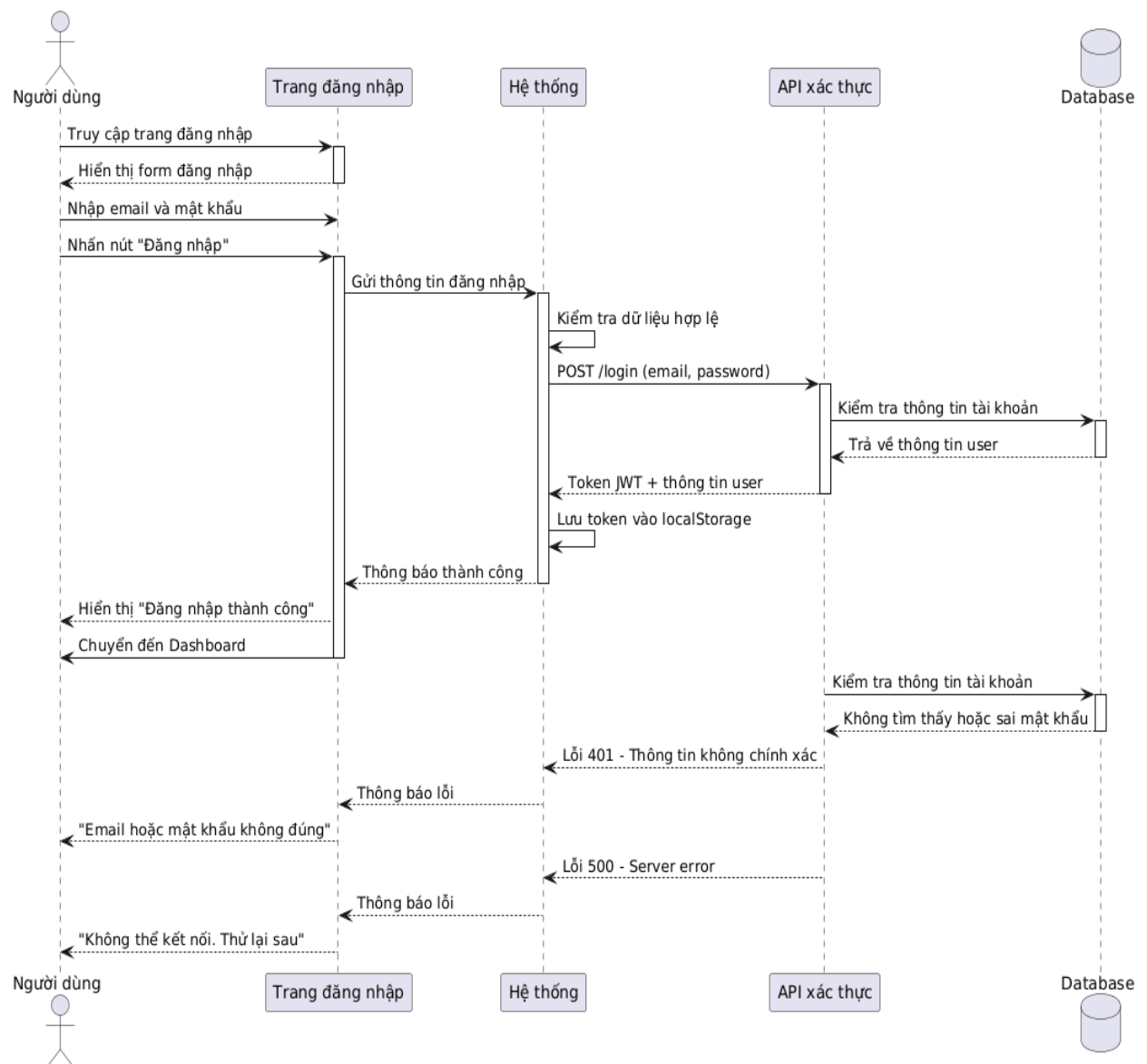
- **Mô tả Use Case đăng nhập**

Bảng 3.1 Mô tả Use Case đăng nhập

Use Case	Đăng nhập
Tác nhân chính	Người dùng (Bệnh nhân), Bác sĩ tình nguyện, Admin
Mô tả	Use case cho phép tác nhân truy cập hệ thống bằng email và mật khẩu hợp lệ. Hệ thống xác thực thông tin và điều hướng đến Dashboard theo vai trò.
Mục tiêu	Đảm bảo người dùng truy cập hệ thống an toàn, đúng quyền, đúng vai trò.
Tiền điều kiện	• Người dùng đã có tài khoản • Tài khoản không bị khóa • API xác thực hoạt động ổn định
Hậu điều kiện	Nếu thành công: • Tạo JWT token • Lưu thông tin người dùng • Chuyển đến Dashboard Nếu thất bại: • Không tạo token • Hiển thị thông báo lỗi
Luồng sự kiện chính	1. Truy cập trang đăng nhập 2. Hệ thống hiển thị form đăng nhập 3. Nhập email và mật khẩu 4. Nhấn nút 'Đăng nhập' 5. Hệ thống kiểm tra dữ liệu 6. Gửi yêu cầu API 7. API xác thực tài khoản

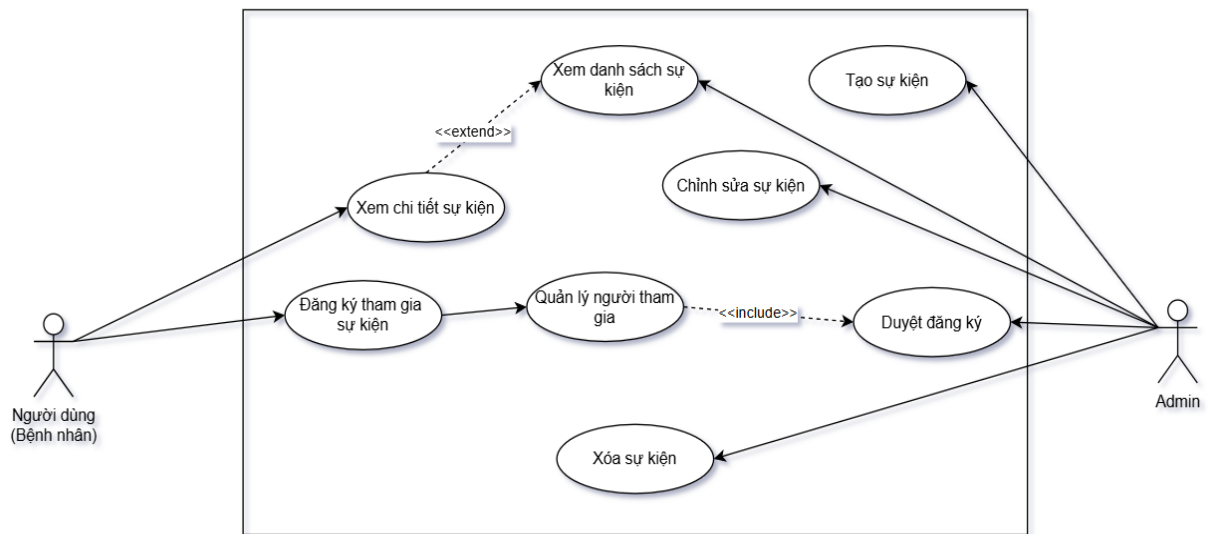
	8. Trả về token nếu hợp lệ 9. Lưu token và thông tin user 10. Điều hướng đến Dashboard
Luồng thay thế	A1 – Email không hợp lệ A2 – Thiếu email hoặc mật khẩu A3 – Sai thông tin đăng nhập A4 – Tài khoản bị khóa A5 – Lỗi hệ thống/API
Yêu cầu đặc biệt	• Mật khẩu mã hóa • Dùng JWT • Hỗ trợ ẩn/hiện mật khẩu

• Biểu đồ tuần tự đăng nhập



Hình 3.3 Biểu đồ tuần tự đăng nhập

3.3.2. Chức năng quản lý sự kiện và hoạt động



Hình 3.4 Biểu đồ Use Case quản lý sự kiện và hoạt động

- **Mô tả Use Case Quản lý sự kiện và hoạt động**

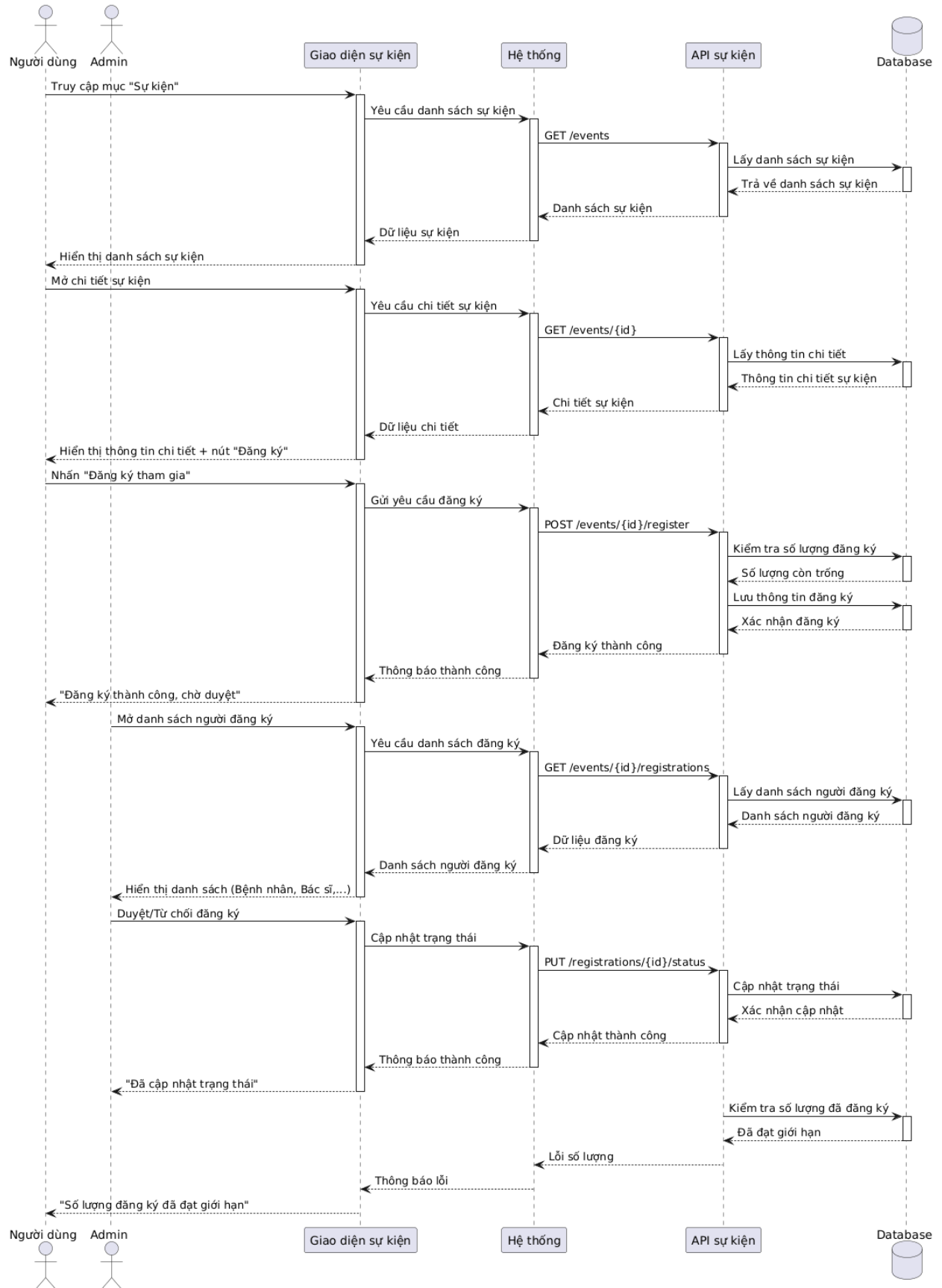
Bảng 3.2 Mô tả Use Case quản lý sự kiện và hoạt động

Use Case	Quản lý sự kiện và hoạt động
Tác nhân	- Người dùng - Admin
Mô tả	Use case mô tả toàn bộ nghiệp vụ tạo – sửa – xoá – xem sự kiện, đăng ký tham gia, duyệt người tham gia.
Tiền điều kiện	- Người dùng không cần đăng nhập vẫn có thể xem và đăng ký sự kiện - Admin đã đăng nhập hệ thống - Quản lý hoặc Admin mới có quyền tạo/sửa/xoá - Sự kiện có đầy đủ thông tin
Hậu điều kiện	- Sự kiện được cập nhật vào hệ thống - Danh sách người tham gia được ghi nhận
Luồng chính	6.1 Tạo sự kiện 6.2 Xem danh sách sự kiện 6.3 Chỉnh sửa sự kiện 6.4 Xoá sự kiện 6.5 Đăng ký tham gia 6.6 Quản lý người tham gia
Luồng thay thế	- Tìm kiếm & lọc sự kiện - Thống kê sự kiện
Ngoại lệ	- Thiếu thông tin: Báo lỗi - Trùng thời gian: Yêu cầu điều chỉnh - Đăng ký quá số lượng: Cảnh báo

- **Biểu đồ tuần tự quản lý sự kiện**

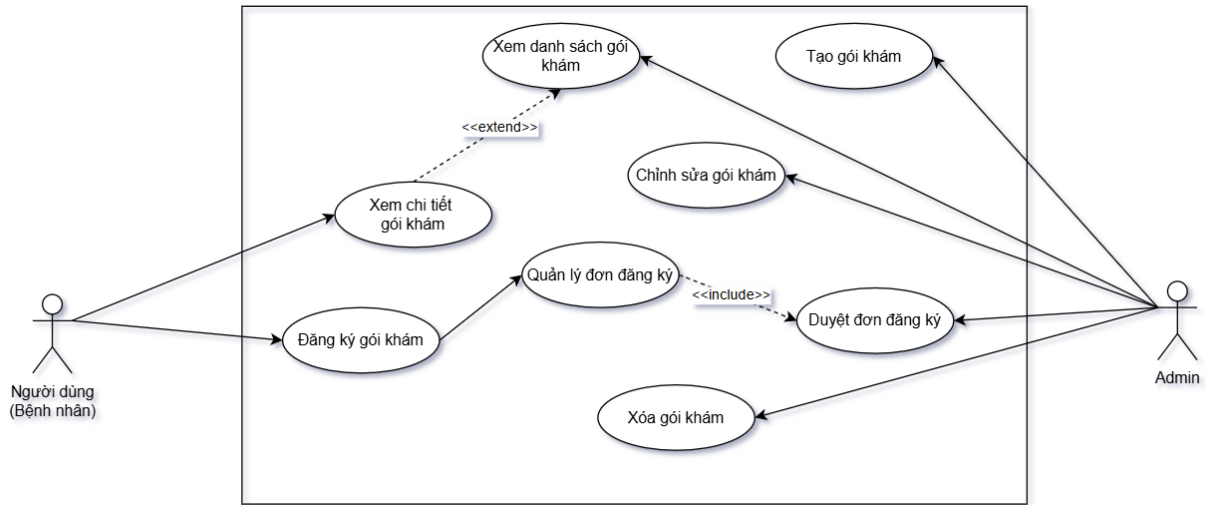


Hình 3.5 Biểu đồ tuần tự quản lý sự kiện cho admin



Hình 3.6 Biểu đồ tuần tự đăng ký sự kiện cho user, xem và duyệt admin

3.3.3. Chức năng quản lý gói khám



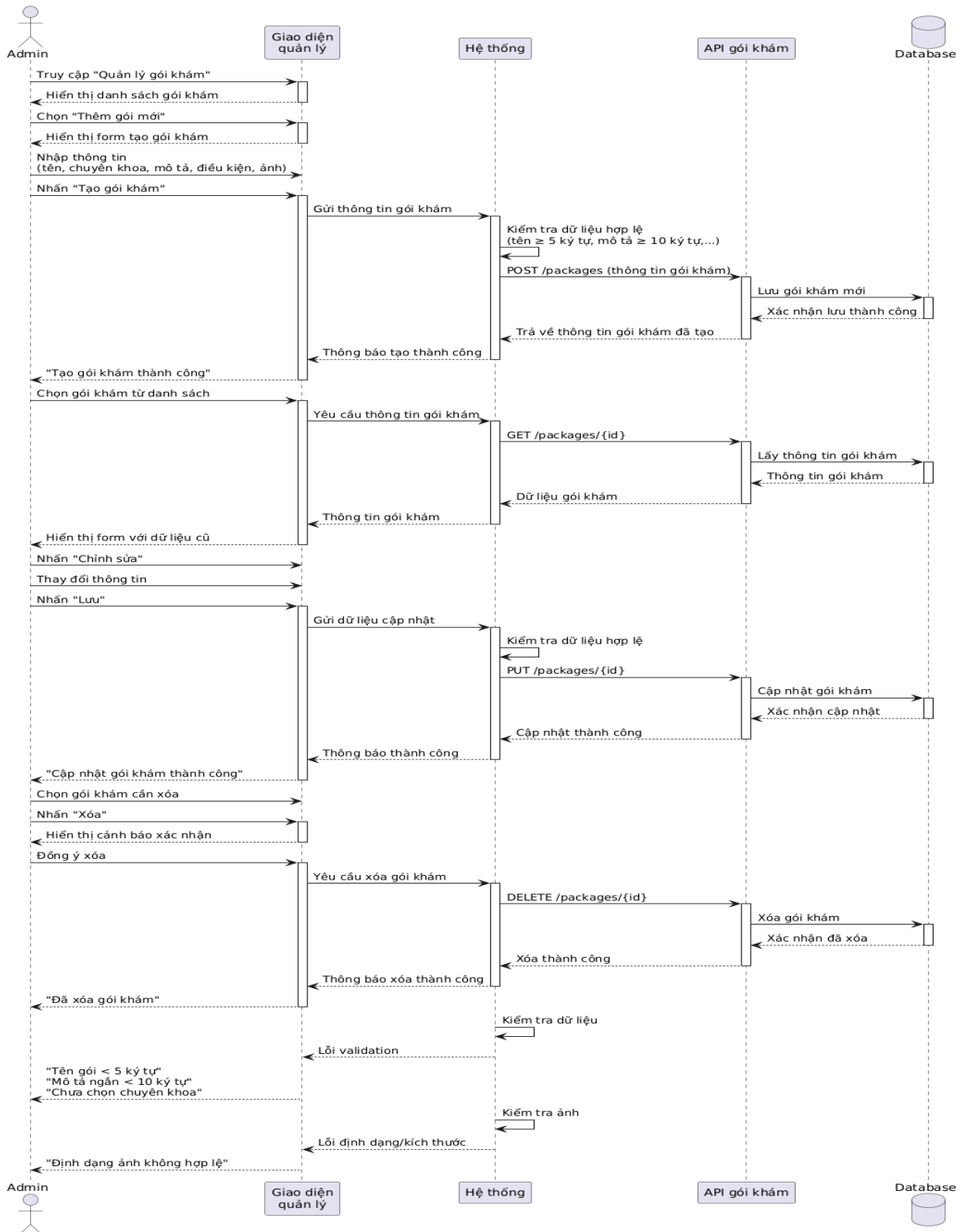
Hình 3.7 Biểu đồ Use Case quản lý gói khám

- **Mô tả Use Case Quản lý gói khám**

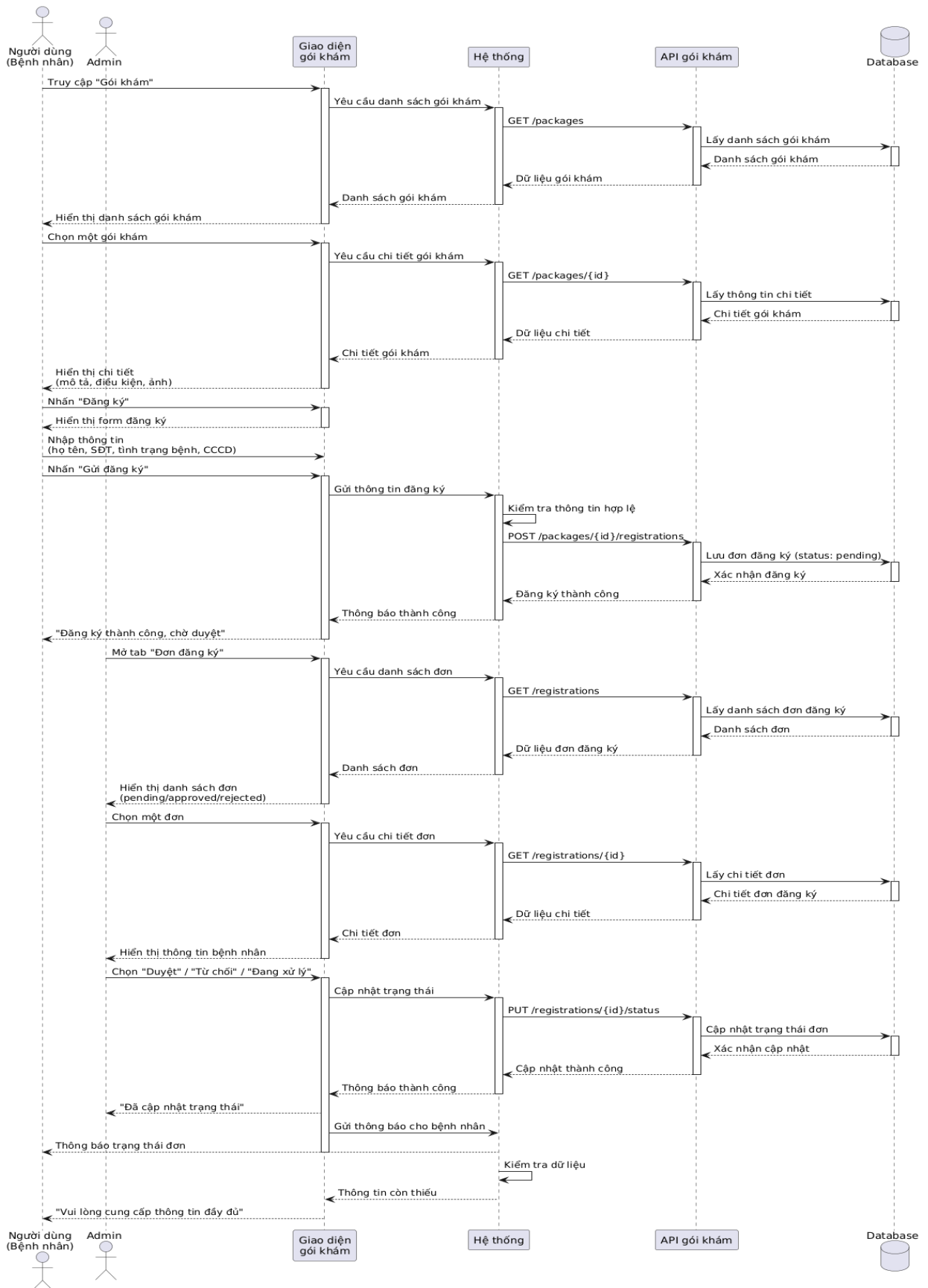
Bảng 3.3 Mô tả Use Case Quản lý gói khám

Use Case	Quản lý gói khám
Mô tả	Hệ thống cho phép Admin tạo, chỉnh sửa, xóa và quản lý các gói khám miễn phí hoặc gói khám đặc biệt dành cho bệnh nhân khó khăn. Người dùng có thể xem thông tin và đăng ký nếu đủ điều kiện. Admin theo dõi và duyệt đơn.
Tác nhân	Admin, Người dùng (bệnh nhân)
Tiền điều kiện	Admin đã đăng nhập; Hệ thống có dữ liệu chuyên khoa; Người dùng đăng ký phải có tài khoản hoặc cung cấp thông tin đầy đủ.
Hậu điều kiện	Gói khám được cập nhật; Danh sách đăng ký được ghi nhận; Trạng thái đơn đăng ký được cập nhật.
Luồng sự kiện chính	Luồng 1: Admin tạo gói khám Luồng 2: Admin chỉnh sửa gói khám Luồng 3: Admin xóa gói khám Luồng 4: Người dùng xem và đăng ký gói khám Luồng 5: Admin duyệt đơn đăng ký
Luồng thay thế	A1: Lỗi ảnh A2: Người dùng chưa đăng nhập A3: Admin từ chối đơn
Luồng ngoại lệ	E1: Server lỗi E2: Không tải được danh sách đơn E3: Sai format dữ liệu nhập

- **Biểu đồ tuần tự quản lý gói khám**

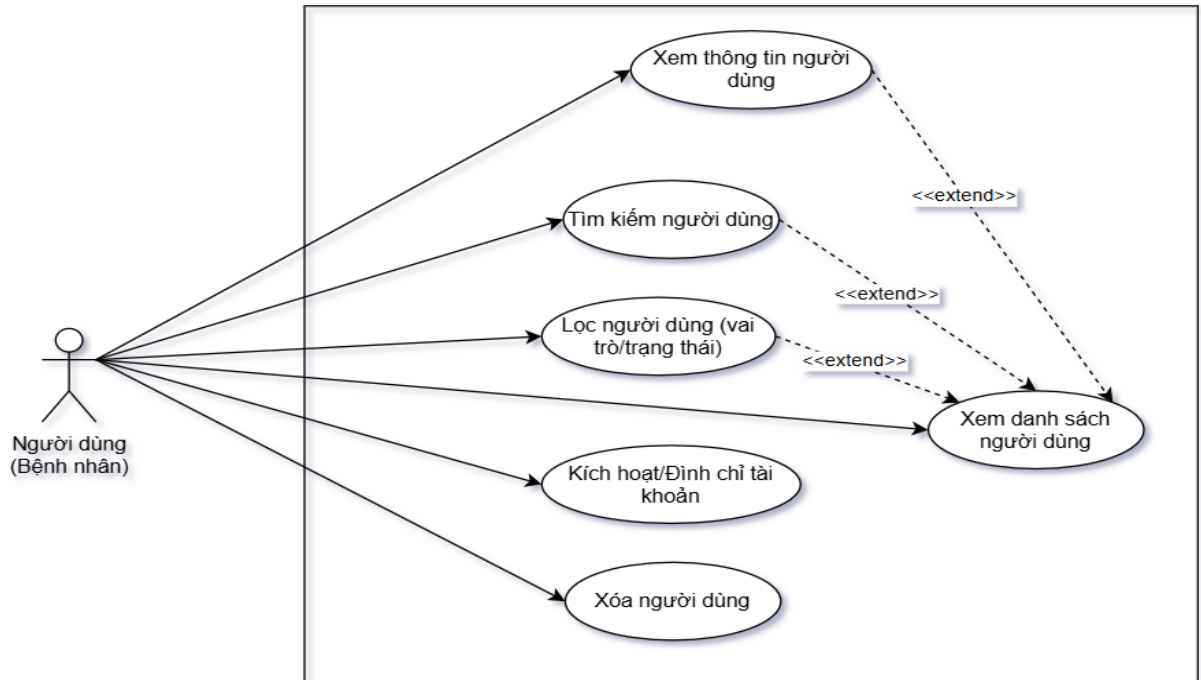


Hình 3.8 Biểu đồ tuần tự quản lý gói khám



Hình 3.9 Biểu đồ tuần tự đăng ký gói khám cho user, xem và duyệt cho admin

3.3.4. Chức năng quản lý người dùng



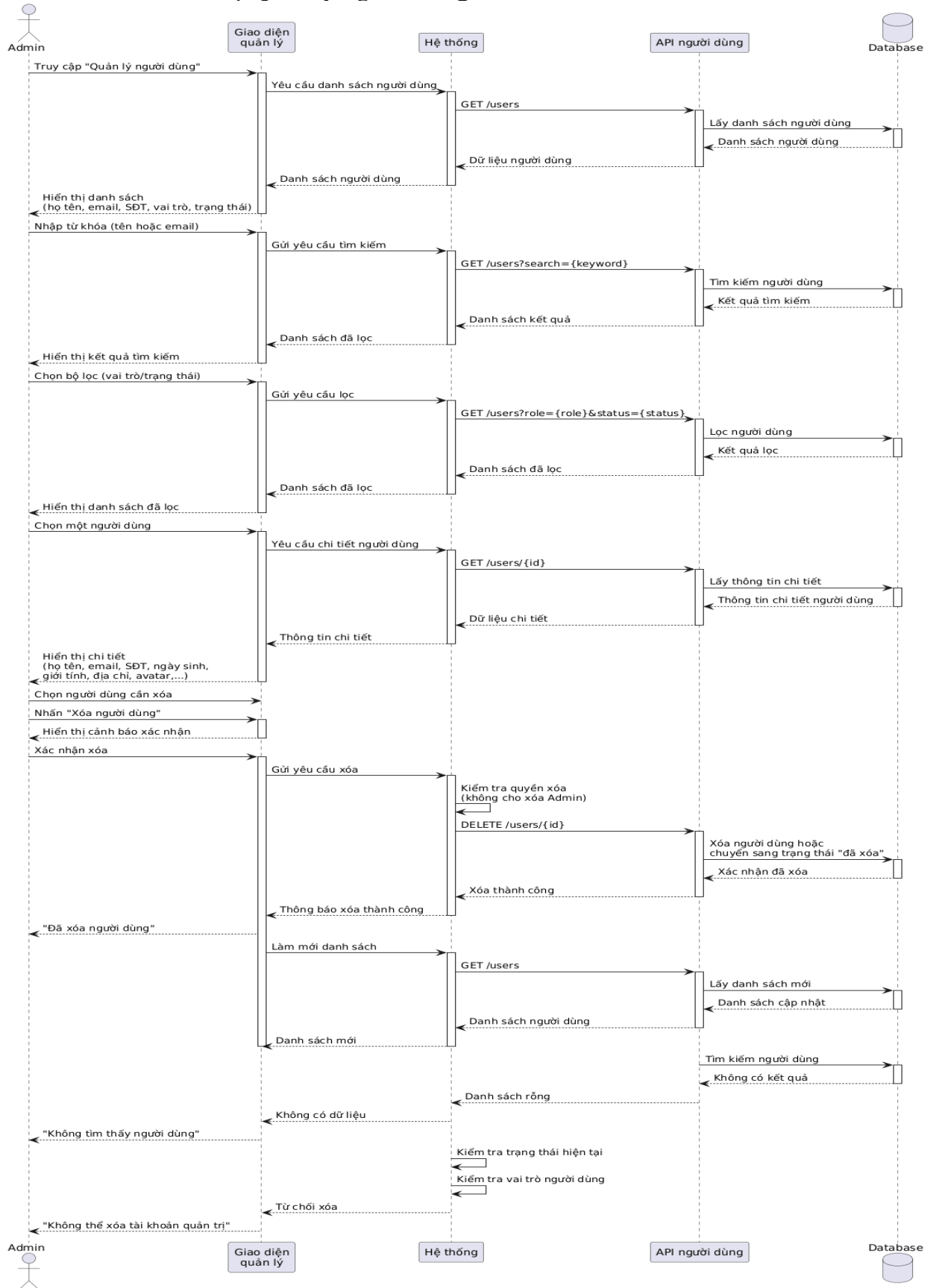
Hình 3.10 Biểu đồ Use Case quản lý người dùng

- Mô tả Use Case Quản lý người dùng**

Bảng 3.4 Mô tả Use Case Quản lý người dùng

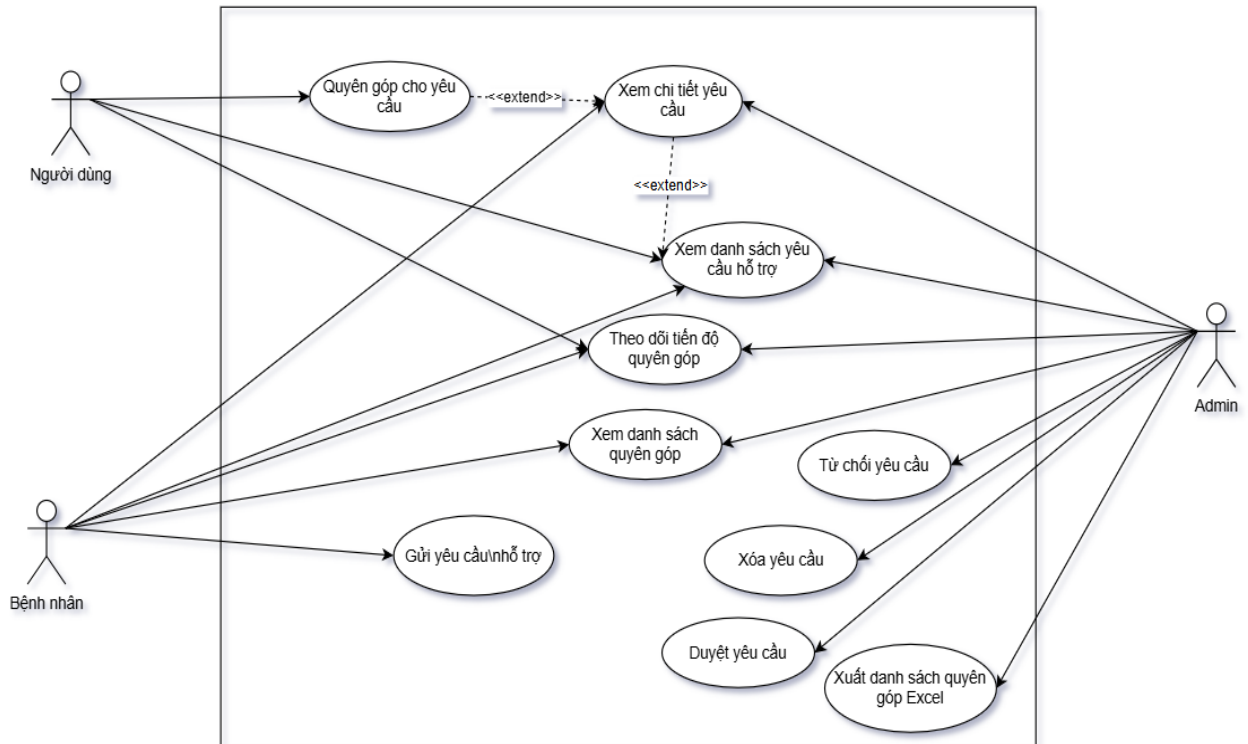
Tên Use Case	Quản lý người dùng
Tác nhân	Admin
Mô tả tóm tắt	Admin quản lý toàn bộ người dùng trong hệ thống MedicalHope: xem danh sách, tìm kiếm, lọc, chỉnh sửa trạng thái, kích hoạt/đình chỉ và xóa tài khoản.
Tiền điều kiện	Admin đã đăng nhập; Có quyền truy cập quản lý người dùng; Dữ liệu người dùng tồn tại trong hệ thống.
Hậu điều kiện	Thông tin người dùng được cập nhật; Thay đổi được lưu và hiển thị lại.
Luồng sự kiện chính	Luồng 1: Xem danh sách người dùng Luồng 2: Tìm kiếm người dùng Luồng 3: Lọc người dùng Luồng 4: Kích hoạt / Đình chỉ tài khoản Luồng 5: Xóa tài khoản Luồng 6: Xem chi tiết người dùng
Luồng thay thế	A1: Không tìm thấy người dùng A2: Tài khoản đã đình chỉ A3: Không thể xóa tài khoản Admin
Yêu cầu đặc biệt	Hiển thị avatar; Tìm kiếm nhanh; Phân trang; Chỉ Admin được phép xóa hoặc đình chỉ.

• **Biểu đồ tuần tự quản lý người dùng**



Hình 3.11 Biểu đồ tuần tự quản lý người dùng

3.3.5. Chức năng quản lý hỗ trợ



Hình 3.12 Biểu đồ Use Case quản lý hỗ trợ

- **Mô tả Use Case Quản lý hỗ trợ**

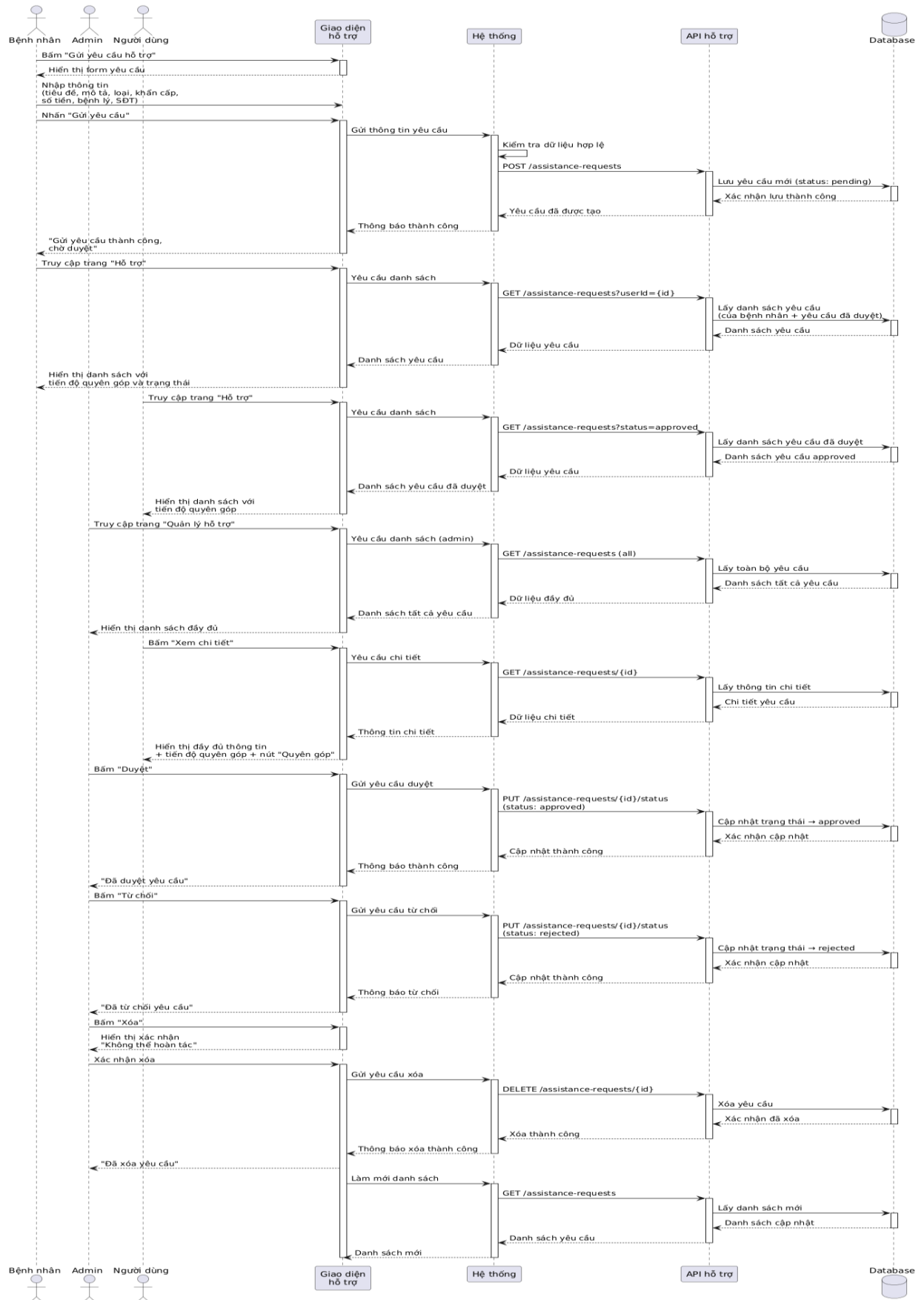
Bảng 3.5 Mô tả Use Case Quản lý hỗ trợ

Use Case	Quản lý hỗ trợ
Tác nhân	- Admin (quản lý toàn bộ yêu cầu hỗ trợ và quyên góp) - Người dùng (nhà hảo tâm, người quyên góp) - Bệnh nhân (gửi yêu cầu hỗ trợ và theo dõi tiến độ)
Mô tả tóm tắt	Use case cho phép quản lý xuyên suốt toàn bộ quy trình hỗ trợ bệnh nhân trong hệ thống MedicalHope, bao gồm: bệnh nhân gửi yêu cầu hỗ trợ, admin kiểm duyệt yêu cầu, người dùng và bệnh nhân xem thông tin hỗ trợ, thực hiện quyên góp, theo dõi tiến độ và quản lý danh sách quyên góp. Toàn bộ hoạt động giúp đảm bảo tính minh bạch, nhanh chóng và chính xác trong công tác hỗ trợ y tế.
Tiền điều kiện	- Hệ thống hoạt động ổn định. - Bệnh nhân/Người dùng/Admin đã đăng nhập đúng vai trò. - Bệnh nhân có hồ sơ hợp lệ trước khi gửi yêu cầu hỗ trợ. - Các yêu cầu đã tồn tại trong hệ thống để xem, lọc hoặc quyên góp.

Hậu điều kiện	- Yêu cầu hỗ trợ được tạo, duyệt, từ chối hoặc cập nhật trạng thái. - Lịch sử quyền góp được ghi nhận và lưu trữ. - Danh sách quyền góp có thể được xuất ra file Excel. - Tất cả thay đổi được cập nhật và hiển thị cho các tác nhân liên quan.
Luồng sự kiện chính	Luồng 1: Bệnh nhân gửi yêu cầu hỗ trợ. Luồng 2: Admin/Người dùng/Bệnh nhân xem danh sách yêu cầu hỗ trợ. Luồng 3: Xem chi tiết yêu cầu hỗ trợ. Luồng 4: Theo dõi tiến độ quyền góp. Luồng 5: Người dùng/Bệnh nhân quyền góp cho yêu cầu. Luồng 6: Admin duyệt yêu cầu. Luồng 7: Admin từ chối yêu cầu. Luồng 8: Admin xóa yêu cầu. Luồng 9: Xem danh sách quyền góp theo từng yêu cầu hoặc theo hệ thống. Luồng 10: Admin xuất danh sách quyền góp ra file Excel.
Luồng thay thế	A1: Bệnh nhân gửi thiếu thông tin → hệ thống thông báo lỗi và yêu cầu bổ sung. A2: Yêu cầu đã bị từ chối hoặc chưa được duyệt → không cho phép quyền góp. A3: Không tìm thấy yêu cầu hỗ trợ → hiển thị thông báo. A4: Quyền góp thất bại do lỗi thanh toán → yêu cầu người dùng thử lại. A5: Admin không thể xóa yêu cầu đã có nhiều quyền góp → thông báo và hủy thao tác.
Yêu cầu đặc biệt	- Hệ thống phải đảm bảo tính minh bạch, hiển thị đầy đủ tiến độ và số tiền quyền góp. - Hỗ trợ phân trang, tìm kiếm, lọc theo loại yêu cầu, trạng thái, thời gian. - Hiển thị hình ảnh hồ sơ, giấy tờ minh chứng của bệnh nhân được duyệt. - File Excel phải đúng định dạng, đầy đủ cột: người quyền góp, số tiền, thời gian, mã giao dịch. - Mọi giao dịch quyền góp phải được ghi log để phục vụ kiểm tra.

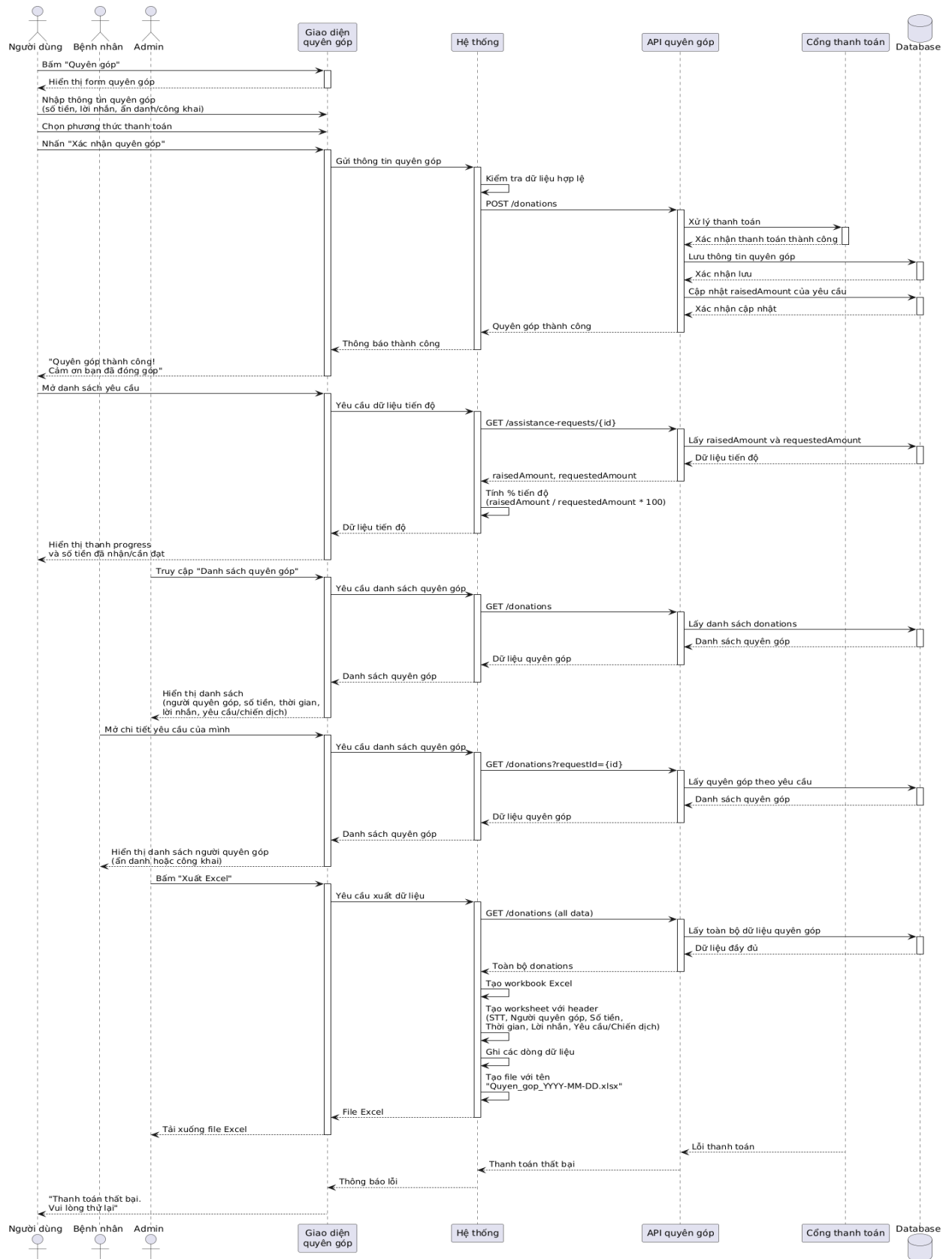
- **Biểu đồ tuần tự quản lý hỗ trợ**

Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI



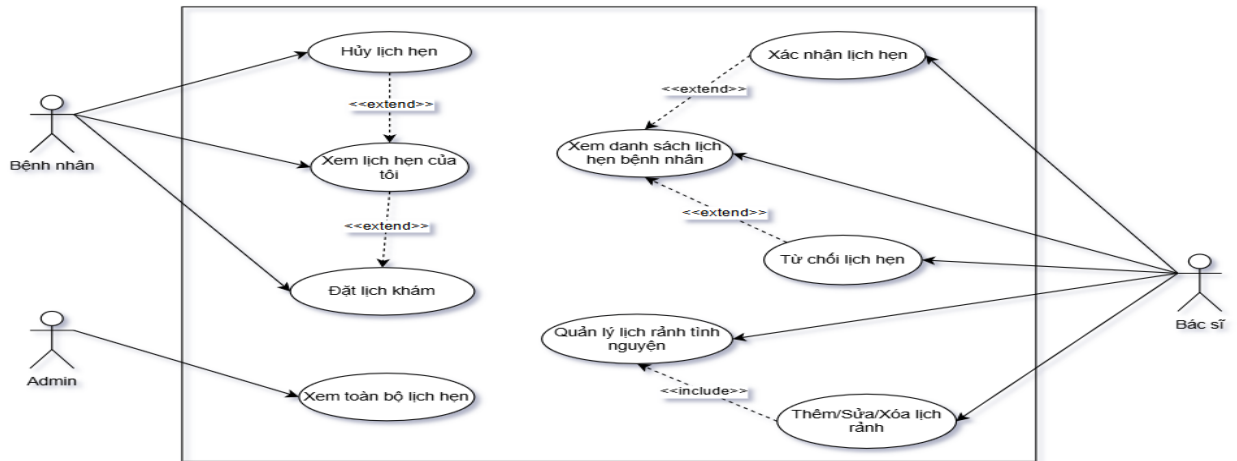
Hình 3.13 Biểu đồ tuần tự quản lý hỗ trợ(gửi yêu cầu hỗ trợ)

Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI



Hình 3.14 Biểu đồ tuần tự quản lý hỗ trợ(Tạo quyền góp), xem danh sách quyền góp.

3.3.6. Chức năng quản lý lịch hẹn



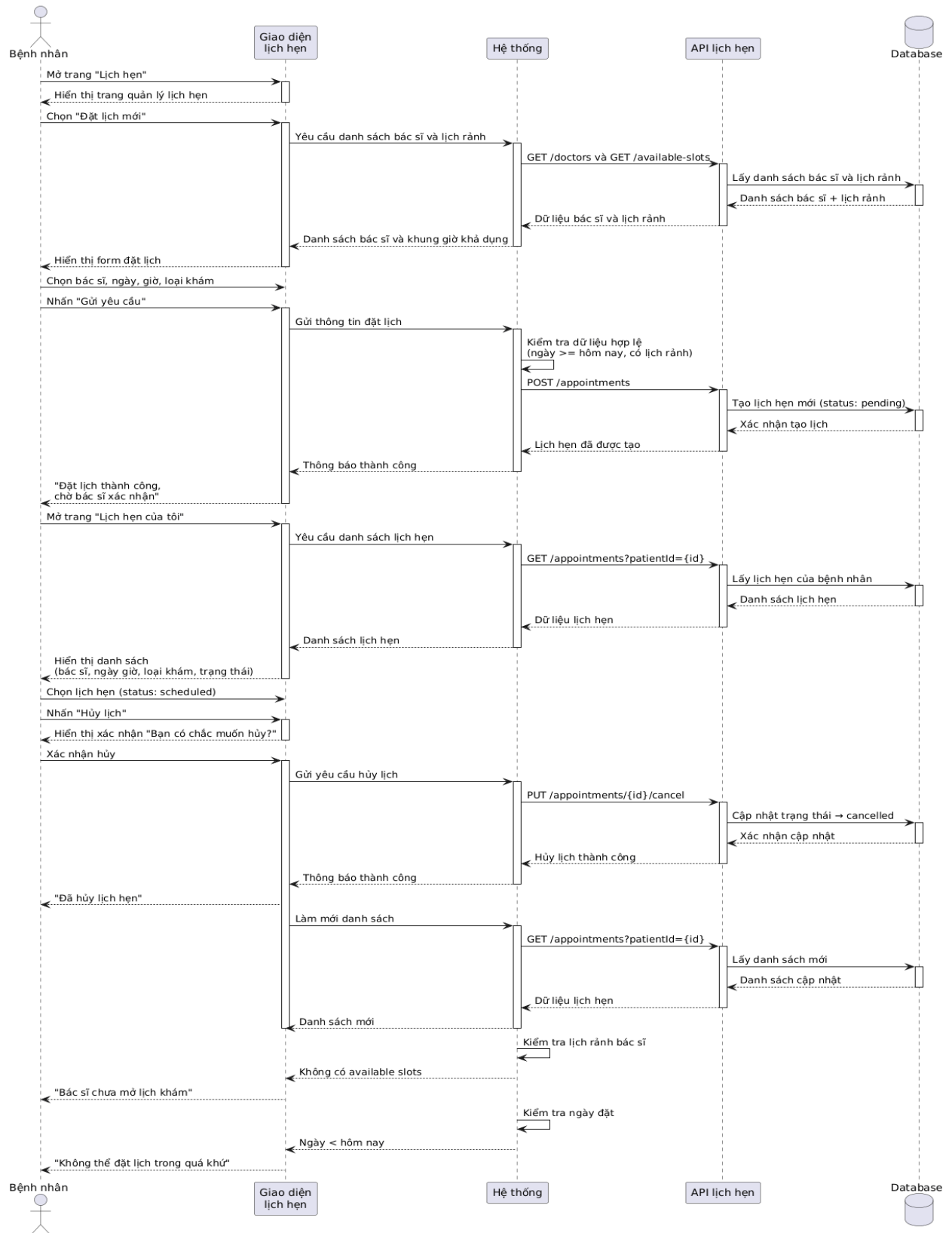
Hình 3.15 Biểu đồ Use Case quản lý lịch hẹn

- **Mô tả Use Case Quản lý lịch hẹn**

Bảng 3.6 Mô tả Use Case Quản lý lịch hẹn

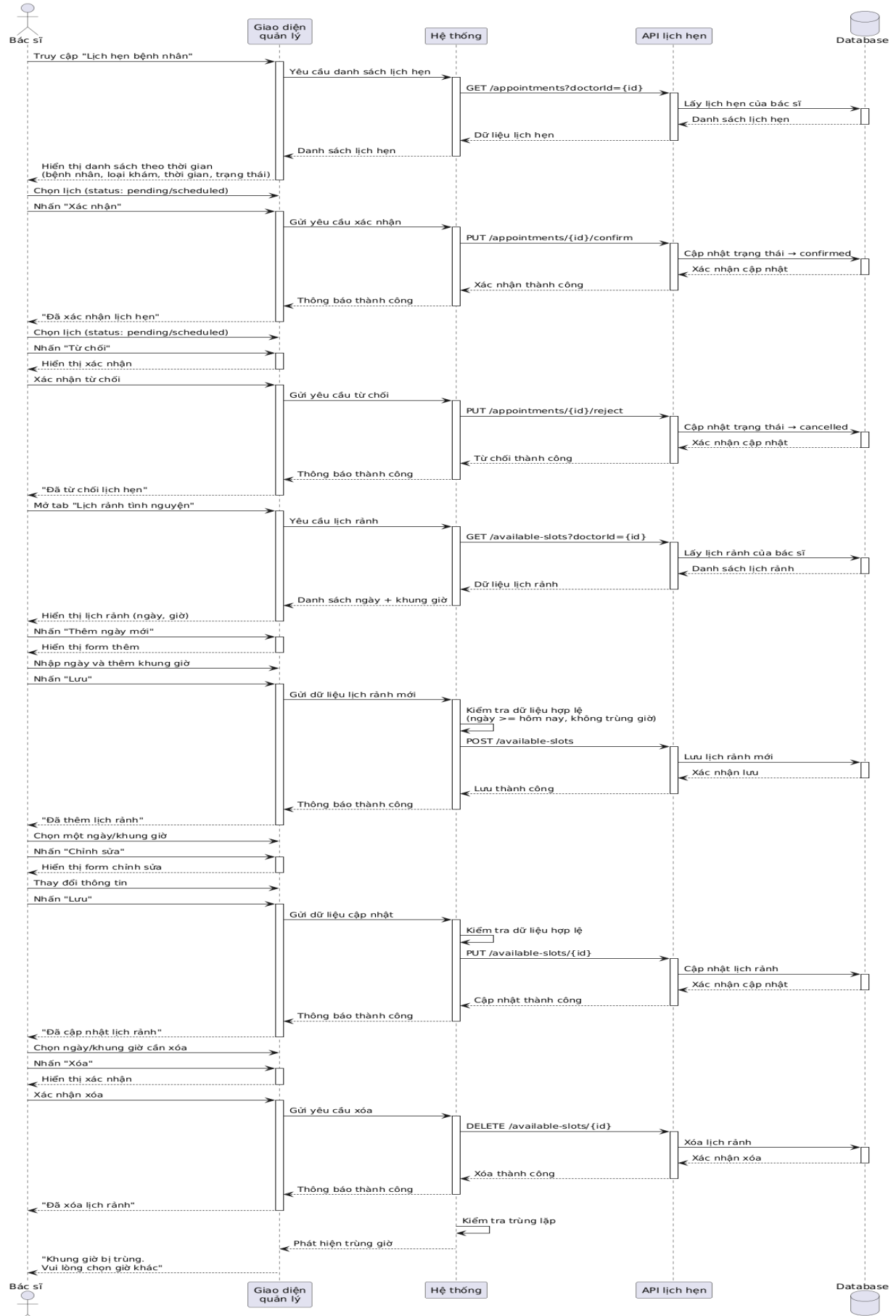
Use Case	Quản lý lịch hẹn
Tác nhân	- Bệnh nhân: Đặt lịch, xem lịch, hủy lịch - Bác sĩ: Quản lý lịch hẹn, xác nhận/từ chối, quản lý lịch rãnh khám tình nguyện
Mô tả tóm tắt	Quy trình quản lý lịch hẹn giữa bệnh nhân và bác sĩ, bao gồm đặt lịch, xem lịch, xác nhận/từ chối, quản lý lịch rãnh và hủy lịch.
Tiền điều kiện	- Người dùng đã đăng nhập đúng vai trò - Bệnh nhân đã có hồ sơ - Bác sĩ có tài khoản hợp lệ - Có dữ liệu lịch rãnh (optional)
Hậu điều kiện	- Lịch hẹn được lưu vào DB - Trạng thái cập nhật - Lịch rãnh được lưu và hiển thị cho bệnh nhân
Luồng sự kiện chính	- Bệnh nhân đặt lịch - Bác sĩ xem lịch - Bác sĩ xác nhận lịch - Hủy hoặc từ chối lịch - Bệnh nhân xem lịch của mình - Bác sĩ quản lý lịch rãnh - Sửa/xóa lịch rãnh
Luồng thay thế	- Không có lịch rãnh - Khung giờ bị trùng - Ngày trong quá khứ
Yêu cầu đặc biệt	- Chỉ bác sĩ chỉnh availableSlots - Chỉ bệnh nhân tạo lịch

- Biểu đồ tuần tự quản lý lịch hẹn



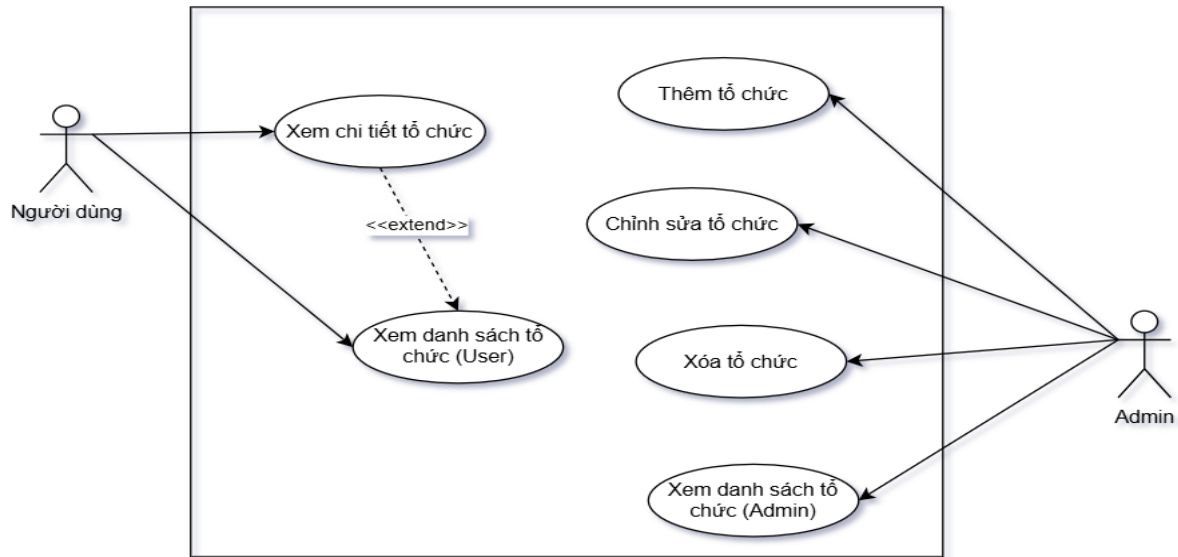
Hình 3.16 Biểu đồ quản lý lịch hẹn cho bệnh nhân

Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI



Hình 3.17 Biểu đồ tuần tự quản lý lịch hẹn cho bác sĩ

3.3.7. Chức năng quản lý tổ chức



Hình 3.18 Biểu đồ Use Case quản lý tổ chức

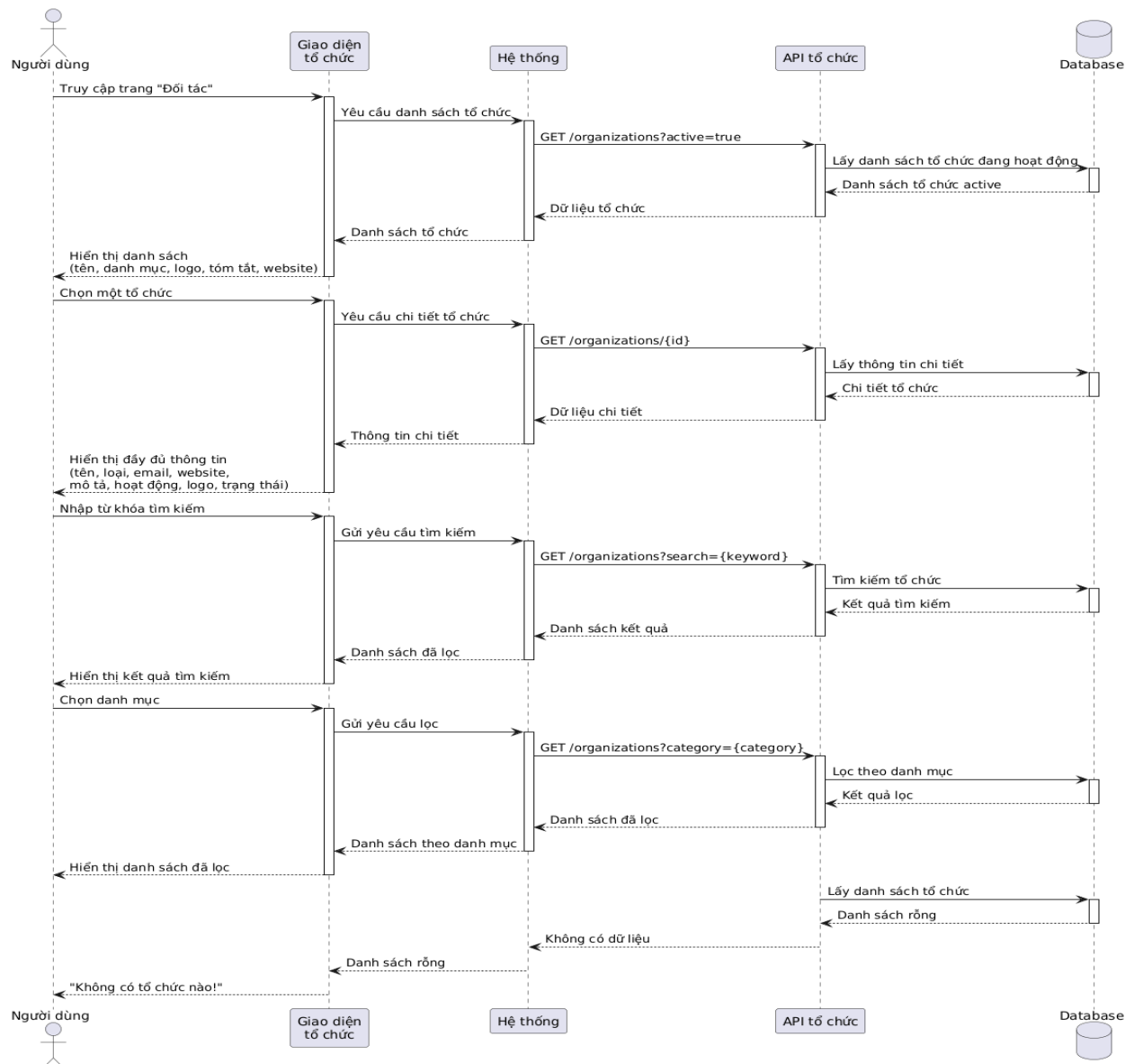
- **Mô tả Use Case Quản lý tổ chức**

Bảng 3.7 Mô tả Use Case Quản lý tổ chức

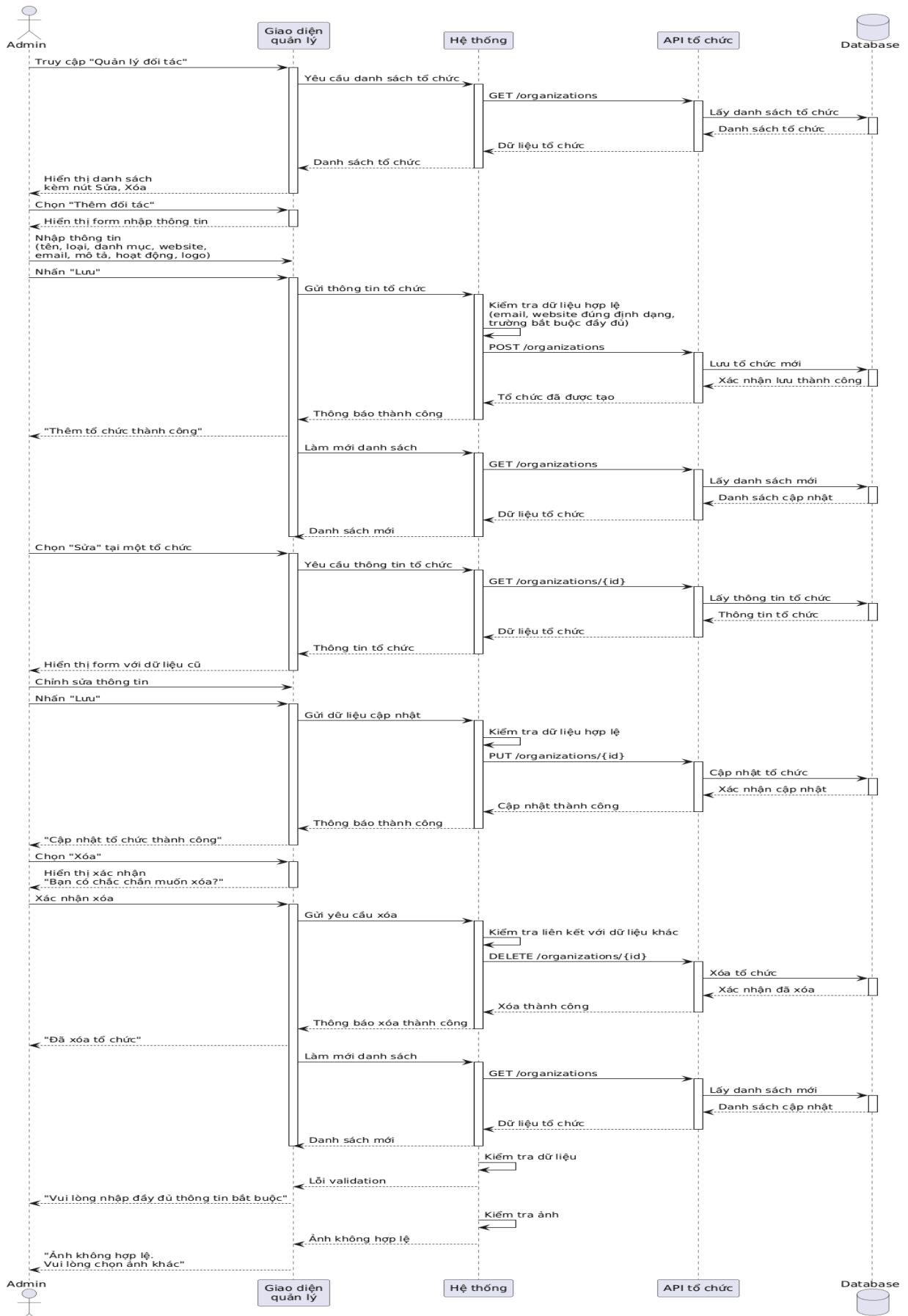
Use Case	Quản lý tổ chức
Tác nhân	• Admin: Toàn quyền quản lý tổ chức (thêm, sửa, xóa, bật/tắt hoạt động) • Người dùng: Xem danh sách và xem chi tiết tổ chức
Mô tả tóm tắt	Chức năng cho phép Admin quản lý toàn bộ tổ chức: thêm mới, chỉnh sửa thông tin, xóa hoặc thay đổi trạng thái. Người dùng có thể xem danh sách và chi tiết.
Tiền điều kiện	Đối với Admin: • Đã đăng nhập • Có quyền truy cập trang Quản lý đối tác/tổ chức Đối với Người dùng: • Có thể không cần đăng nhập (tùy hệ thống) • Truy cập được trang danh sách tổ chức
Hậu điều kiện	• Dữ liệu tổ chức được lưu/cập nhật/xóa đúng trên hệ thống • Người dùng xem được danh sách và chi tiết chính xác
Luồng sự kiện chính	A. Dành cho Admin UC1 – Thêm tổ chức: Thêm → Nhập thông tin → lưu UC2 – Chỉnh sửa tổ chức: Sửa → Lưu UC3 – Xóa tổ chức: Xóa → Xác nhận UC4 – Xem danh sách (dạng quản trị) B. Dành cho Người dùng UC5 – Xem danh sách tổ chức đang hoạt động UC6 – Xem chi tiết tổ chức

Luồng thay thế	UC1A – Thêm tổ chức thất bại (thiếu trường, lỗi ảnh) UC3A – Xóa thất bại (đang liên kết dữ liệu khác) UC5A – Không có tổ chức
Ràng buộc	- Chỉ Admin có quyền CRUD - Người dùng chỉ được xem - Website/email phải hợp lệ

• Biểu đồ tuần tự quản lý tổ chức

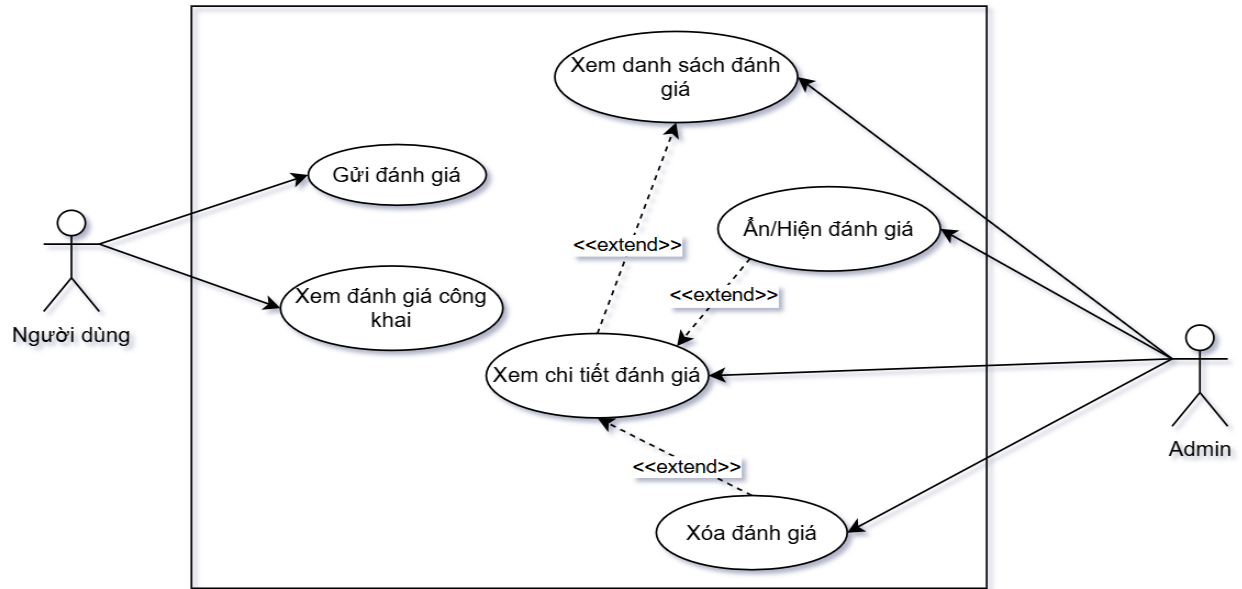


Hình 3.19 Biểu đồ tuần tự quản lý đối tác(xem danh sách) cho người dùng



Hình 3.20 Biểu đồ tuần tự quản lý đối tác cho admin

3.3.8. Chức năng quản lý đánh giá



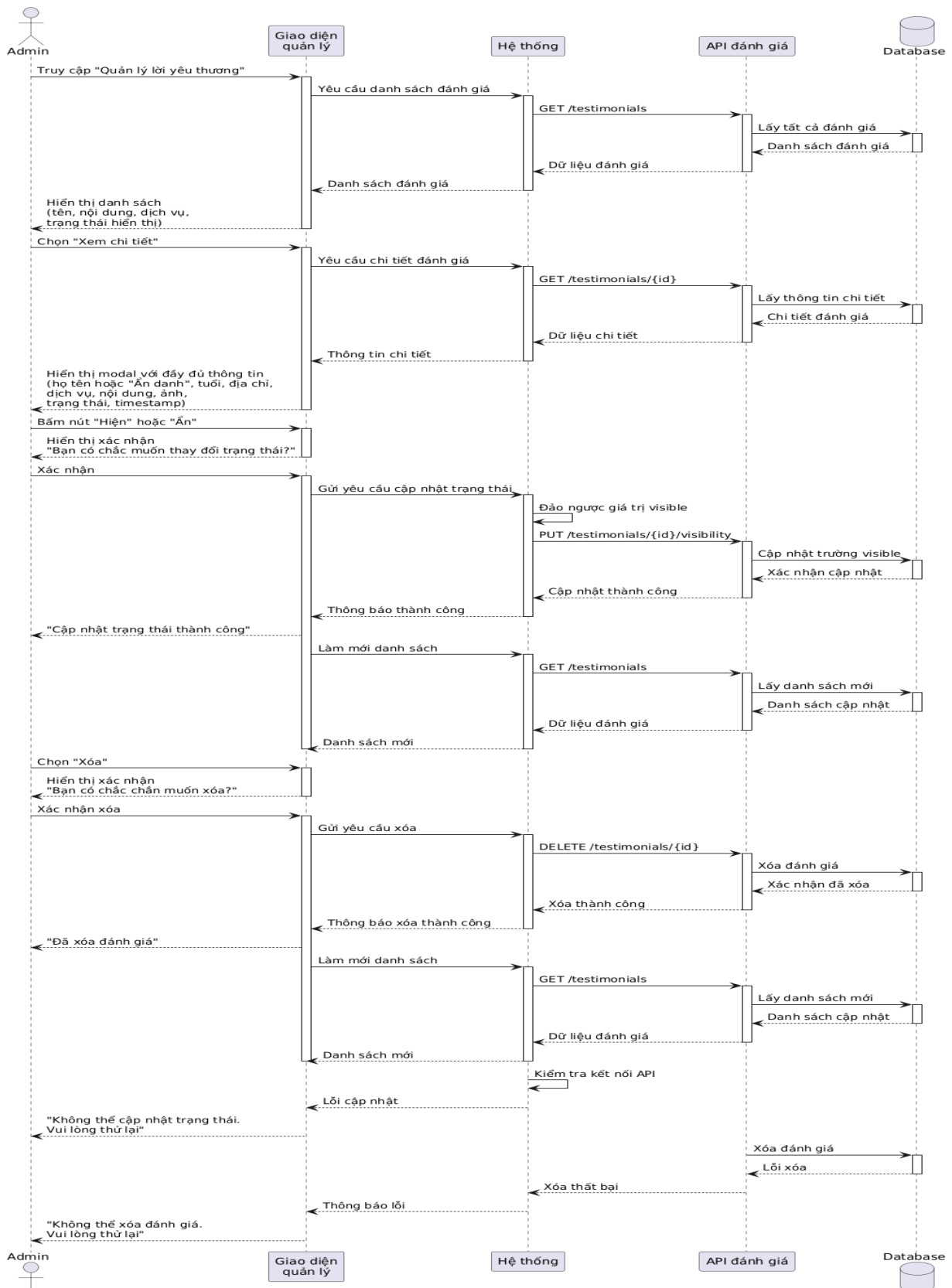
Hình 3.21 Biểu đồ Use Case quản lý đánh giá

- **Mô tả Use Case Quản lý đánh giá**

Bảng 3.8 Mô tả Use Case Quản lý đánh giá

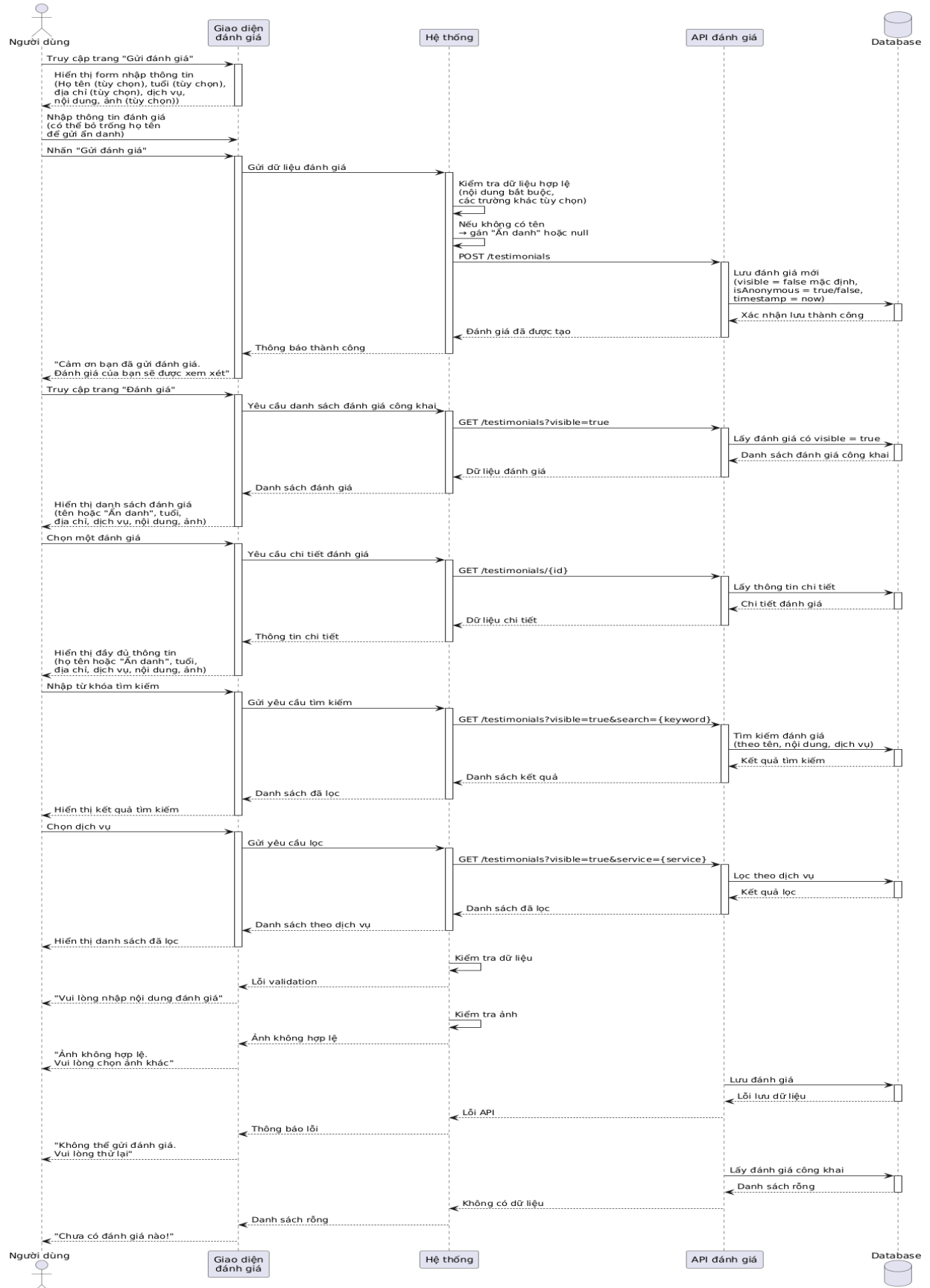
Use Case	Quản lý đánh giá
Tác nhân	Người dùng: Gửi đánh giá Admin: Xem, duyệt, ẩn/hiện, xóa
Mô tả tóm tắt	Người dùng gửi đánh giá; Admin quản lý, duyệt và kiểm soát hiển thị.
Tiền điều kiện	Người dùng: Có thể cần hoặc không cần đăng nhập; nhập đủ thông tin. Admin: Đăng nhập với role = admin.
Hậu điều kiện	Đánh giá được lưu; Admin quản lý; User chỉ thấy đánh giá visible = true.
Luồng sự kiện chính	UC1 – Gửi đánh giá UC2 – Admin xem danh sách UC3 – Admin xem chi tiết UC4 – Admin ẩn/hiện UC5 – Admin xóa UC6 – User xem đánh giá công khai
Luồng thay thế	UC1A – Thiếu thông tin UC1B – Lỗi gửi đánh giá UC4A – Lỗi cập nhật trạng thái
Ràng buộc hệ thống	User không được chỉnh sửa/xóa sau khi gửi. Chỉ Admin có quyền duyệt/ẩn/hiện/xóa. Đánh giá phải có timestamp.

- Biểu đồ tuần tự quản lý đánh giá



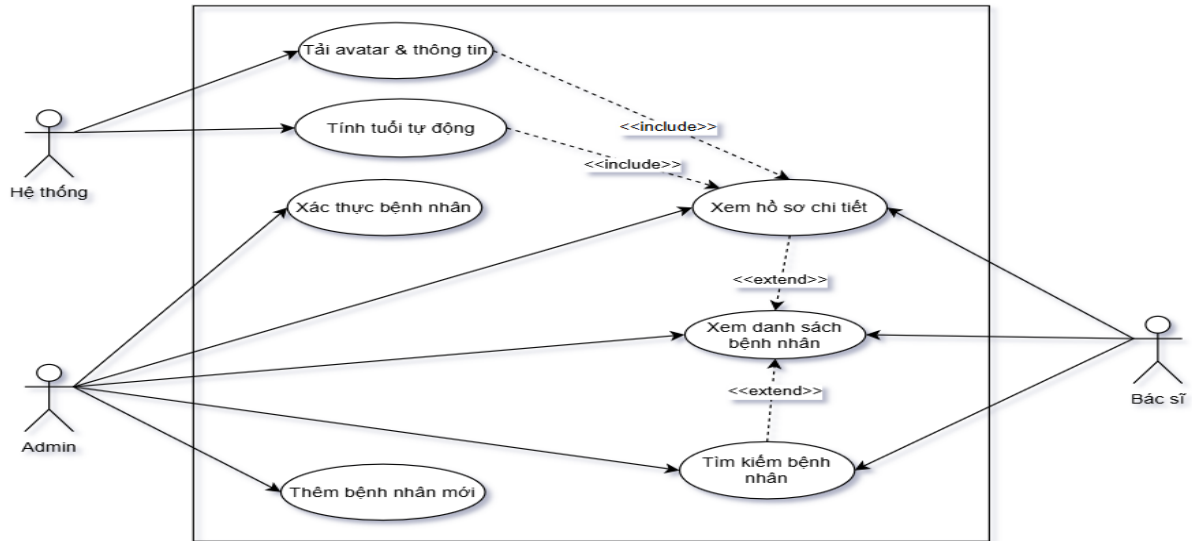
Hình 3.22 Biểu đồ tuần tự quản lý đánh giá cho admin

Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI



Hình 3.23 Biểu đồ tuần tự quản lý đánh giá cho người dùng

3.3.9. Chức năng quản lý bệnh nhân



Hình 3.24 Biểu đồ Use Case quản lý bệnh nhân

- **Mô tả Use Case Quản lý bệnh nhân**

Bảng 3.9 Mô tả Use Case Quản lý bệnh nhân

Use Case	Quản lý bệnh nhân
Tác nhân	Admin, Bác sĩ, Hệ thống
Mô tả tóm tắt	Admin và bác sĩ có thể xem, tìm kiếm, phân trang, xác thực, xem hồ sơ, tạo mới và quản lý thông tin bệnh nhân trong MedicalHope.
Tiền điều kiện	- Tác nhân đã đăng nhập. - Quyền truy cập trang bệnh nhân: • Admin: toàn quyền (xem, thêm, xác thực). • Bác sĩ: xem danh sách & hồ sơ. - Dữ liệu bệnh nhân tồn tại (nếu có).
Hậu điều kiện	- Thông tin bệnh nhân được cập nhật/thêm mới/xác thực. - Danh sách hiển thị đúng theo thay đổi.
Luồng sự kiện chính	Luồng 1: Xem danh sách Luồng 2: Thêm bệnh nhân (Admin) Luồng 3: Xác thực bệnh nhân (Admin) Luồng 4: Xem hồ sơ (Bác sĩ) Luồng 5: Tải avatar & thông tin người dùng
Luồng thay thế	A1: Không có bệnh nhân. → “Không có”. A2: Ngày sinh sai. → “Ngày sinh không hợp lệ”. A3: Lỗi tải danh sách. → “Không thể lấy danh sách”
Yêu cầu đặc biệt	- Tính tuổi tự động. - Hợp nhất dữ liệu profile + bệnh án. - Bệnh nhân không có quyền truy cập.

• **Biểu đồ tuần tự quản lý bệnh nhân**



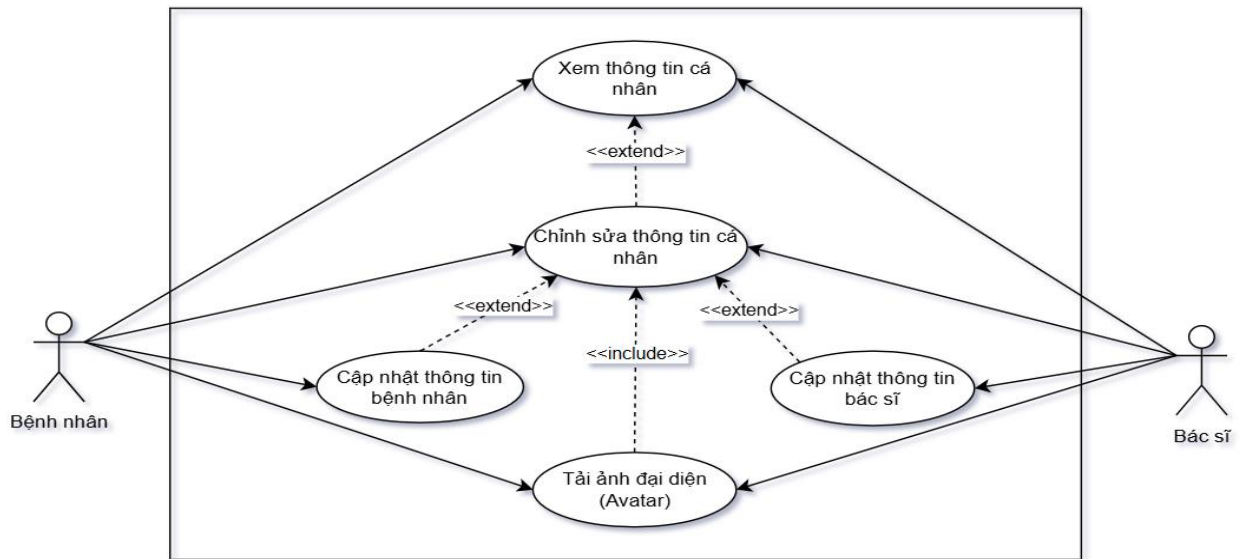
Hình 3.25 Biểu đồ tuần tự quản lý bệnh nhân cho admin

Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI



Hình 3.26 Biểu đồ tuần tự quản lý bệnh nhân cho bác sĩ

3.3.10. Chức năng quản lý thông tin cá nhân



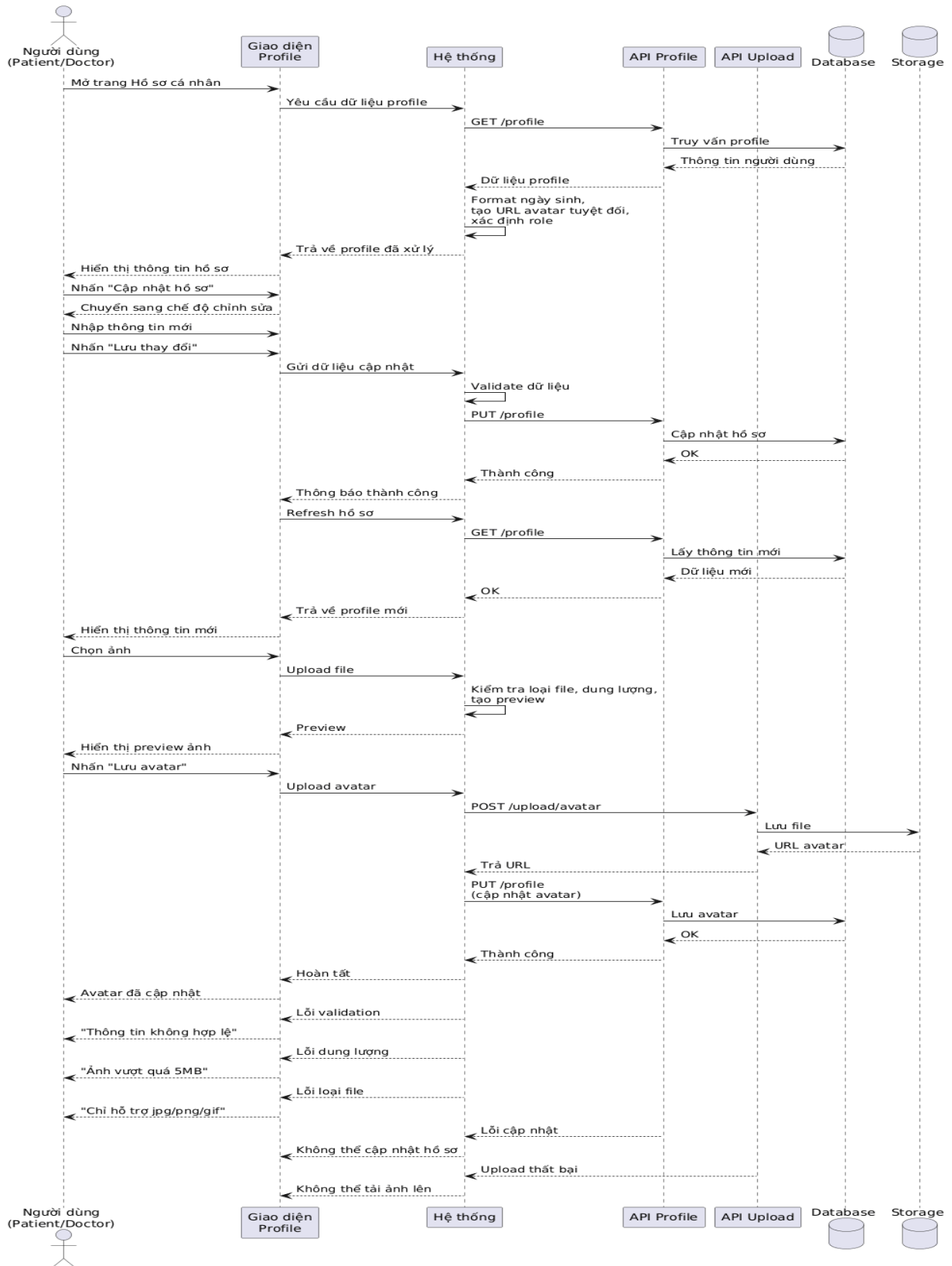
Hình 3.27 Biểu đồ Use Case quản lý thông tin cá nhân

- **Mô tả Use Case Quản lý thông tin cá nhân**

Bảng 3.10 Mô tả Use Case Quản lý thông tin cá nhân

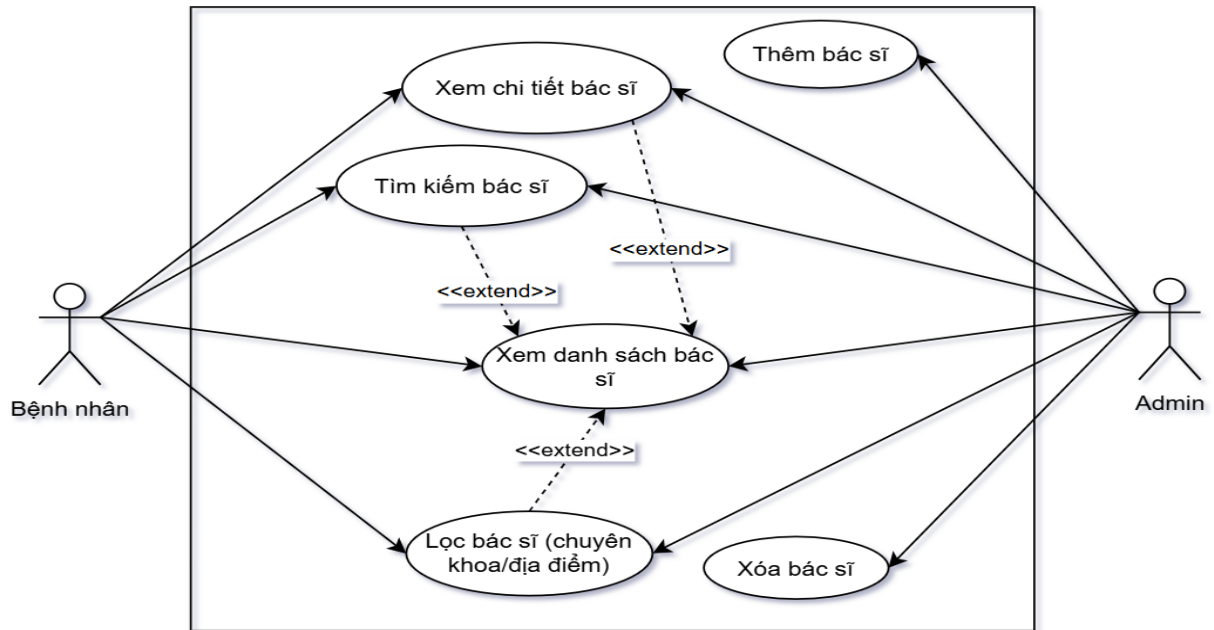
Use Case	Quản lý thông tin cá nhân
Tác nhân	Bệnh nhân (Patient), Bác sĩ (Doctor)
Mô tả tóm tắt	Hệ thống cho phép bệnh nhân và bác sĩ xem và cập nhật thông tin cá nhân như: họ tên, số điện thoại, ngày sinh, giới tính, địa chỉ, nghề nghiệp. Riêng bệnh nhân được cập nhật thêm: bảo hiểm y tế, tình trạng bệnh.
Tiền điều kiện	- Người dùng đã đăng nhập. - Tài khoản thuộc role patient hoặc doctor. - Hồ sơ tồn tại hoặc ở trạng thái trống để cập nhật.
Hậu điều kiện	- Thông tin được cập nhật & lưu vào hệ thống. - Avatar được cập nhật nếu có tải lên.
Luồng sự kiện chính	Luồng 1: Xem thông tin cá nhân Luồng 2: Chỉnh sửa thông tin cá nhân Luồng 3: Tải avatar Luồng 4: Thông tin riêng của bệnh nhân
Luồng thay thế	A1: Lỗi dữ liệu → báo lỗi: ngày sinh sai, avatar >5MB, sai định dạng. A2: Upload avatar lỗi → báo lỗi không thể tải ảnh. A3: Không thể lưu hồ sơ → “Không thể cập nhật hồ sơ”.
Yêu cầu đặc biệt	- Avatar preview trước khi lưu. - Bác sĩ không chỉnh sửa được: BHYT, tình trạng bệnh. - Tự động chuẩn hóa URL avatar. - Format ngày sinh: yyyy-MM-dd → dd/MM/yyyy.

• Biểu đồ tuần tự quản lý thông tin cá nhân



Hình 3.28 Biểu đồ tuần tự quản lý hồ sơ cá nhân

3.3.11. Chức năng quản lý bác sĩ



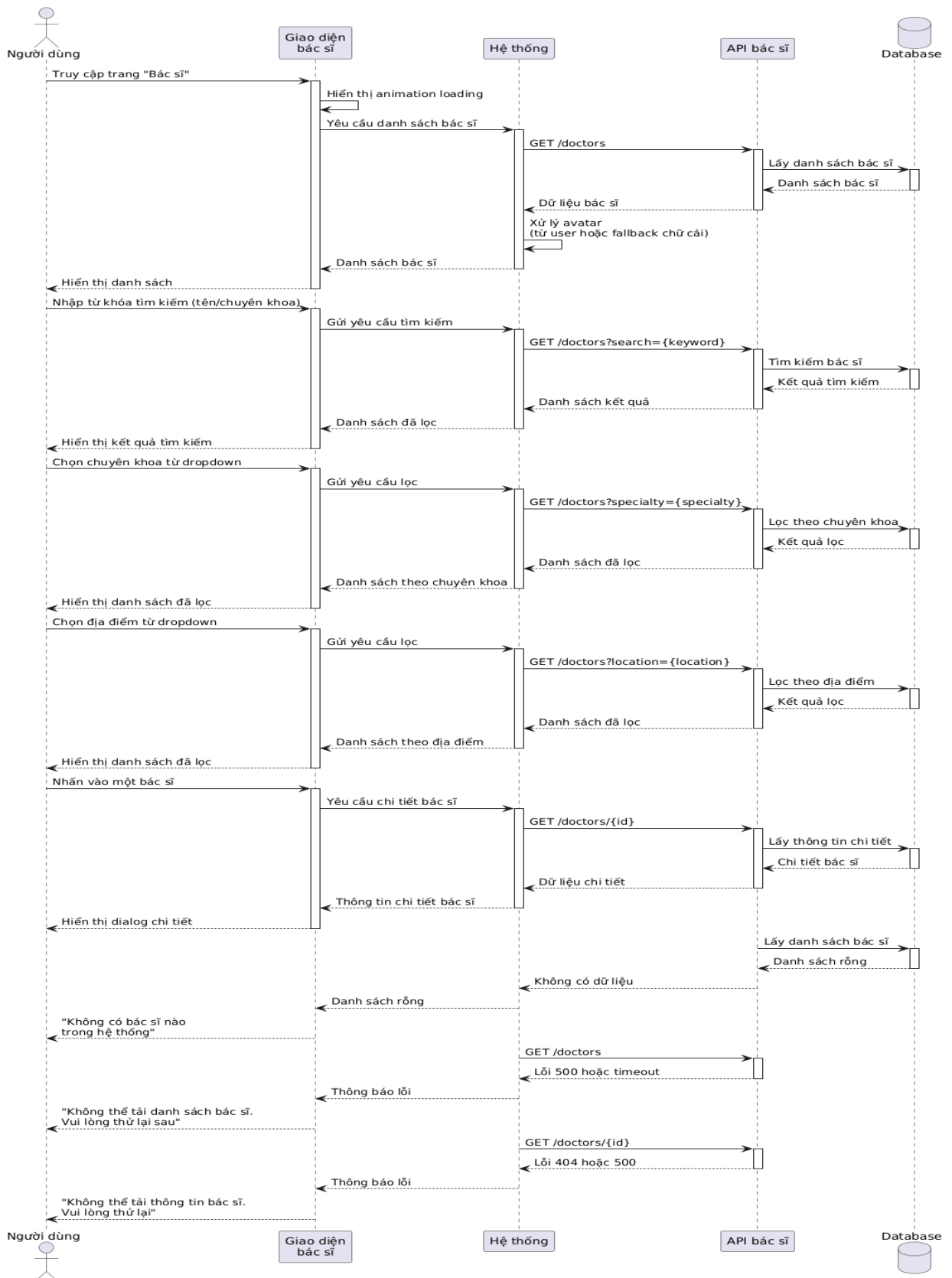
Hình 3.29 Biểu đồ Use Case quản lý bác sĩ

- **Mô tả Use Case Quản lý bác sĩ**

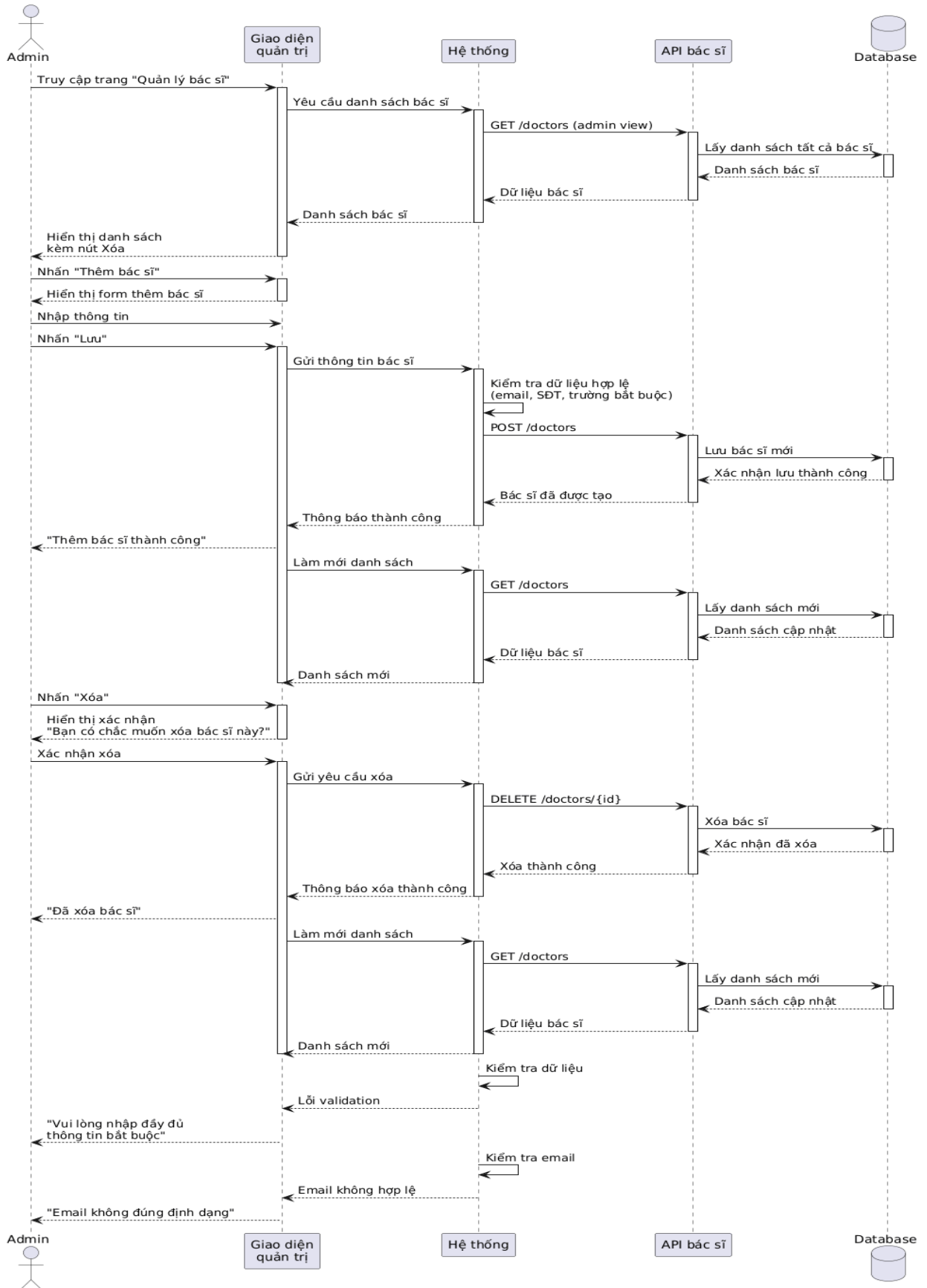
Bảng 3.11 Mô tả Use Case Quản lý bác sĩ

Use Case	Quản lý bác sĩ
Tác nhân	• Admin • Người dùng
Mô tả tóm tắt	Admin có thể quản lý danh sách bác sĩ. Người dùng có thể xem danh sách và xem chi tiết bác sĩ.
Tiền điều kiện	- Người dùng đã đăng nhập (với vai trò Admin nếu quản lý) - Backend có dữ liệu bác sĩ hoặc trống
Hậu điều kiện	- Danh sách bác sĩ được cập nhật - Người dùng xem được thông tin bác sĩ chính xác
Luồng sự kiện chính	1. Hệ thống tải danh sách bác sĩ 2. Hiển thị avatar, chuyên khoa, địa điểm, số năm kinh nghiệm 3. Người dùng nhấn vào bác sĩ để xem chi tiết 4. Admin có thể CRUD qua trang quản trị
Luồng thay thế	A1: Không có bác sĩ → Hiển thị thông báo A2: Lỗi tải danh sách → Thông báo “Không thể tải danh sách bác sĩ” A3: Lỗi tải chi tiết → Hiển thị thông báo lỗi
Yêu cầu đặc biệt	- Avatar tự động lấy từ user hoặc fallback bằng chữ cái - Animation khi load danh sách - Dialog hiển thị chi tiết bác sĩ

- **Biểu đồ tuần tự quản lý bác sĩ**

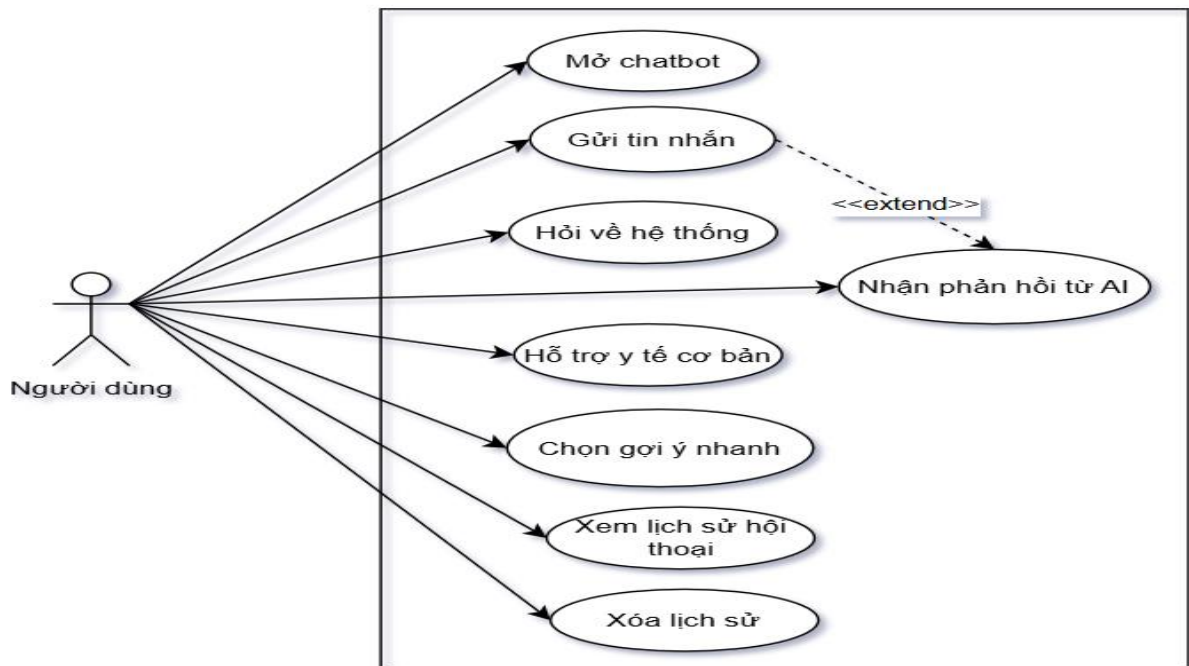


Hình 3.30 Biểu đồ tuần tự quản lý bác sĩ cho bệnh nhân



Hình 3.31 Biểu đồ tuần tự quản lý bác sĩ cho admin

3.3.12. Chức năng Chatbot AI



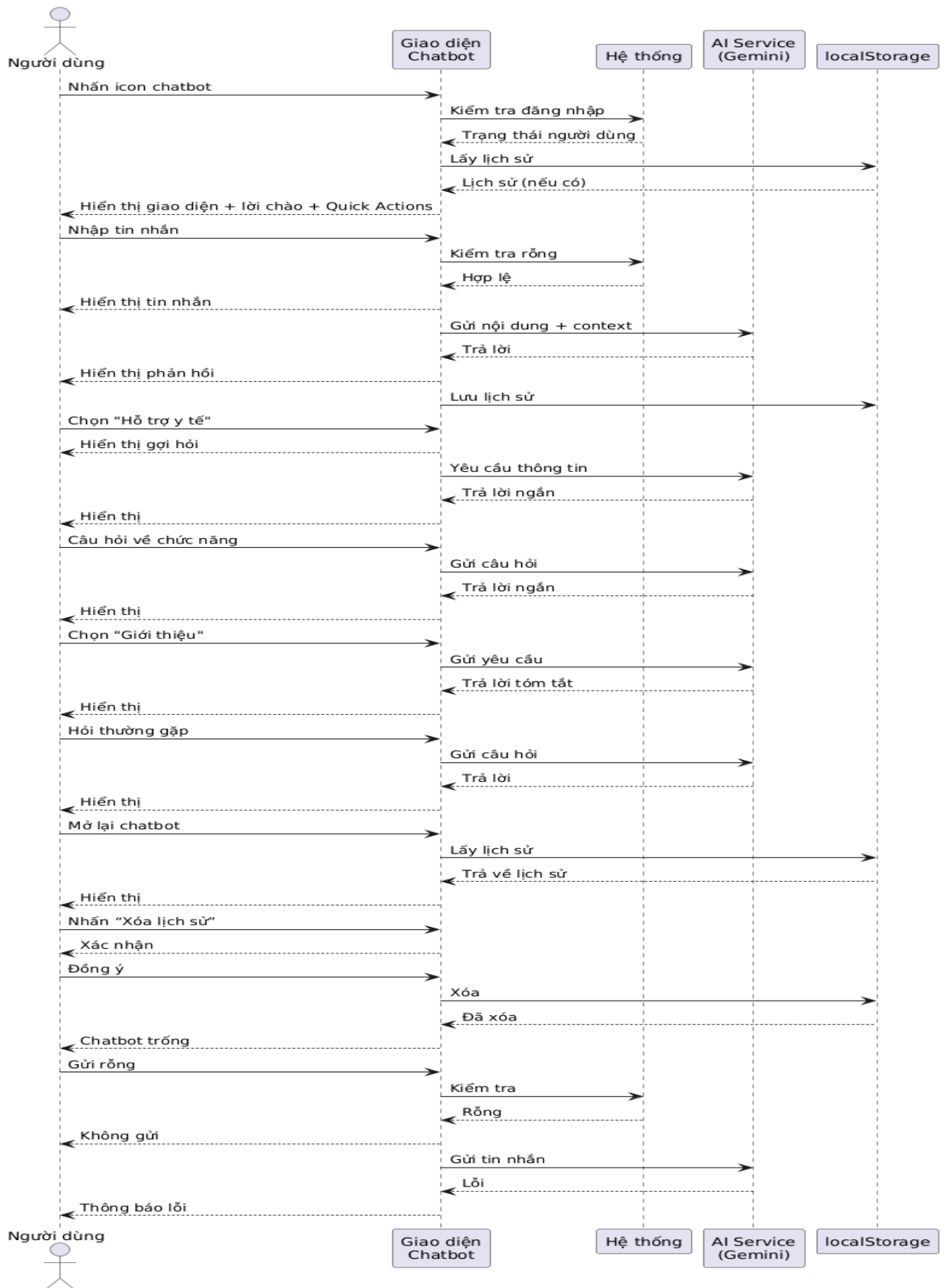
Hình 3.32 Biểu đồ Use Case chatbotAI

- **Mô tả Use Case ChatbotAI**

Bảng 3.12 Mô tả Use Case ChatbotAI

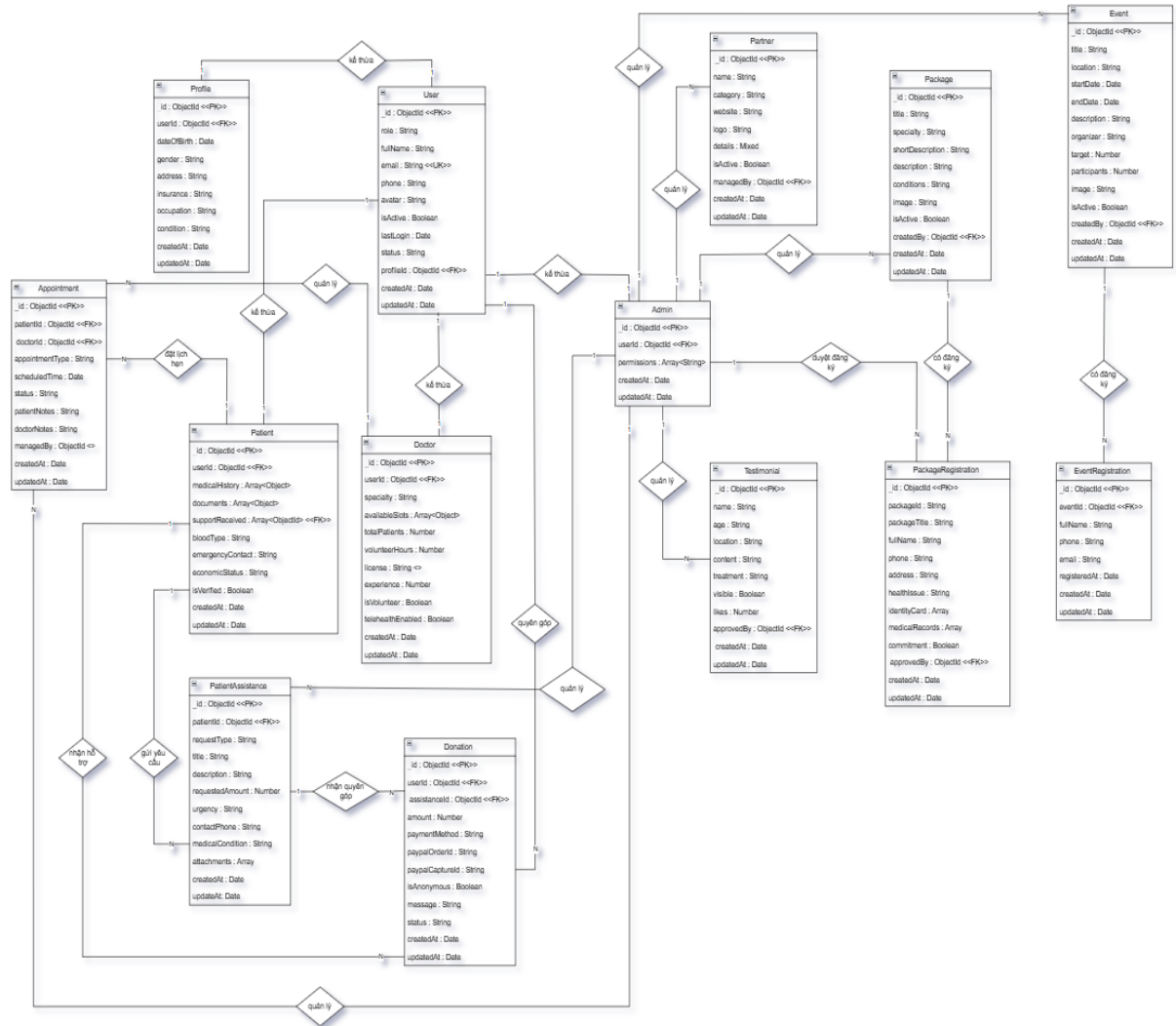
Use Case	Chatbot AI
Tác nhân	Người dùng
Mô tả tóm tắt	Chatbot AI hỗ trợ người dùng và bác sĩ 24/7, cung cấp thông tin, hỗ trợ y tế, hướng dẫn sử dụng hệ thống.
Tiền điều kiện	- Người dùng có thể truy cập chatbot không cần đăng nhập - Nếu đăng nhập → chatbot cá nhân hoá
Hậu điều kiện	- Tin nhắn được lưu vào localStorage theo từng người dùng - Chatbot phản hồi đúng nội dung - Không thực thi lệnh admin
Luồng sự kiện chính	UC1 – Mở chatbot: Hiển thị lời chào tùy loại tài khoản UC2 – Gửi tin nhắn: Người dùng nhập → AI trả lời UC3 – Quick Actions: Người dùng chọn gợi UC4 – Lưu lịch sử: Lưu vào localStorage theo user
Luồng thay thế	UC2A – Lỗi AI: Hiển thị “AI tạm thời không khả dụng” UC2B – Tin nhắn rỗng: Không gửi UC4A – Không có lịch sử: Tạo mới
Yêu cầu đặc biệt	- AI dùng Gemini - Chatbot trả lời tiếng Việt - Hiển thị dạng hội thoại 2 chiều

• **Biểu đồ tuần tự ChatbotAI**



Hình 3.33 Biểu đồ tuần tự chatbotAI

3.4. Thiết kế cơ sở dữ liệu



Hình 3.34 Mô hình quan hệ

3.5. Danh mục các bảng

Bảng 3.13 Mô tả các bảng cơ sở dữ liệu

TT	Tên bảng	Mô tả
1	User	Lưu trữ thông tin tài khoản, phân quyền hệ thống và đăng nhập.
2	Patient	Lưu trữ thông tin bệnh nhân, hồ sơ y tế và trạng thái xác minh.
3	Doctor	Lưu trữ thông tin chuyên môn, kinh nghiệm và lịch rảnh của bác sĩ.
4	Appointment	Lưu trữ thông tin lịch hẹn giữa bệnh nhân và bác sĩ.

5	Donation	Lưu trữ thông tin giao dịch quyên góp và xác nhận thanh toán.
6	Event	Lưu trữ thông tin sự kiện cộng đồng và chiến dịch.
7	Event_Registration	Lưu trữ thông tin người đăng ký tham gia sự kiện.
8	Package	Lưu trữ thông tin gói khám và điều kiện sử dụng.
9	PackageRegistration	Lưu trữ thông tin đăng ký gói khám của bệnh nhân.
10	Partner	Lưu trữ thông tin đối tác hỗ trợ dự án.
11	Patient_Assistance	Lưu trữ thông tin yêu cầu hỗ trợ tài chính của bệnh nhân.
12	Testimonial	Lưu trữ lời chia sẻ, cảm ơn hoặc phản hồi từ bệnh nhân.

Bảng User

Bảng 3.14 Mô tả bảng User

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã người dùng
role	String	Không		patient, doctor, admin
fullName	String	Không		Họ và tên
email	String	Không	Unique	Email duy nhất
phone	String	Không		Số điện thoại
passwordHash	String	Không		Mật khẩu mã hóa
avatar	String	Có		Ảnh đại diện
isActive	Boolean	Có		Tài khoản hoạt động?
lastLogin	Date	Có		Lần đăng nhập cuối
status	String	Có		active, inactive, suspended
profileId	ObjectId	Có	Foreign Key	Tham chiếu Profile
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Profile

Bảng 3.15 Mô tả bảng Profile

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã hồ sơ
userId	ObjectId	Không	Foreign Key	Liên kết User
dateOfBirth	Date	Có		Ngày sinh
gender	String (male, female, other)	Có		Giới tính
address	String	Có		Địa chỉ
insurance	String	Có		Bảo hiểm
occupation	String	Có		Nghề nghiệp
condition	String	Có		Tình trạng sức khỏe
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Patient

Bảng 3.16 Mô tả bảng Patient

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã bệnh nhân
userId	ObjectId	Không	Foreign Key	Liên kết User
medicalHistory	Array<Object>	Có		Bệnh án
documents	Array<Object>	Có		Tài liệu
supportReceived	Array<ObjectId>	Có	Foreign Key	Tham chiếu Donation
bloodType	String	Có		Nhóm máu
emergencyContact	String	Có		SĐT khẩn cấp
economicStatus	String	Có		very_poor, poor, middle, good
isVerified	Boolean	Có		Xác minh tài khoản
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Doctor

Bảng 3.17 Mô tả bảng Doctor

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã bác sĩ
userId	ObjectId	Không	Foreign Key	Tham chiếu User
specialty	String	Không		Chuyên khoa
availableSlots	Array<Object>	Có		Lịch rảnh
totalPatients	Number	Có		Tổng bệnh nhân
volunteerHours	Number	Có		Giờ khám thiện nguyện
license	String	Không	Unique	Chứng chỉ hành nghề
experience	Number	Không		Số năm kinh nghiệm
isVolunteer	Boolean	Có		Bác sĩ thiện nguyện?
telehealthEnabled	Boolean	Có		Cho phép khám online
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Package Registration

Bảng 3.18 Mô tả bảng Package Registration

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã đăng ký
packageId	String	Không		ID gói
packageTitle	String	Không		Tên gói
fullName	String	Không		Tên người đăng ký
phone	String	Không		SĐT
address	String	Không		Địa chỉ
healthIssue	String	Không		Tình trạng bệnh
identityCard	Array	Không		URL CMND/CCCD
medicalRecords	Array	Có		Hồ sơ
commitment	Boolean	Có		Mặc định: true
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Donation

Bảng 3.19 Mô tả bảng Donation

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã giao dịch
userId	ObjectId	Có	Foreign Key	Người quyên góp
assistanceId	ObjectId	Không	Foreign Key	Yêu cầu hỗ trợ
amount	Number	Không		Số tiền
paymentMethod	String	Không		vnpay, paypal
paypalOrderId	String	Có		
paypalCaptureId	String	Có		
isAnonymous	Boolean	Có		Ẩn danh?
donorName	String	Có		Tên người tặng
donorEmail	String	Có		Email
donorPhone	String	Có		SĐT
message	String	Có		Lời nhắn
status	String	Không		pending, completed...
confirmedAt	Date	Có		Ngày xác nhận
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Event

Bảng 3.20 Mô tả bảng Event

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã sự kiện
title	String	Không		Tiêu đề
location	String	Không		Địa điểm
startDate	Date	Không		Ngày bắt đầu
endDate	Date	Có		Ngày kết thúc
description	String	Không		Mô tả
organizer	String	Không		Đơn vị tổ chức
target	Number	Không		Mục tiêu
participants	Number	Có		Số người
image	String	Có		Ảnh
isActive	Boolean	Có		Hoạt động?
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Event Registration

Bảng 3.21 Mô tả bảng Event Registration

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã đăng ký
eventId	ObjectId	Không	Foreign Key	Tham chiếu Event
fullName	String	Không		Họ tên
phone	String	Không		SĐT
email	String	Có		Email
registeredAt	Date	Có		Ngày đăng ký
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Package

Bảng 3.22 Mô tả bảng Package

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã gói
title	String	Không		Tên gói
specialty	String	Không		Chuyên khoa
shortDescription	String	Không		Mô tả ngắn
description	String	Không		Mô tả chi tiết
conditions	String	Không		Điều kiện
image	String	Có		Ảnh
isActive	Boolean	Có		Hoạt động
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Appointment

Bảng 3.23 Mô tả bảng Appointment

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã lịch hẹn
patientId	ObjectId	Không	Foreign Key	Tham chiếu Patient
doctorId	ObjectId	Không	Foreign Key	Tham chiếu Doctor
appointmentType	String	Không		consultation...
scheduledTime	Date	Không		Thời gian hẹn
status	String	Không		scheduled, confirmed...
patientNotes	String	Có		Ghi chú
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập nhật

Bảng Testimonial

Bảng 3.24 Mô tả bảng Testimonial

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã testimonial
name	String	Không		Tên người gửi
age	String	Có		Tuổi
location	String	Có		Địa điểm
content	String	Không		Nội dung
treatment	String	Có		Loại điều trị
visible	Boolean	Có		Hiển thị?
likes	Number	Có		Mặc định: 0
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập

Bảng Partner

Bảng 3.25 Mô tả bảng Partner

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã đối tác
name	String	Không		Tên
type	String	Không		transportation...
category	String	Không		Nhóm phân loại
website	String	Có		Web
logo	String	Có		Logo
details	Mixed	Có		Tùy loại
isActive	Boolean	Có		Hoạt động
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập

Bảng Patient Assistance

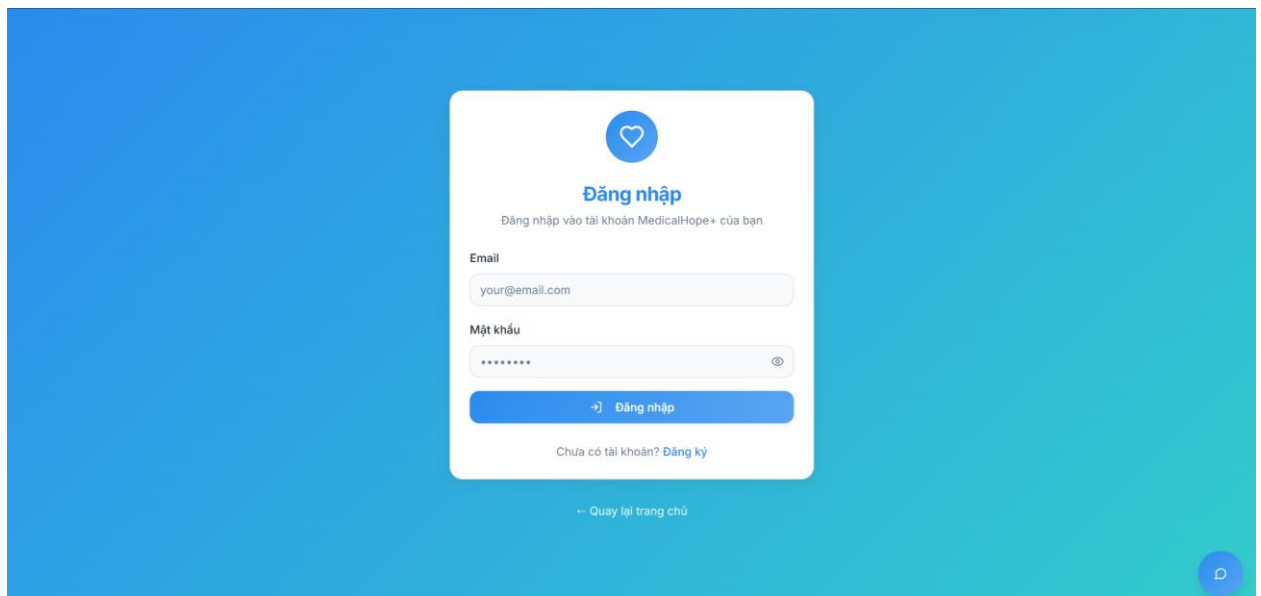
Bảng 3.26 Mô tả bảng Patient Assistance

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
_id	ObjectId	Không	Primary Key	Mã yêu cầu
patientId	ObjectId	Không	Foreign Key	Bệnh nhân
requestType	String	Không		Loại yêu cầu
title	String	Không		Tiêu đề
description	String	Không		Mô tả
medicalCondition	String	Không		Tình trạng
requestedAmount	Number	Không		Số tiền cần
urgency	String	Không		Mức độ

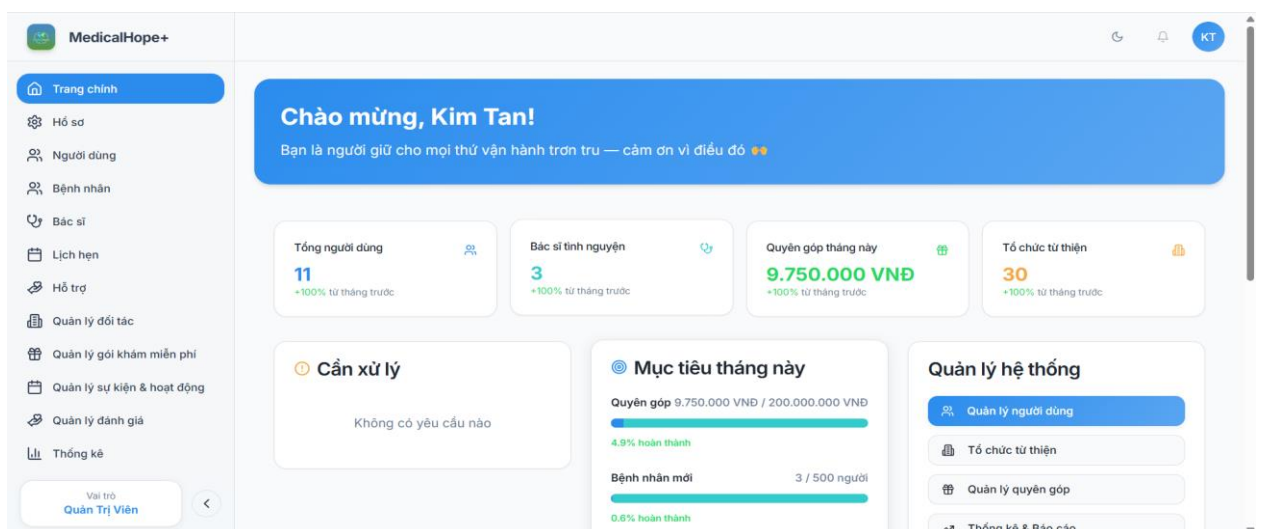
contactPhone	String	Không		SĐT
raisedAmount	Number	Có		Đã quyên góp
remainingAmount	Number	Có		Còn lại
attachments	Array	Có		File
donationIds	Array	Có		Danh sách donation
status	String	Có		pending → completed
approvedBy	ObjectId	Có		User duyệt
createdAt	Date	Không		Ngày tạo
updatedAt	Date	Không		Ngày cập

3.6. Thiết kế giao diện

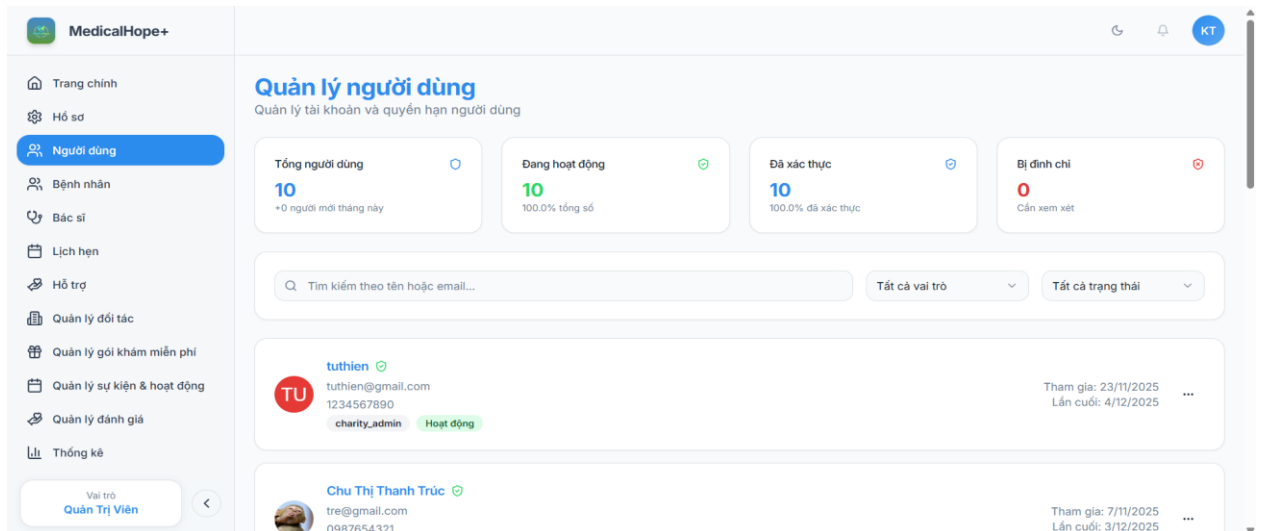
3.6.1. Giao diện Admin



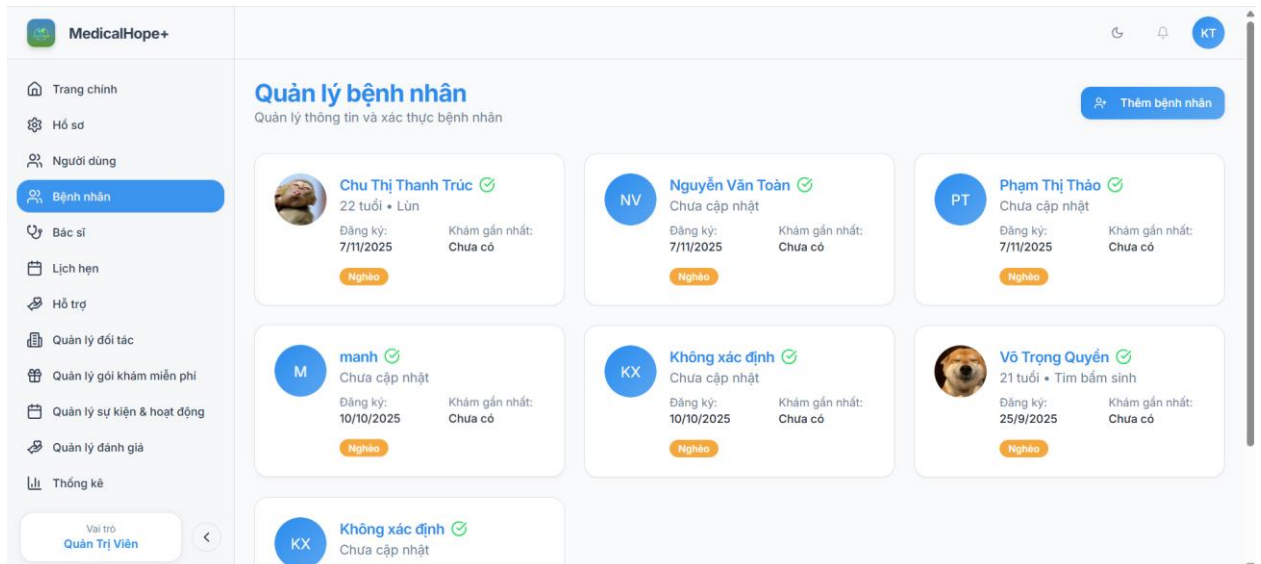
Hình 3.35 Giao diện đăng nhập



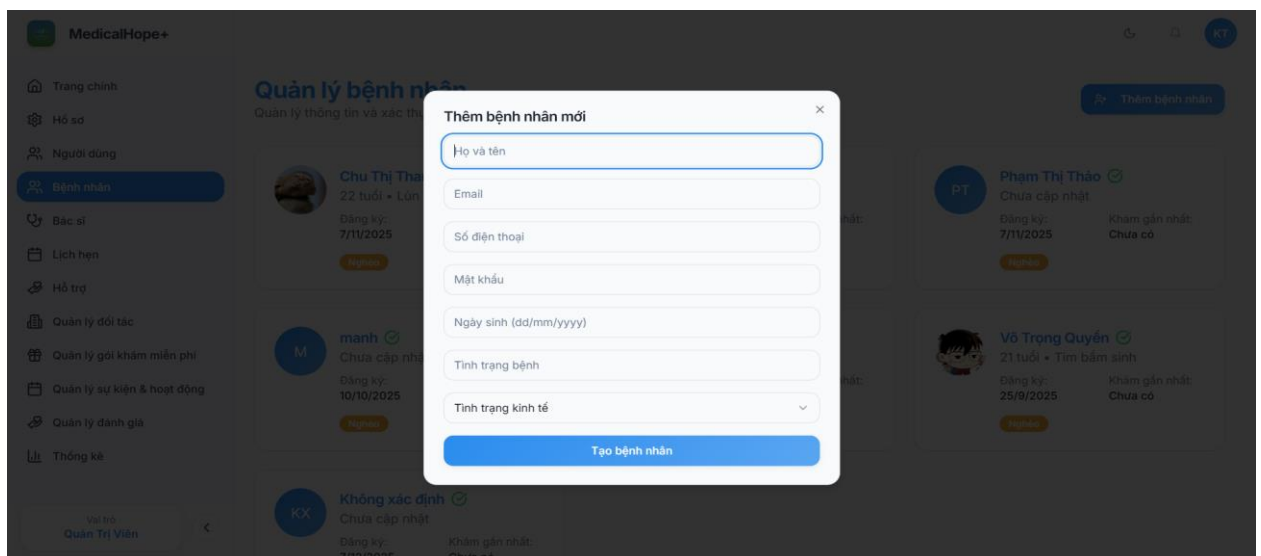
Hình 3.36 Giao diện chính Admin



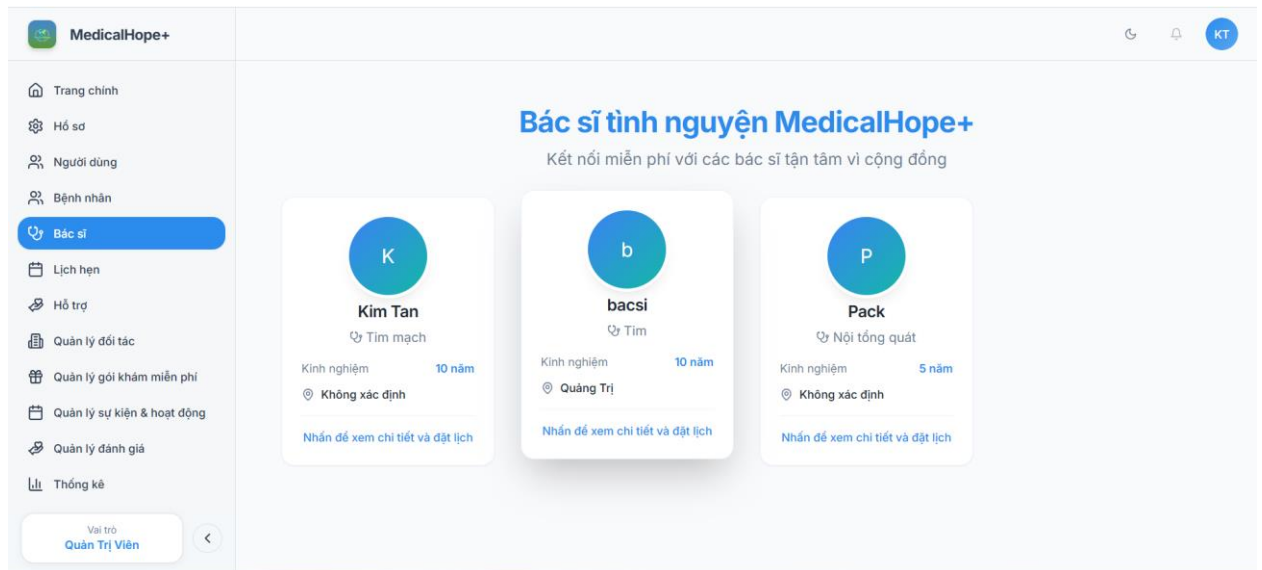
Hình 3.37 Giao diện quản lý người dùng



Hình 3.38 Giao diện quản lý bệnh nhân



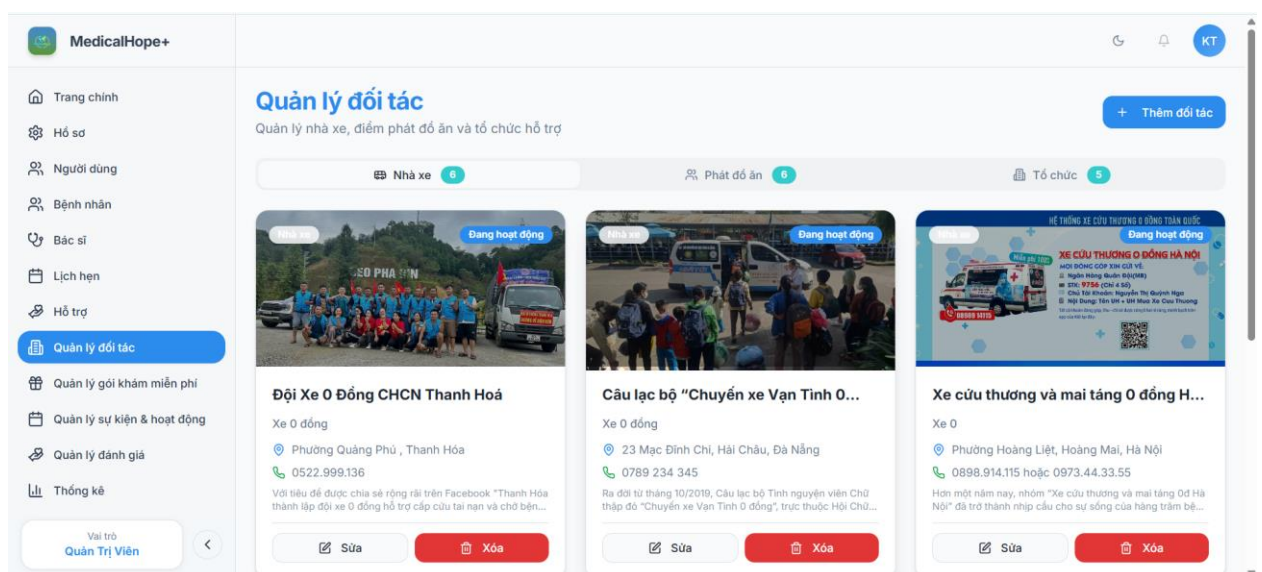
Hình 3.39 Giao diện thêm bệnh nhân



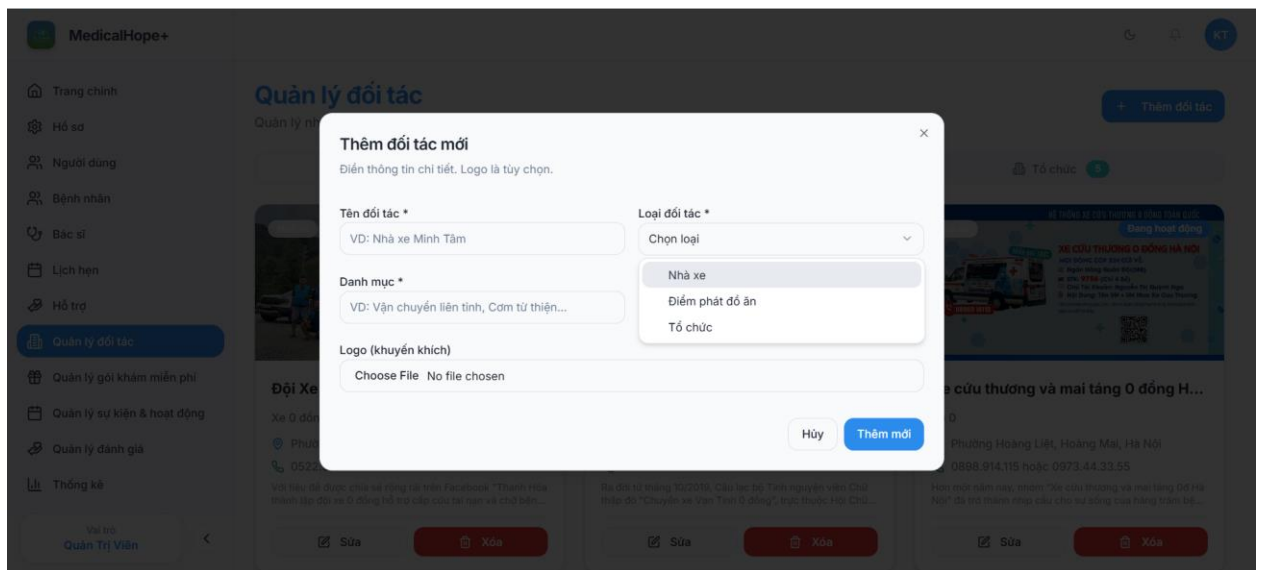
Hình 3.40 Giao diện quản lý bác sĩ



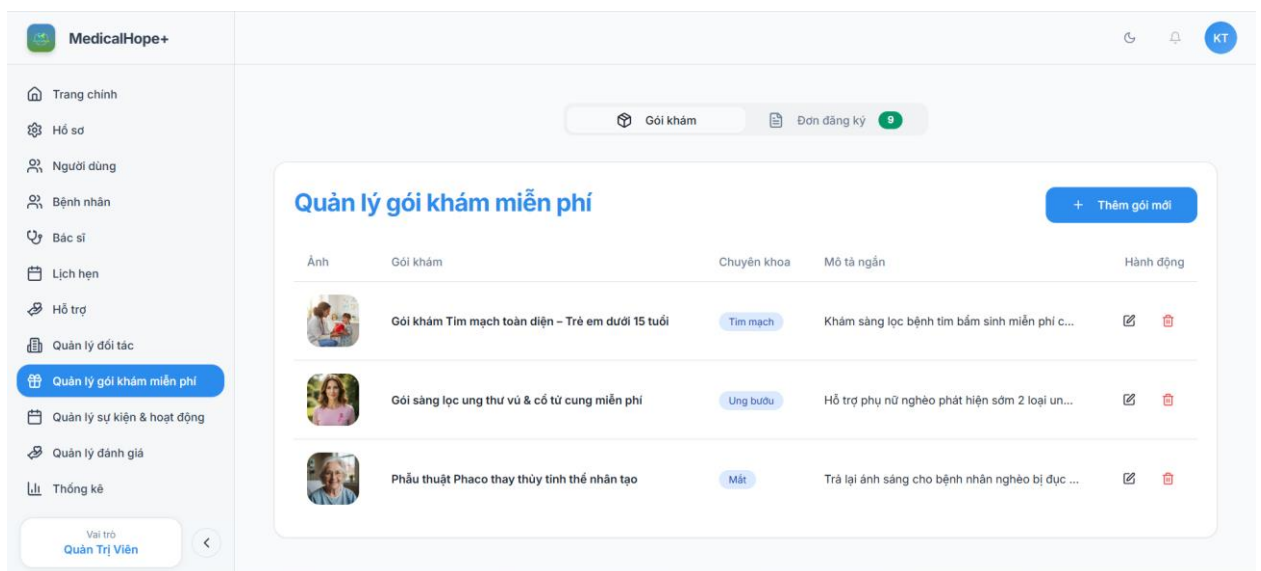
Hình 3.41 Giao diện quản lý hỗ trợ



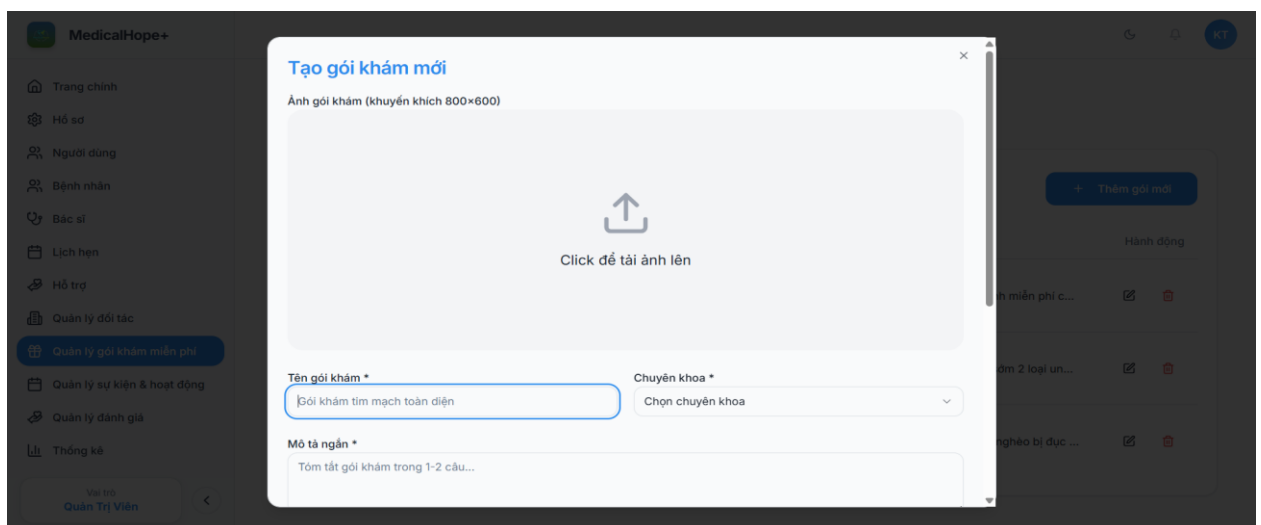
Hình 3.42 Giao diện quản lý đối tác



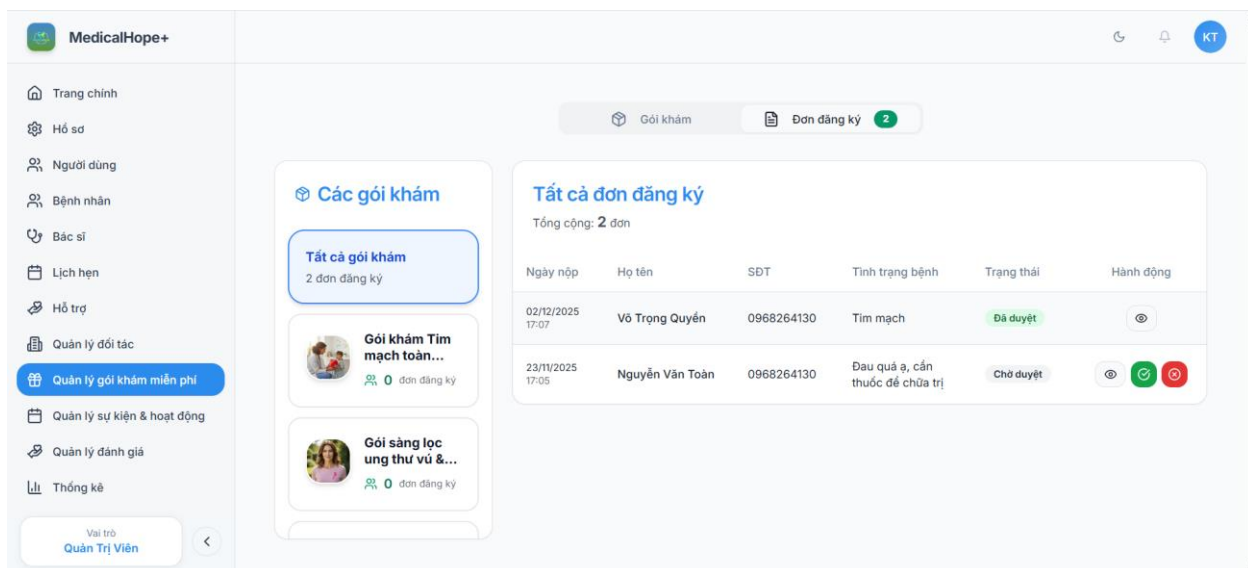
Hình 3.43 Giao diện thêm đối tác



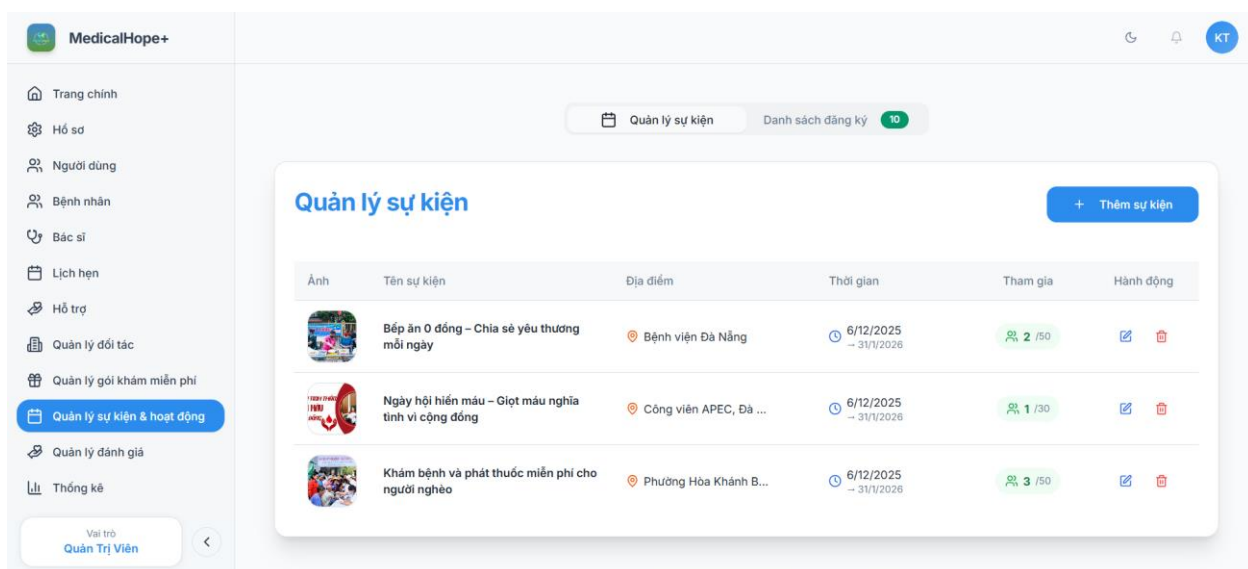
Hình 3.44 Giao diện quản lý gói khám



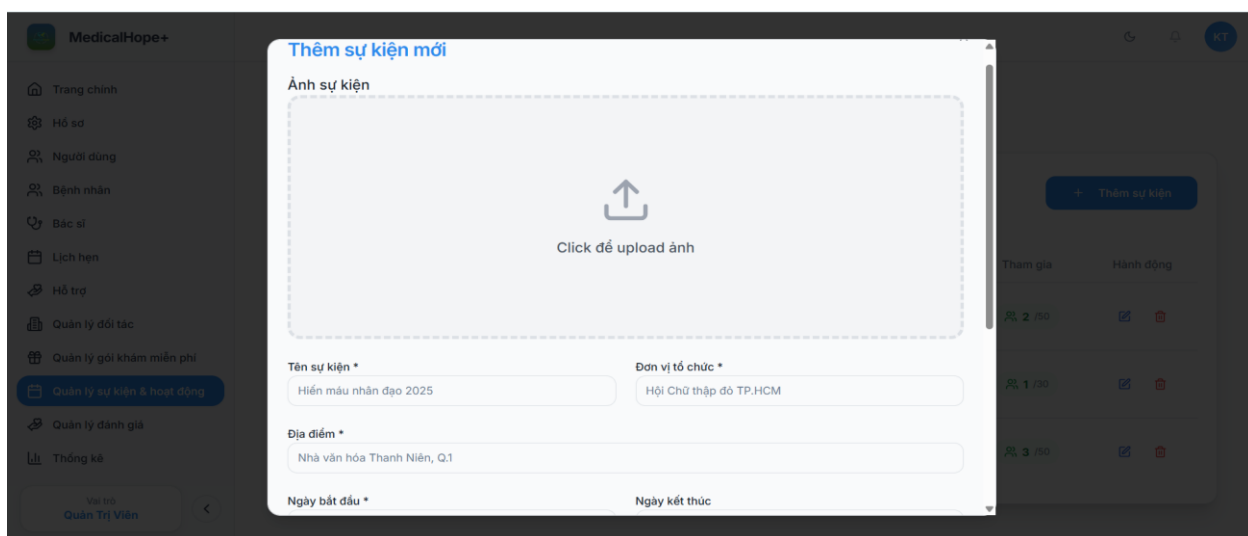
Hình 3.45 Giao diện tạo gói khám



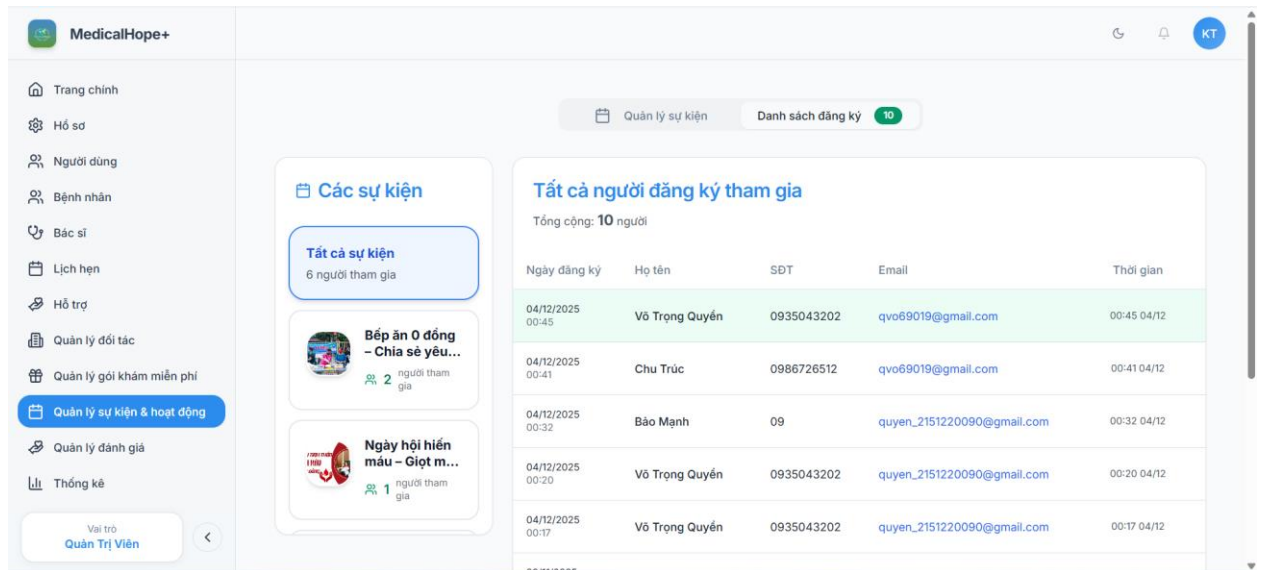
Hình 3.46 Giao diện đơn đăng ký gói khám



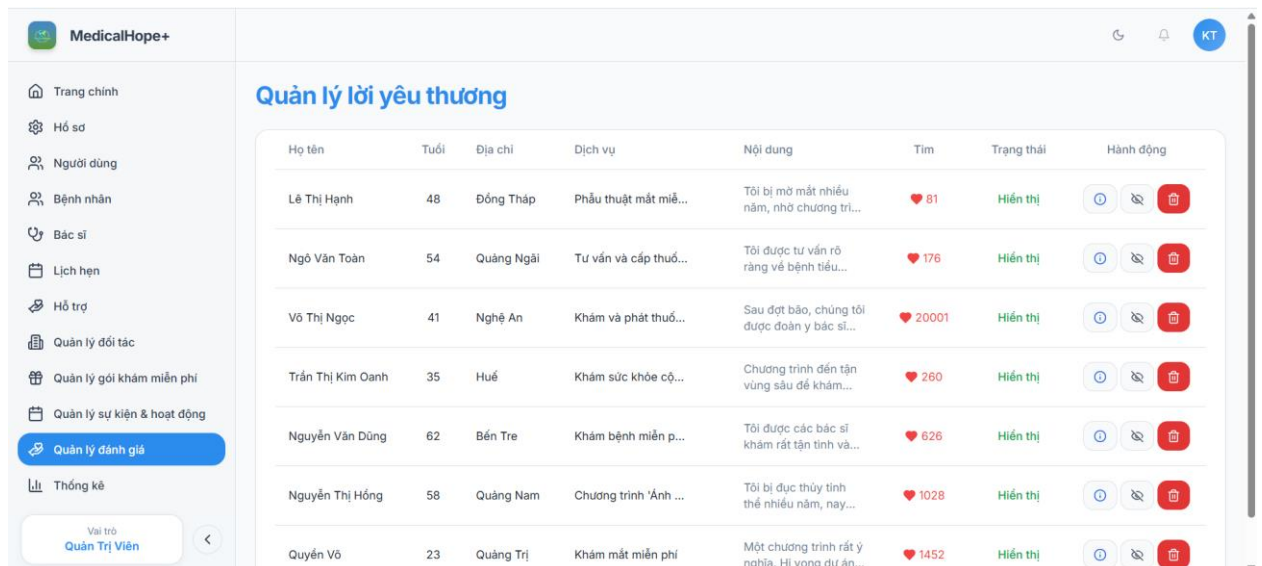
Hình 3.47 Giao diện quản lý sự kiện



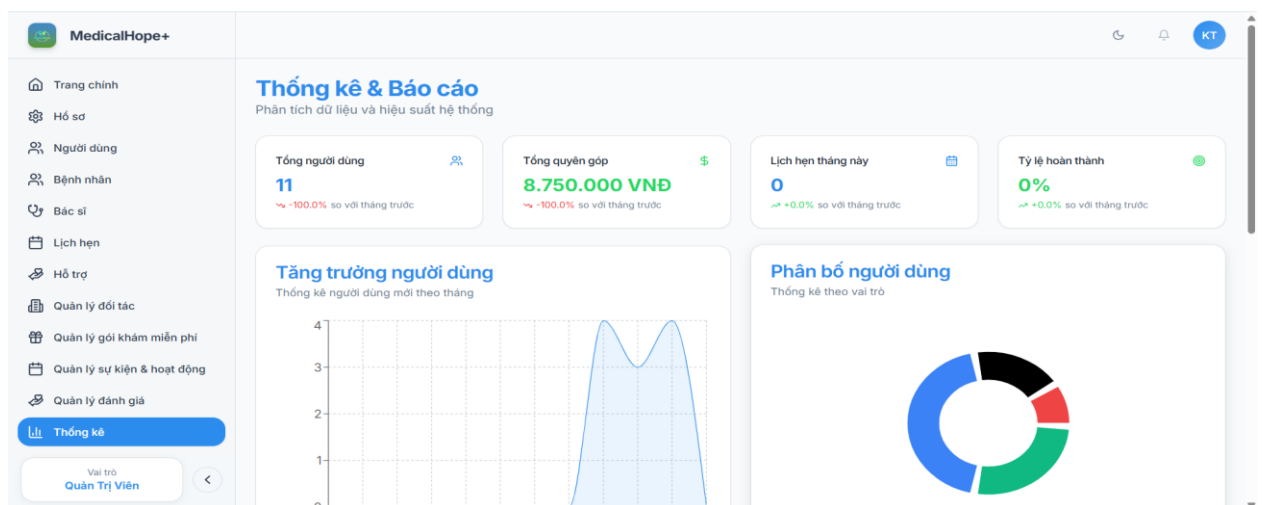
Hình 3.48 Giao diện thêm sự kiện



Hình 3.49 Giao diện danh sách đơn đăng ký tham gia sự kiện

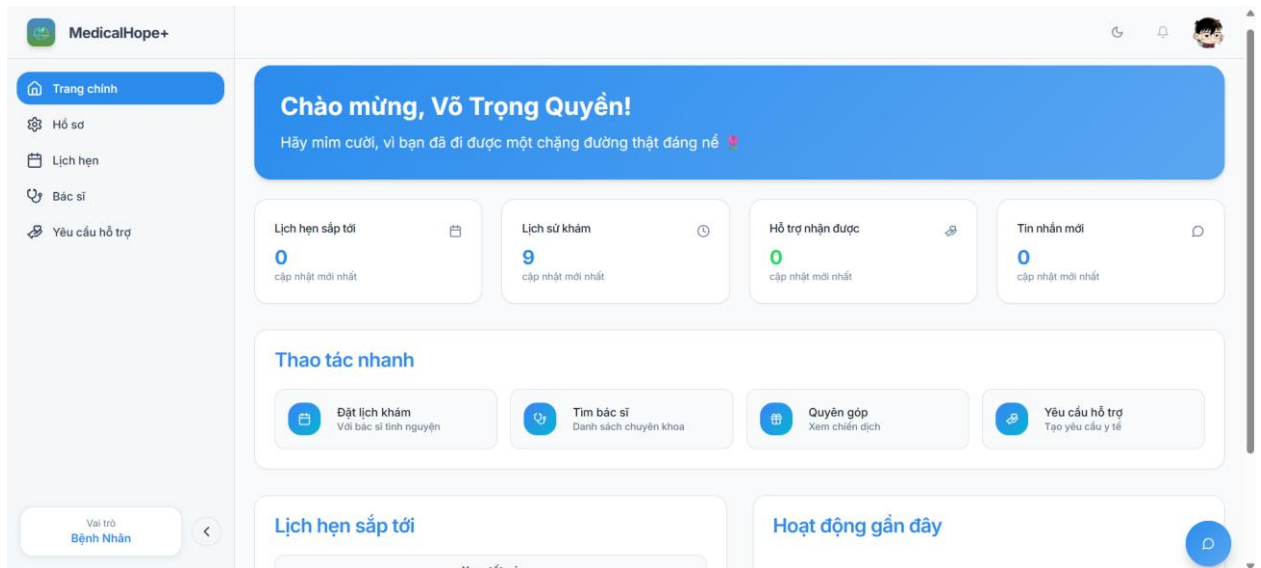


Hình 3.50 Giao diện quản lý lời yêu thương

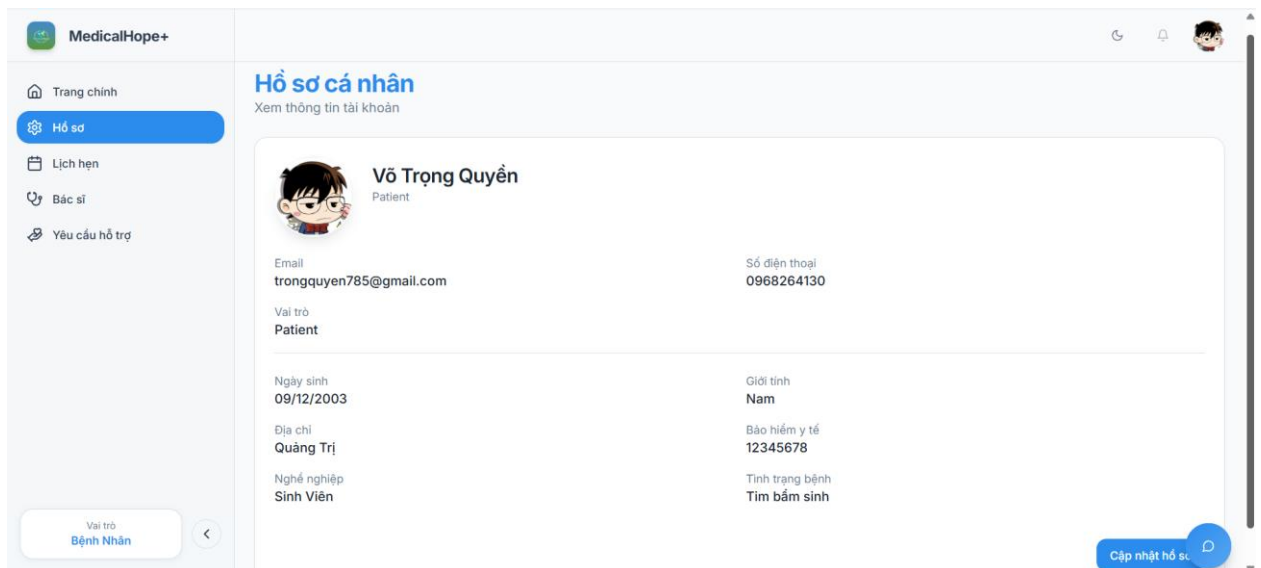


Hình 3.51 Giao diện Thống kê báo cáo

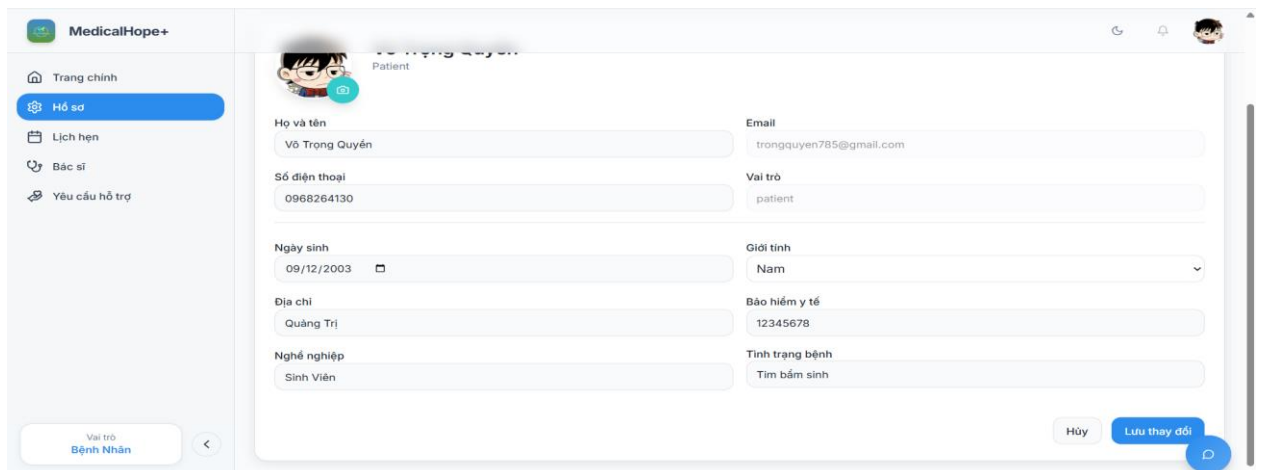
3.6.2. Giao diện Bệnh nhân



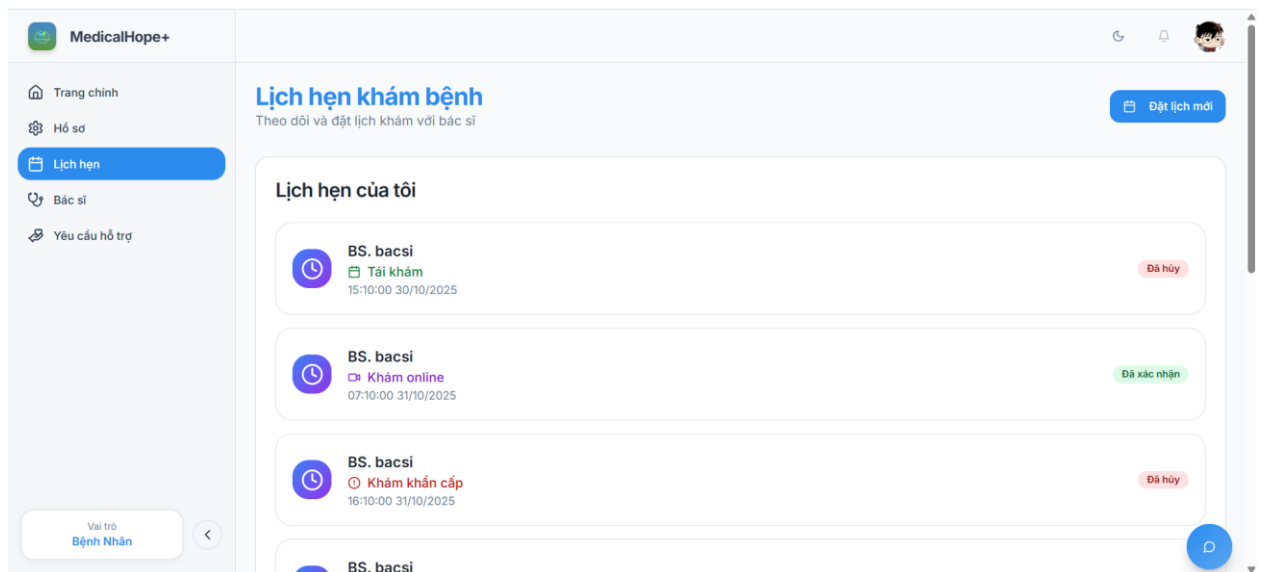
Hình 3.52 Giao diện chính của bệnh nhân



Hình 3.53 Giao diện quản lý thông tin cá nhân



Hình 3.54 Giao diện cập nhật thông tin cá nhân



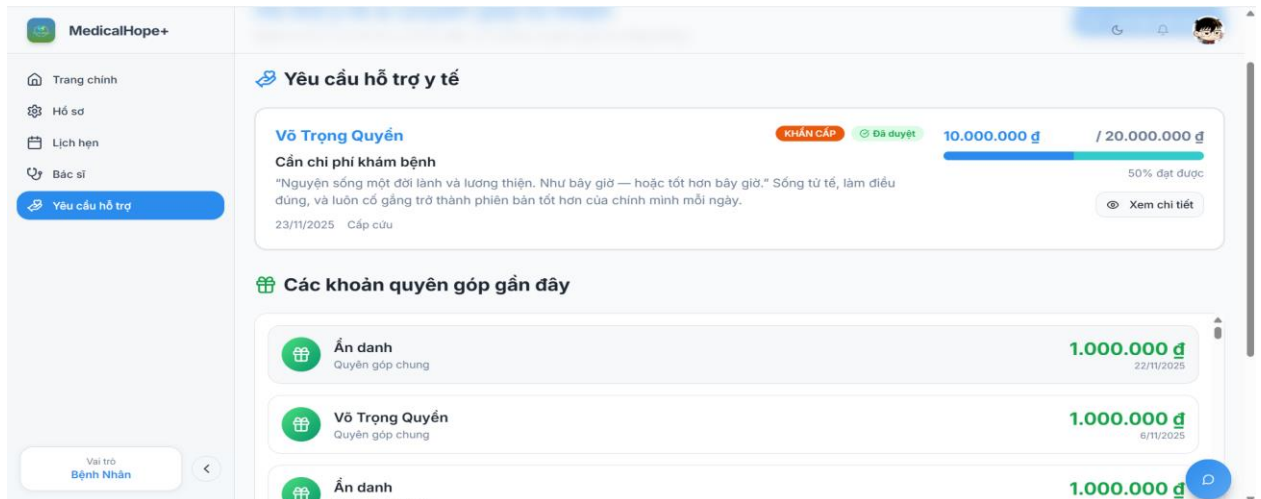
Hình 3.55 Giao diện danh sách lịch hẹn



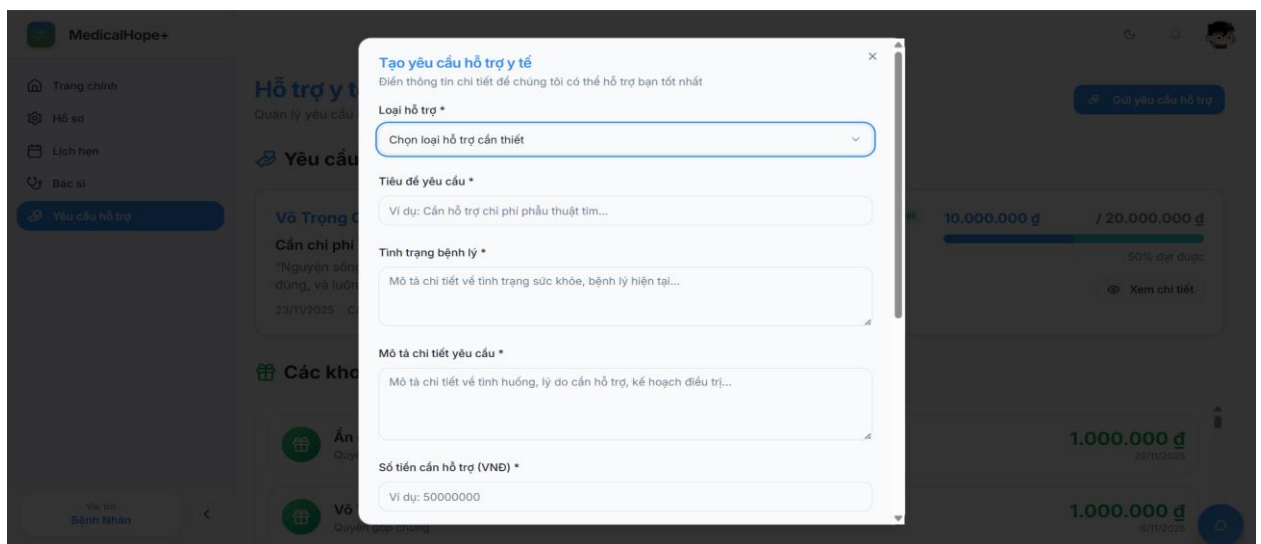
Hình 3.56 Giao diện đặt lịch khám



Hình 3.57 Giao diện danh sách bác sĩ

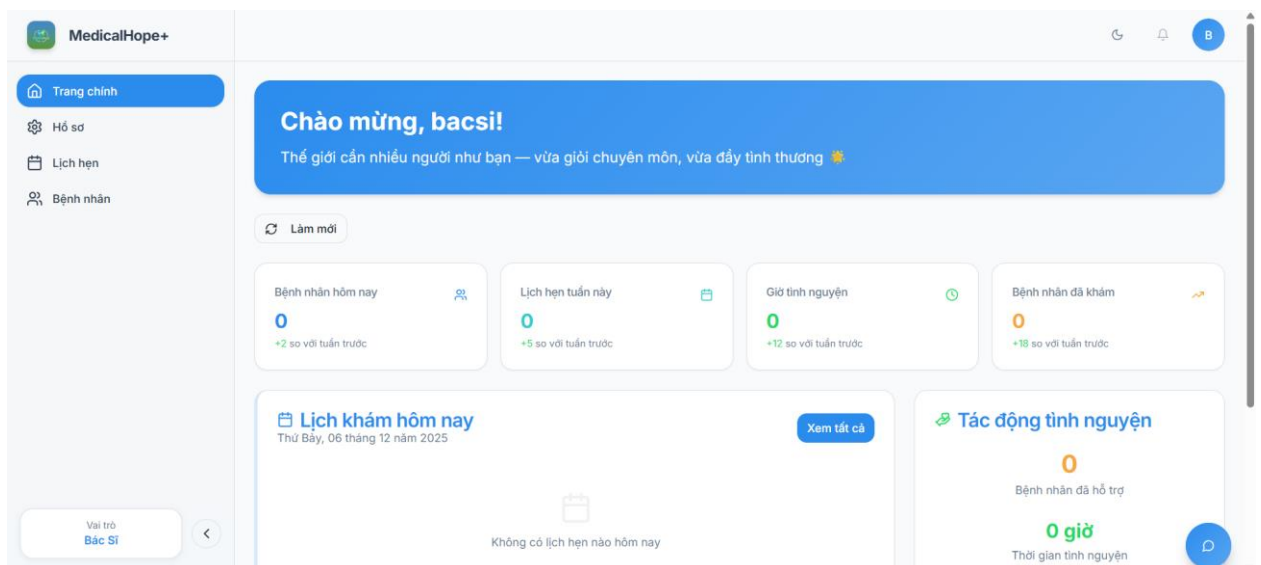


Hình 3.58 Giao diện danh sách yêu cầu hỗ trợ và ủng hộ



Hình 3.59 Giao diện tạo yêu cầu hỗ trợ

3.6.3. *Giao diện Bác sĩ*



Hình 3.60 Giao diện chính cho bác sĩ

MedicalHope+

Trang chính
Hồ sơ
Lịch hẹn
Bệnh nhân

Vai trò
Bác Sĩ

Hồ sơ cá nhân

Xem thông tin tài khoản

B bacsil
Doctor

Email: bacsil@gmail.com
Số điện thoại: 0968264130

Vai trò: Doctor

Ngày sinh: 09/12/2003
Giới tính: Nam

Địa chỉ: Quảng Trị
Nghề nghiệp: Bác Sĩ

Cập nhật hồ sơ

Hình 3.61 Giao diện hồ sơ bác sĩ

MedicalHope+

Trang chính
Hồ sơ
Lịch hẹn
Bệnh nhân

Vai trò
Bác Sĩ

Hồ sơ cá nhân

Chỉnh sửa thông tin của bạn

B bacsil
Doctor

Họ và tên: bacsil
Email: bacsil@gmail.com

Số điện thoại: 0968264130
Vai trò: doctor

Ngày sinh: 09/12/2003
Giới tính: Nam

Địa chỉ: Quảng Trị
Nghề nghiệp: Bác Sĩ

Hình 3.62 Giao diện cập nhật hồ sơ

MedicalHope+

Trang chính
Hồ sơ
Lịch hẹn
Bệnh nhân

Vai trò
Bác Sĩ

Quản lý lịch khám

Quản lý lịch hẹn và lịch rảnh tình nguyện

Lịch hẹn bệnh nhân
Lịch rảnh tình nguyện

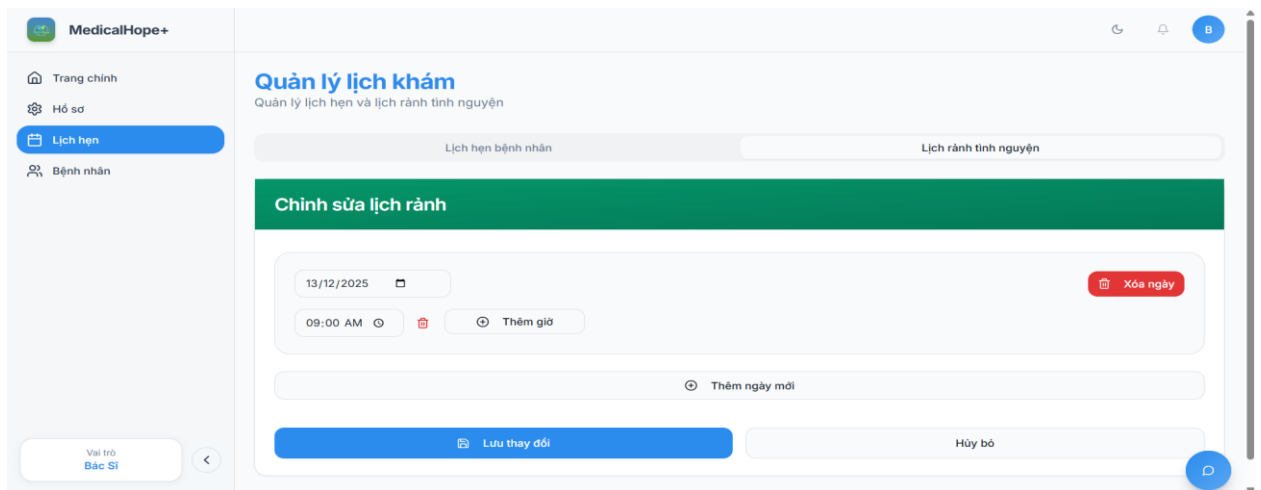
Lịch hẹn từ bệnh nhân

Võ Trọng Quyền
Tái khám
15:10:00 30/10/2025
Đã hủy

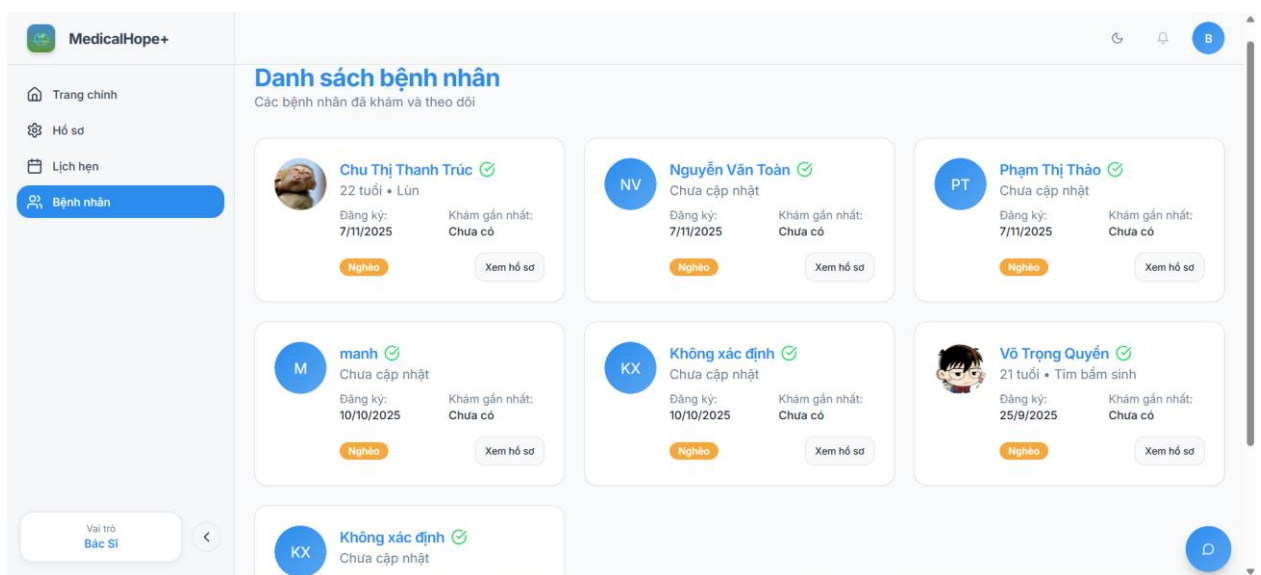
Võ Trọng Quyền
Khám online
07:10:00 31/10/2025
Đã xác nhận

Võ Trọng Quyền
Khám khẩn cấp
16:10:00 31/10/2025
Đã hủy

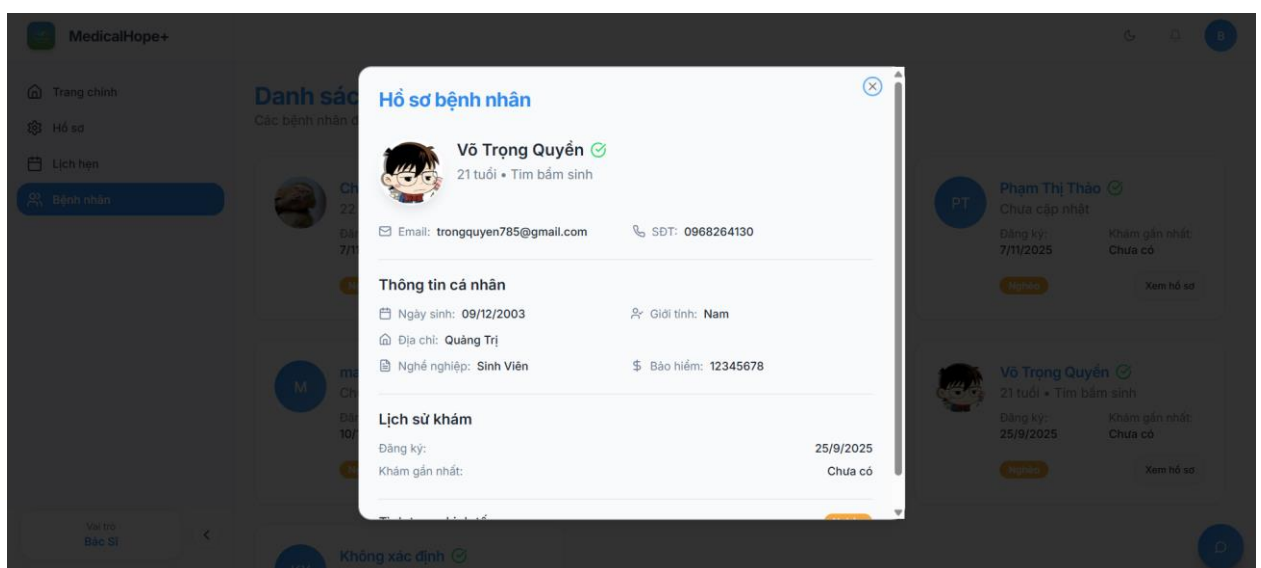
Hình 3.63 Giao diện quản lý lịch khám



Hình 3.64 Giao diện cập nhật lịch rảnh bác sĩ



Hình 3.65 Giao diện danh sách bệnh nhân



Hình 3.66 Giao diện thông tin chi tiết bệnh nhân

3.6.4. Giao diện Người dùng

The registration form is titled "Đăng ký" (Register) and "Tạo tài khoản MedicalHope+ mới" (Create new MedicalHope+ account). It includes fields for "Họ và tên" (Full name) with the example "Nguyễn Văn A", "Email" with "your@email.com", "Số điện thoại" (Phone number) with "0123456789", and "Mật khẩu" (Password) with a masked input. A "Vai trò" (Role) dropdown menu is set to "Chọn vai trò của bạn" (Select your role), with options for "Bệnh nhân" (Patient) - "Tôi cần được hỗ trợ y tế" (I need medical support) and "Bác sĩ" (Doctor) - "Tôi muốn làm tình nguyện viên" (I want to be a volunteer). A blue "Đăng ký" button is at the bottom, with a link "Đã có tài khoản? Đăng nhập" (Already have an account? Log in) below it. A footer link "Quay lại trang chủ" (Return to home) is also present.

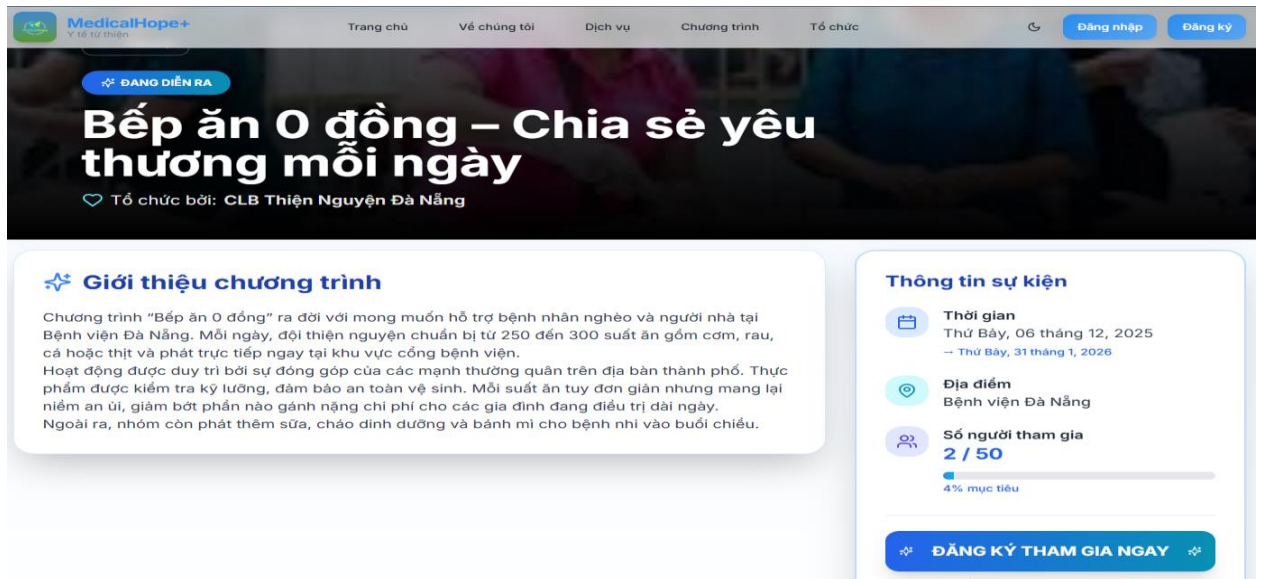
Hình 3.67 Giao diện đăng ký



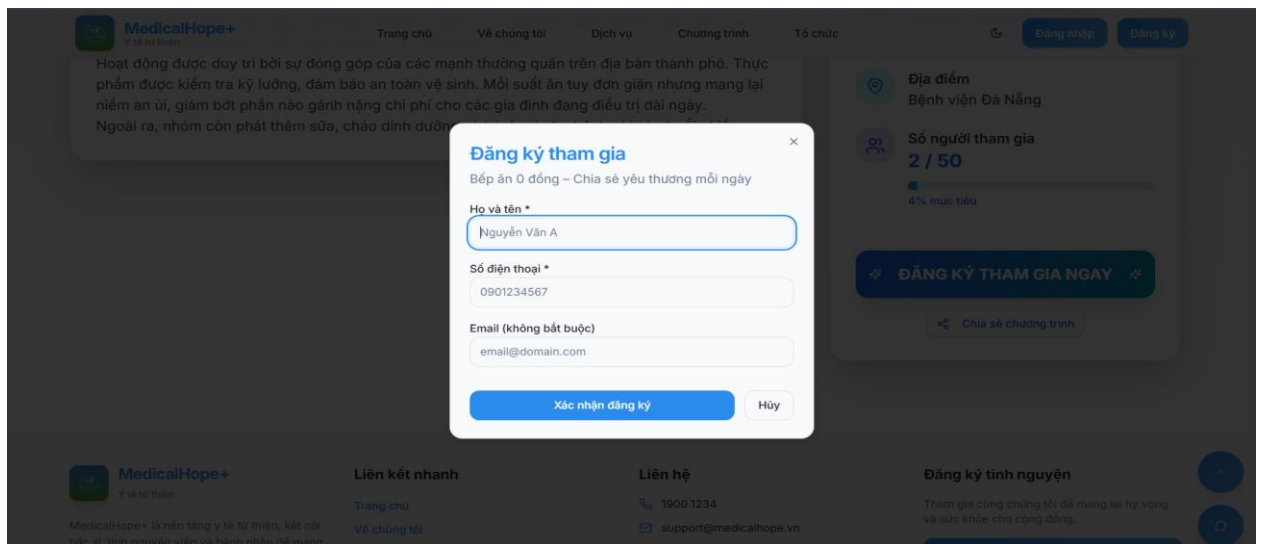
Hình 3.68 Giao diện chính



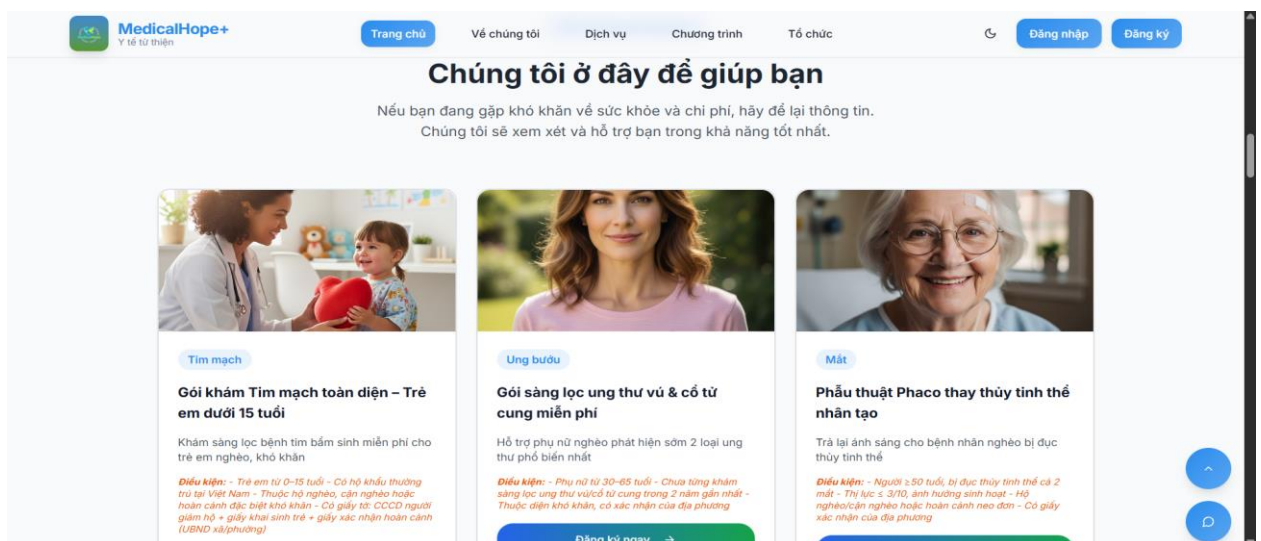
Hình 3.69 Giao diện các hoạt động thiện nguyện đang diễn ra



Hình 3.70 Giao diện chi tiết hoạt động



Hình 3.71 Giao diện đăng ký tham gia hoạt động



Hình 3.72 Giao diện các gói khám miễn phí

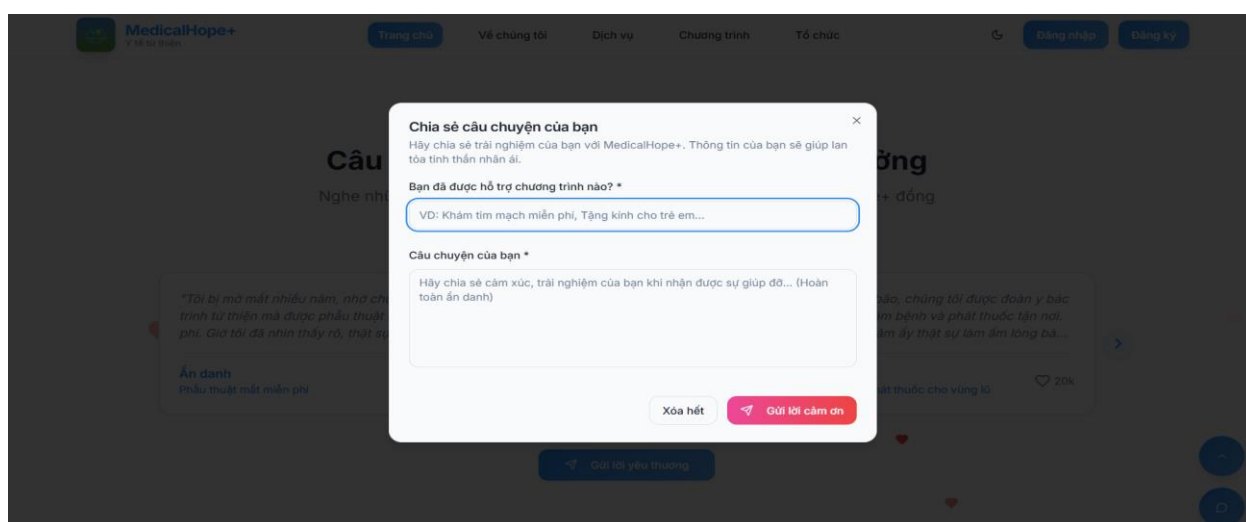
Hình 3.73 Giao diện đăng ký gói khám miễn phí

Hình 3.74 Giao diện các yêu cầu hỗ trợ cần ủng hộ

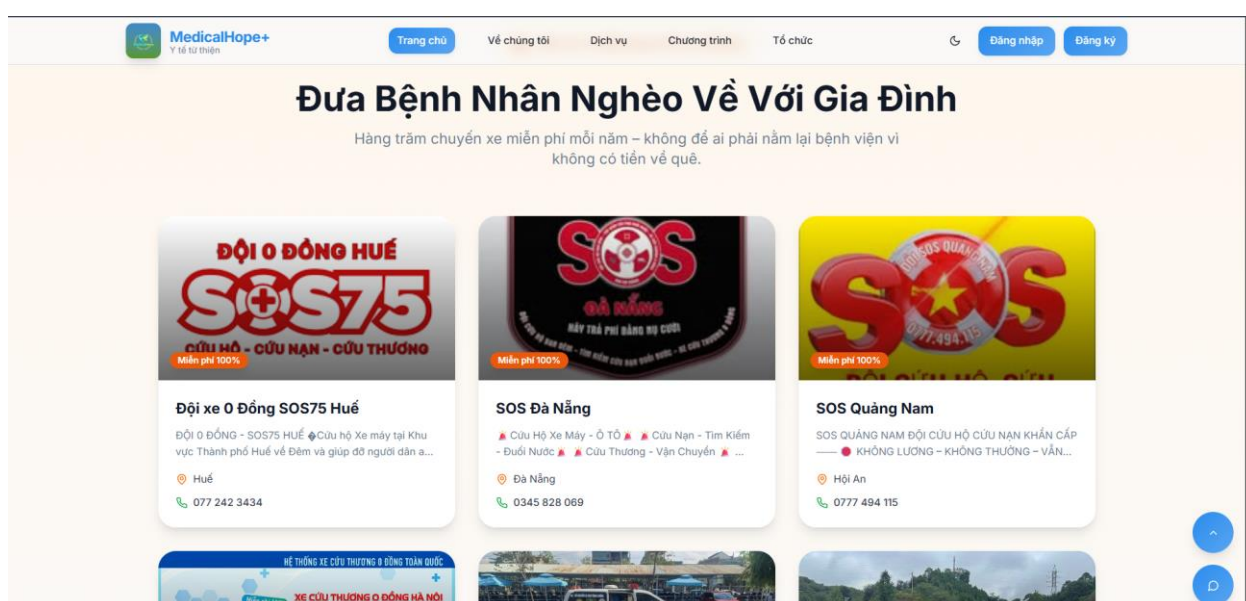
Hình 3.75 Giao diện thanh toán ủng hộ



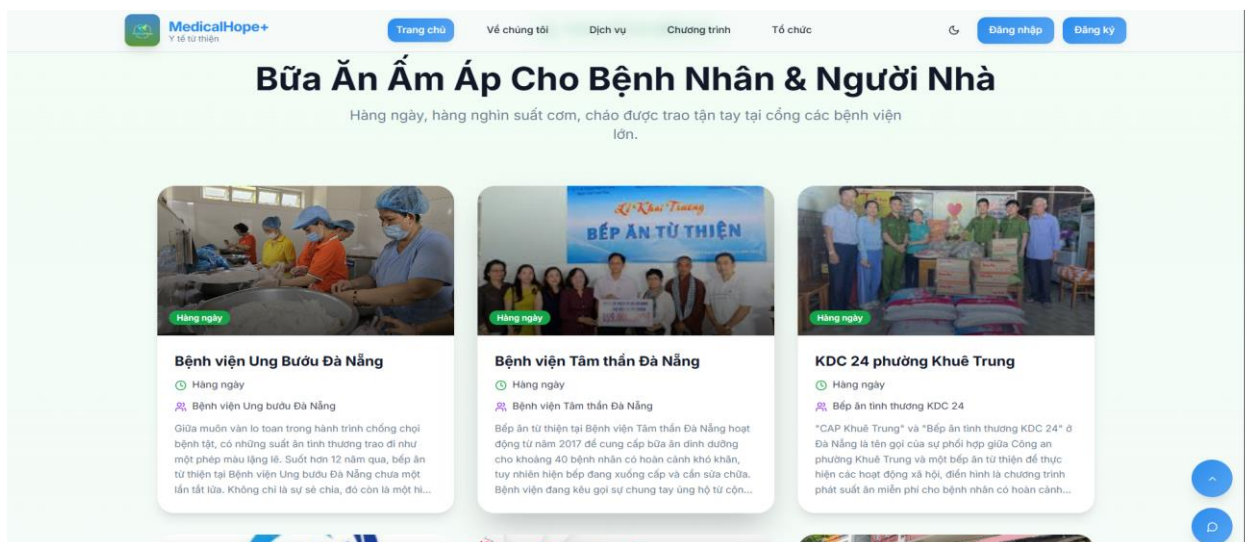
Hình 3.76 Giao diện danh sách chia sẻ lời yêu thương



Hình 3.77 Giao diện tạo lời yêu thương



Hình 3.78 Giao diện thông tin các chuyến xe 0 đồng



Hình 3.79 Giao diện thông tin các bữa cơm 0 đồng



Hình 3.80 Giao diện các tổ chức



Hình 3.81 Giao diện ChatbotAI

Chương 4: CÀI ĐẶT VÀ PHÁT TRIỂN WEBSITE

4.1. Cấu trúc thư mục code

4.1.1. Cấu trúc thư mục code Back End

DATN_MEDICALHOPE/

└─ backend/	
└─  config/	# Tập cấu hình (DB, môi trường...)
└─  controllers/	# Xử lý logic API
└─  middleware/	# Middleware xử lý request/response
└─  models/	# Định nghĩa dữ liệu (ORM models)
└─  node_modules/	# Thư viện bên thứ ba được cài qua npm
└─  routes/	# Định nghĩa các endpoint API
└─  uploads/	# Thư mục lưu trữ file upload
└─  utils/	# Các hàm tiện ích, helpers
└─  .env	# Biến môi trường
└─  .gitignore	# Tập cấu hình bỏ qua các file trong Git
└─  package-lock.json	# File khóa dependency
└─  package.json	# Thông tin gói và phụ thuộc
└─  pnpm-lock.yaml	# File khóa dependency của pnpm
└─  server.js	# Điểm khởi tạo chính của server

4.1.2. Cấu trúc thư mục code Front End

DATN_MEDICALHOPE/

└─ frontend/	
└─  node_modules/	# Thư viện bên thứ ba được cài qua npm
└─  public/	# Tài nguyên tĩnh công khai
└─  src/	# Mã nguồn chính
└─  assets/	# Tài nguyên tĩnh (images, fonts, icons...)
└─  components/	# Các component tái sử dụng
└─  contexts/	# React Context API
└─  hooks/	# Custom React Hooks

└─  i18n/	# Đa ngôn ngữ (internationalization)
└─  lib/	# Thư viện và utilities tùy chỉnh
└─  pages/	# Các trang chính của ứng dụng
└─  stores/	# State management (Redux, Zustand)
└─  App.css	# Style cho App component
└─  App.tsx	# Component gốc của ứng dụng
└─  index.css	# Global styles
└─  main.tsx	# Entry point của ứng dụng
└─  vite-env.d.ts	# TypeScript definitions cho Vite
└─  .env	# Biến môi trường (không commit)
└─  .gitignore	# Các file/folder bỏ qua trong Git
└─  bun.lockb	# Lock file của Bun package manager
└─  components.json	# Cấu hình components (shadcn/ui, etc.)
└─  eslint.config.js	# Cấu hình ESLint (code quality)
└─  index.html	# HTML template chính
└─  package-lock.json	# Lock file của npm
└─  package.json	# Thông tin package và dependencies
└─  pnpm-lock.yaml	# Lock file của pnpm
└─  postcss.config.js	# Cấu hình PostCSS
└─  README.md	# Tài liệu hướng dẫn dự án
└─  tailwind.config.ts	# Cấu hình Tailwind CSS
└─  tsconfig.app.json	# TypeScript config cho app
└─  tsconfig.json	# TypeScript config chính
└─  tsconfig.node.json	# TypeScript config cho Node.js
└─  vite.config.ts	# Cấu hình Vite build tool

KẾT LUẬN

Trên đây là toàn bộ báo cáo đồ án tốt nghiệp. Như vậy, sau một thời gian nỗ lực tìm hiểu, nghiên cứu và phát triển, em đã hoàn thành đồ án tốt nghiệp với đề tài “Xây dựng website quản lý bệnh nhân cho trung tâm y tế MedicalHope tích hợp chatbot AI”.

I. KẾT QUẢ ĐẠT ĐƯỢC

Về công nghệ

- Nắm vững và thành thạo việc sử dụng TypeScript, ReactJS (với Vite + TailwindCSS + shadcn/ui), Node.js, ExpressJS, MongoDB (Mongoose) trong một dự án thực tế quy mô lớn.
- Hiểu sâu về kiến trúc full-stack JavaScript/TypeScript, cách thiết kế RESTful API, quản lý trạng thái phức tạp ở cả frontend và backend, bảo mật JWT, upload file, phân quyền theo vai trò.
- Thành công tích hợp Chatbot AI sử dụng Google Gemini API, hoạt động ổn định, trả lời tiếng Việt, cá nhân hóa theo vai trò người dùng.
- Áp dụng được các kiến thức về phân tích yêu cầu, thiết kế Use Case, Sequence Diagram, mô hình cơ sở dữ liệu NoSQL và triển khai giao diện người dùng hiện đại, responsive, thân thiện với mọi đối tượng (admin – bệnh nhân – bác sĩ).

Về chương trình

- Đã hoàn thành 80% các mục tiêu chức năng đã đề ra trong đề cương: quản lý người dùng đa vai trò, quản lý bệnh nhân & bác sĩ, đặt lịch khám, quản lý gói khám miễn phí, quản lý sự kiện thiện nguyện, tiếp nhận & xử lý yêu cầu hỗ trợ + quyên góp minh bạch, quản lý đối tác, lời cảm ơn, thống kê báo cáo.
- Giao diện được thiết kế thân thiện, trực quan, dễ sử dụng cho mọi đối tượng, đặc biệt là người lớn tuổi và bệnh nhân có hoàn cảnh khó khăn.
- Hệ thống hoạt động ổn định, bảo mật tốt, có khả năng mở rộng và bảo trì lâu dài.

II. HẠN CHẾ

Mặc dù đã cố gắng hết sức, đồ án vẫn còn một số hạn chế mà em nhận thức rõ và mong muốn cải thiện trong tương lai:

- Hiệu suất: Khi dữ liệu lớn (hàng chục nghìn bệnh nhân, hàng nghìn giao dịch quyên góp), một số truy vấn chưa được tối ưu hoàn toàn (chưa áp dụng đầy đủ indexing, pagination ở tất cả các danh sách).

- Bảo mật: Đã triển khai JWT + bcrypt, nhưng chưa tích hợp xác thực hai yếu tố (2FA) và chưa thực hiện audit bảo mật chuyên sâu (penetration testing).
- Tính năng thanh toán: Hiện tại mới chỉ tích hợp được PayPal (chưa triển khai VNPay, Momo, ZaloPay hoặc các cổng thanh toán nội địa khác).
- Ứng dụng di động: Chưa phát triển phiên bản mobile app (React Native hoặc PWA hoàn chỉnh).
- Thông báo thời gian thực: Chưa tích hợp Socket.io hoặc Firebase để gửi thông báo đẩy khi có lịch hẹn mới, yêu cầu hỗ trợ được duyệt, hoặc có người quyên góp.

III. HƯỚNG PHÁT TRIỂN

Dựa trên những hạn chế trên, em đề xuất một số hướng phát triển trong tương lai để hoàn thiện hơn sản phẩm:

- Tối ưu hiệu suất bằng cách áp dụng Redis caching, aggregation pipeline tối ưu, full-text search với MongoDB Atlas Search.
- Nâng cấp bảo mật: triển khai 2FA (OTP/email), rate limiting, helmet + CORS chặt chẽ hơn, và audit bảo mật định kỳ.
- Hoàn thiện thanh toán đa kênh: tích hợp thêm VNPay, Momo, ZaloPay và chuyển PayPal sang môi trường production.
- Phát triển **PWA hoặc ứng dụng di động React Native để người dùng có thể truy cập mọi lúc mọi nơi mà không cần trình duyệt.
- Tích hợp thông báo thời gian thực (Socket.io hoặc Firebase Cloud Messaging) cho lịch hẹn, quyên góp, duyệt yêu cầu.
- Mở rộng AI: nâng cấp chatbot thành trợ lý ảo thông minh hơn (hỗ trợ voice, phân tích hình ảnh bệnh án, gợi ý bác sĩ phù hợp).
- Triển khai thực tế cho trung tâm MedicalHope và các tổ chức từ thiện khác, thu thập phản hồi thực tế để liên tục cải tiến.

TÀI LIỆU THAM KHẢO

- [1] React Team. React – The library for web and native user interfaces.
- [2] Microsoft. TypeScript – JavaScript with syntax for types.
- [3] OpenJS Foundation. Node.js – JavaScript runtime built on Chrome’s V8 engine.
- [4] Express.js Team. Express – Fast, unopinionated, minimalist web framework for Node.js.
- [5] MongoDB, Inc. MongoDB – The Developer Data Platform.
- [6] Google AI. Gemini API – Build with Google’s most capable AI models.
- [7] PayPal Developer. PayPal Checkout Integration – Accept payments securely.