

**TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**



**KHÓA LUẬN TỐT NGHIỆP
PHÁT TRIỂN ỨNG DỤNG ĐẶT LỊCH
KHÁM BỆNH CHO PHÒNG KHÁM TƯ
NHÂN**

Giảng viên hướng dẫn **ThS NGUYỄN KHẮC QUỐC**

Sinh viên thực hiện: **NGUYỄN MAI DUY KHOA**

Mã số sinh viên: **110121211**

Lớp: **DA21TTC**

Khoá: **2021**

Trà Vinh, tháng ... năm 2025

**TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**



**KHÓA LUẬN TỐT NGHIỆP
PHÁT TRIỂN ỨNG DỤNG ĐẶT LỊCH
KHÁM BỆNH CHO PHÒNG KHÁM TƯ
NHÂN**

Giảng viên hướng dẫn **ThS NGUYỄN KHẮC QUỐC**
Sinh viên thực hiện: **NGUYỄN MAI DUY KHOA**
Mã số sinh viên: **110121211**
Lớp **DA21TTC**
Khoá **2021**

Trà Vinh, tháng ... năm 2025

LỜI MỞ ĐẦU

Trong cuộc sống hiện đại, internet đã phát triển một cách mạnh mẽ và chiếm một phần quan trọng trong cuộc sống con người, từ đó các dịch vụ online cũng được thúc đẩy và bắt đầu trở thành một phương thức kinh doanh hoàn toàn mới trên thị trường hiện nay. Từ việc mua sắm, đặt hàng người dùng chỉ cần vài thao tác đơn giản trên điện thoại hoặc máy tính là có thể dễ dàng hoàn tất với một sự phục vụ nhanh chóng. Điều này cho thấy các ứng dụng phục vụ việc khai thác thị trường online ngày càng phát triển và trở thành một xu hướng trong hoạt động kinh doanh hiện nay.

Xuất phát từ thực tế đó, em chọn thực hiện đề tài phát triển ứng dụng đặt lịch hẹn khám bệnh cho phòng khám tư nhân. Hệ thống được xây dựng với mục tiêu không chỉ phục vụ cho việc học tập, mà còn mang tính thực tiễn cao có thể áp dụng cho các phòng khám trong việc phục vụ và hỗ trợ khách hàng một cách nhanh chóng. Thông qua các tình trạng thực tế để đánh giá khách quan thì việc phát triển một ứng dụng đặt lịch khám bệnh thông qua internet sẽ làm giảm thiểu hàng chờ cho các phòng khám luôn trong tình trạng đông kín người hiện nay, việc giảm hàng chờ cũng kéo theo việc tăng trải nghiệm người dùng, mang đến sự hài lòng của khách hàng khi không phải chờ đợi dịch vụ có thể lên đến vài giờ đồng hồ.

Đề tài sử dụng các công nghệ hiện đại như: React Vite để dễ dàng thiết kế frontend, NodeJS cho backend, MongoDB để lưu trữ dữ liệu, tích hợp Cloudinary để quản lý hình ảnh và sử dụng JWT để xác thực người dùng, Vonage để gửi OTP xác thực, thuật toán mã hóa để bảo vệ tài khoản người dùng. Hệ thống hỗ trợ phân quyền rõ ràng (admin, khách hàng, bác sĩ, trợ tá), cho phép người dùng đăng nhập dễ dàng thông qua số điện thoại và OTP.

Thông qua đề tài này, em mong muốn có cơ hội áp dụng kiến thức lập trình web vào một sản phẩm cụ thể, đồng thời rèn luyện kỹ năng xây dựng hệ thống, làm việc với cơ sở dữ liệu, bảo mật và tổ chức mã nguồn theo hướng chuyên nghiệp hơn.

LỜI CẢM ƠN

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Nguyễn Khắc Quốc là người đã tận tình hướng dẫn, truyền đạt kiến thức chuyên môn cũng như định hướng thực hiện đề tài trong suốt thời gian qua. Sự tận tâm, trách nhiệm và những góp ý quý báu từ Thầy là yếu tố quan trọng giúp em hoàn thiện khóa luận này.

Em cũng xin chân thành cảm ơn các Thầy Cô trong Khoa Công nghệ thông tin đã giảng dạy và tạo điều kiện thuận lợi để em tích lũy được nền tảng kiến thức vững chắc trong suốt quá trình học tập.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến gia đình, bạn bè và những người thân yêu, những người luôn đồng hành, động viên và tiếp thêm tinh thần cho em trong suốt hành trình học tập và thực hiện khóa luận.

Mặc dù đã nỗ lực hoàn thành tốt nhất trong khả năng của mình, nhưng chắc chắn khóa luận vẫn không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ Thầy Cô để em có thể rút kinh nghiệm và tiếp tục hoàn thiện bản thân trong tương lai.

[illegible]

(ký và ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC TRÀ VINH

Độc lập – Tự do – Hạnh Phúc

(Của giảng viên hướng dẫn)

Chức danh: Học vị:

NHẬN XÉT

2. Ưu điểm:

.....

.....

.....

.....

3. Khuyết điểm:

.....

.....

.....

.....

.....

4. Điểm mới đề tài:

.....

.....

.....

.....

.....

5. Giá trị thực trên đề tài:

.....

.....

.....

.....

.....

.....

.....

7. Đề nghị sửa chữa bổ sung:

.....

.....

.....

.....

.....

.....

.....

8. Đánh giá:

.....

.....

.....

.....

Trà Vinh, *ngày tháng năm 20...*

Giảng viên hướng dẫn

(Ký & ghi rõ họ tên)

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the entire width of the page, providing a guide for handwriting or typing. There are no margins, text, or other markings on the page.

(ký và ghi rõ họ tên)

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP

(Của cán bộ chấm đồ án, khóa luận)

Họ và tên người nhận xét:

Chức danh: Học vị:

Chuyên ngành:

Cơ quan công tác:

Tên sinh viên:

Tên đề tài đồ án, khóa luận tốt nghiệp:

.....

.....

I. Ý KIẾN NHẬN XÉT

1. Nội dung:

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Điền mới các kết quả của đồ án, khóa luận:

.....

.....

3. Ứng dụng thực tế:

(Các câu hỏi của giáo viên phản biện)

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

III. KẾT LUẬN

(Ghi rõ đồng ý hay không đồng ý cho bảo vệ đồ án khóa luận tốt nghiệp)

.....

.....

.....

.....

.....

....., ngày tháng năm 20...

Người nhận xét

(Ký & ghi rõ họ tên)

MỤC LỤC

CHƯƠNG 1 TỔNG QUAN	2
1.1 Lý do chọn đề tài	2
1.2 Mục tiêu đề tài	2
1.3 Đối tượng nghiên cứu	3
1.4 Phạm vi nghiên cứu	3
1.5 Phương pháp nghiên cứu	3
CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT	4
2.1 API và Restful API:	4
2.1.1 Khái niệm:	4
2.1.2 Lợi ích của API RESTful:	5
2.1.3 Phương thức hoạt động:	6
2.2 Tổng quan về Node.js:	7
2.2.1 Giới thiệu Node.js:	7
2.2.2 Cách hoạt động của Node.js:	7
2.2.3 Ưu điểm của Node.js:	9
2.2.4 Nhược điểm của Node.js:	10
2.2.5 Ứng dụng của Node.js:	11
2.3 Tổng quan về ReactJs và React Vite:	12
2.3.1 Giới thiệu ReactJs và React Vite:	12
2.3.2 Cấu trúc của ReactJs:	13
2.3.3 Cách cài đặt và tạo dự án ReactJs:	14
2.3.4 Giới thiệu Tailwin CSS:	16
2.3.5 Ưu điểm của Tailwin CSS:	17
2.3.6 Nhược điểm của Tailwin CSS:	17
2.3.7 Cài đặt và sử dụng Tailwin CSS:	17

2.4 Tổng quan về MongoDB:	19
2.4.1 Sơ lược về NoSQL:	19
2.4.2 Giới thiệu MongoDB:	19
2.4.3 Một số khái niệm trong MongoDB:	20
2.4.4 Ưu điểm của MongoDB:	21
2.4.5 Nhược điểm của MongoDB	21
CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU	23
3.1 Phân tích và đặt tả yêu cầu hệ thống:	23
3.1.1 Phân tích chức năng:	23
3.1.2 Yêu cầu phi chức năng	23
3.2 Thiết kế hệ thống	24
3.2.1 Thiết kế cơ sở dữ liệu	24
3.2.2 Sơ đồ Use Case:	26
CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU	27
4.1 Giao diện khách hàng:	27
4.1.1 Trang chủ:	27
4.1.2 Trang đăng nhập:	27
4.1.3 Trang đặt hẹn khám:	28
4.1.4 Trang lịch khám chữa:	28
4.1.5 Trang lịch sử đặt hẹn:	29
4.1.6 Trang cá nhân:	29
4.2 Giao diện quản trị:	30
4.2.1 Trang quản lý thông tin phòng khám:	30
4.2.2 Trang quản lý tài khoản bác sĩ:	31
4.2.3 Trang quản lý đơn hẹn:	31
CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	33

5.1 Kết luận:	33
5.2 Hướng phát triển:.....	33
DANH MỤC TÀI LIỆU THAM KHẢO	34

DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH

Hình 1. Mô hình hoạt động của API.....	7
Hình 2. Mô hình hoạt động của Node.js.....	9
Hình 3. Giới thiệu Tailwind CSS	16
Hình 4. Nội dung tệp postcss.config.js.....	18
Hình 5. Nội dung tệp styles.css	18
Hình 6. Thay đổi nội dung của file tailwind.config.js.....	18
Hình 7. Ví dụ về Collection trong MongoDB	20
Hình 8. Ví dụ minh họa cho Document trong MongoDB	21
Hình 9. Sơ đồ usecase đối với khách hàng.....	26
Hình 10. Sơ đồ usecase đối với quản trị viên.....	26
Hình 11. Giao diện khách hàng: Trang chủ.....	27
Hình 12. Trang đăng nhập	28
Hình 13. Trang đặt hẹn khám	28
Hình 14 Trang lịch khám chữa	29
Hình 15. Trang lịch sử đặt hẹn	29
Hình 16. Trang cá nhân	30
Hình 17. Trang quản lý thông tin phòng khám	30
Hình 18. Trang quản lý tài khoản bác sĩ.....	31
Hình 19. Trang quản lý đơn hẹn.....	32

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

API	Application Programming Interface(Giao diện lập trình ứng dụng)
CLI	Command Line Interface (Giao diện dòng lệnh)
CSS	Cascading Style Sheets (Tập tin định kiểu theo tầng)
DB	Database (Cơ sở dữ liệu)
DOM	Document Object Model
HMR	Hot Module Replacement
IoT	Internet of Things (Internet vạn vật)
I/O	Input/Output (Nhập/Xuất)
JS	JavaScript
JWT	JSON Web Token
NPM	Node Package Manager
RDBMS	Relational Database Managemen System (Cơ sở dữ liệu quan hệ)
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol (Giao thức truyền thư đơn giản)
SSG	Static Site Generation (Tạo trang tĩnh sẵn)
SSR	Server-Side Rendering (Render phía server)
URI	Uniform Resource Identifier (Bộ định vị tài nguyên đồng nhất)

CHƯƠNG 1 TỔNG QUAN

1.1 Lý do chọn đề tài

Trong công cuộc chuyển đổi số của đất nước ngày nay các dịch vụ thông qua mạng internet ngày càng phát triển mạnh mẽ, các ứng dụng phục vụ cho các dịch vụ trên cũng xuất hiện ngày một nhiều. Việc xây dựng một ứng dụng đặt lịch khám không chỉ giúp giảm hàng chờ ở các phòng khám, làm giảm thiểu việc quá tải mà còn tăng trải nghiệm người dùng khi khách hàng không còn phải chờ quá lâu để được đăng ký sử dụng dịch vụ của phòng khám. Từ thực tiễn đó, em lựa chọn thực hiện đề tài "Xây dựng ứng dụng đặt lịch khám bệnh cho phòng khám tư nhân" với mong muốn tạo ra một ứng dụng web hiện đại, có thể áp dụng thực tế, dễ sử dụng cho mọi đối tượng khách hàng từ người trẻ đến người cao tuổi, đồng thời giúp em củng cố và vận dụng kiến thức đã học vào một sản phẩm cụ thể.

1.2 Mục tiêu đề tài

Đề tài hướng đến việc xây dựng một ứng dụng web phục vụ cho việc đặt lịch khám bệnh và truyền tải những thông tin mà phòng khám muốn truyền tải đến khách hàng, ứng dụng được thiết kế nhằm tối ưu thao tác, hướng dẫn rõ ràng dễ phù hợp với mọi đối tượng khách hàng, nhằm đáp ứng nhu cầu về việc thăm khám và kiểm tra sức khỏe luôn được người Việt Nam chú trọng từ trước đến nay. Việc xây dựng một ứng dụng đặt khám online cũng giúp những khách hàng cách xa phòng khám có thể thông qua ứng dụng hiểu khái quát về phòng khám và đưa ra những lựa chọn phù hợp mà không phải tốn những khoản phí phát sinh khác. Cụ thể, đề tài đặt ra các mục tiêu như sau:

- Xây dựng giao diện website thân thiện, dễ sử dụng cho cả người dùng và quản trị viên, đảm bảo hiển thị đầy đủ những thông tin và hướng dẫn cần thiết cho người dùng.

- Phát triển hệ thống backend sử dụng NodeJS, thiết kế kiến trúc mô-đun rõ ràng, dễ mở rộng và bảo trì.

- Thiết kế cơ sở dữ liệu sử dụng MongoDB, đảm bảo lưu trữ hiệu quả và truy xuất nhanh chóng dữ liệu.

- Tích hợp chức năng xác thực người dùng bằng JWT và OTP hỗ trợ phân quyền rõ ràng giữa admin, khách hàng, bác sĩ và trợ tá.

- Xây dựng các chức năng chính như: cài đặt thông tin phòng khám, đặt lịch hẹn khám, xác nhận phản hồi lịch hẹn.

1.3 Đối tượng nghiên cứu

Đề tài tập trung nghiên cứu các thành phần chính như:

- Giao diện người dùng với các thông tin về phòng khám được hiển thị tại trang chủ, chức năng đặt lịch hẹn khám với thông tin về bác sĩ được hẹn (Tên, hình ảnh, giới thiệu cơ bản) sẽ hiển thị với người dùng.

- Trang quản trị với chức năng quản lý các đơn hẹn và phản hồi với người dùng.

1.4 Phạm vi nghiên cứu

Về chức năng:

- Xây dựng ứng dụng với các tính năng cốt lõi: đăng nhập, phân quyền, quản lý đơn hẹn, quản lý bác sĩ, thay đổi thông tin phòng khám, đăng tải bài viết.

- Hệ thống phân quyền gồm: admin, khách hàng, bác sĩ và trợ tá.

Về công nghệ:

- Frontend: sử dụng React Vite để xây dựng giao diện người dùng.

- Backend: sử dụng NodeJS để xây dựng API RESTful với bảo mật JWT.

- Cơ sở dữ liệu: sử dụng MongoDB, kết hợp thư viện Mongoose.

- Tích hợp Cloudinary để lưu trữ ảnh và Vonage để gửi OTP qua số điện thoại.

1.5 Phương pháp nghiên cứu

Phương pháp khảo sát và thu thập quy trình:

- Tìm hiểu quy trình đăng ký và khám chữa trên các website của các bệnh viện và phòng khám hiện tại nhằm xác định những thông tin tối thiểu cần thiết cho quy trình đặt hẹn.

- Xác định đặc điểm của các nhóm người dùng chính: Admin, Khách hàng từ đó xây dựng các luồng chức năng và phân quyền phù hợp.

Phương pháp phân tích và thiết kế hệ thống:

- Phân tích chức năng: Xây dựng sơ đồ use case để xác định đầy đủ các chức năng của hệ thống như: đăng nhập, quản lý đơn hẹn, xác nhận đơn hẹn, quản lý thông tin phòng khám, tài khoản nhân sự và các bài viết.

- Phân tích yêu cầu phi chức năng: Bao gồm khả năng mở rộng, hiệu năng, bảo mật (JWT, phân quyền), tính thân thiện với người dùng.

Phương pháp thiết kế:

- Thiết kế giao diện người dùng: với các chức năng được hiển thị rõ ràng không phức tạp, màu sắc đơn giản, hài hòa.

- Thiết kế hệ thống backend: Xây dựng backend với NodeJS, phân tách các lớp controller, model, route dễ dàng mở rộng và bảo trì hệ thống.

- Thiết kế cơ sở dữ liệu: Sử dụng MongoDB kết hợp Mongoose, đảm bảo khả năng mở rộng và linh hoạt trong lưu trữ tài liệu phi cấu trúc.

Phương pháp triển khai: Sử dụng React Vite cùng với Tailwind CSS để xây dựng giao diện website. Xây dựng API RESTful với NodeJs. Tích hợp các dịch vụ khác: Vonage để gọi OTP với số điện thoại thực, Cloudinary để lưu trữ hình ảnh.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

2.1 API và Restful API:

2.1.1 Khái niệm:

API (Application Programming Interface) là một giao diện lập trình ứng dụng, cho phép các hệ thống hoặc thành phần phần mềm giao tiếp với nhau thông qua một tập hợp các quy tắc và giao thức nhất định. API không nhất thiết phải là web – nó có thể là một thư viện trong hệ điều hành, một interface trong chương trình C/C++, hoặc một RESTful service trên Internet. Trong ngữ cảnh Web API, API được hiểu là một

HTTP-based interface cho phép client (ví dụ: frontend, mobile app, system khác...) gửi request đến backend/server để thực hiện một hành động nào đó (lấy dữ liệu, thêm, xóa, sửa...).[7]

REST (Representational State Transfer) là là một kiến trúc phần mềm được giới thiệu bởi Roy Fielding trong luận án tiến sĩ năm 2000, REST không phải là một giao thức hay công nghệ, mà là tập hợp các ràng buộc giúp xây dựng hệ thống phân tán linh hoạt, mở rộng, dễ bảo trì. Sáu ràng buộc bao gồm:[6]

- Client-Server: Phân tách giao diện người dùng (client) và lưu trữ/logic (server).

- Stateless: Mỗi request từ client phải tự chứa đủ thông tin, server không lưu trạng thái giữa các lần gọi.

- Cacheable: Các response phải cho phép hoặc không cho phép cache rõ ràng để cải thiện hiệu suất.

- Uniform Interface: Giao diện thống nhất – điều này quan trọng nhất, bao gồm:

 - + Sử dụng đúng HTTP verbs (GET, POST...)

 - + Tài nguyên được định danh bằng URI

 - + Dữ liệu đại diện (representation) dùng JSON/XML/etc

 - + HATEOAS (Hypermedia as the Engine of Application State)

- Layered System: Kiến trúc gồm nhiều lớp – client không cần biết backend có qua proxy/gateway/load balancer nào.

- Code on Demand: Server có thể cung cấp mã (JS...) cho client thực thi động.

RESTful là các API tuân thủ đúng (hoặc gần đúng) các ràng buộc của REST.[6]

2.1.2 Lợi ích của API RESTful:

API RESTful có những lợi ích sau:

- Khả năng thay đổi quy mô: Các hệ thống triển khai API REST có thể thay đổi quy mô một cách hiệu quả vì REST tối ưu hóa các tương tác giữa client và server. Tình trạng phi trạng thái loại bỏ tải của server vì server không phải giữ lại thông tin

yêu cầu của client trong quá khứ. Việc lưu bộ nhớ đệm được quản lý tốt sẽ loại bỏ một phần hoặc hoàn toàn một số tương tác giữa client và server. Tất cả các tính năng này hỗ trợ khả năng thay đổi quy mô mà không gây ra tắc nghẽn giao tiếp làm giảm hiệu suất.

- Sự linh hoạt: Các dịch vụ web RESTful hỗ trợ phân tách hoàn toàn giữa client và server. Các dịch vụ này đơn giản hóa và tách riêng các thành phần server khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi ở nền tảng hoặc công nghệ tại ứng dụng server không ảnh hưởng đến ứng dụng client. Khả năng phân lớp các chức năng ứng dụng làm tăng tính linh hoạt hơn nữa. Ví dụ: các nhà phát triển có thể thực hiện các thay đổi đối với lớp cơ sở dữ liệu mà không cần viết lại logic ứng dụng.

- Sự độc lập: Các API REST không phụ thuộc vào công nghệ được sử dụng. Có thể viết cả ứng dụng client và server bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API. Đồng thời cũng có thể thay đổi công nghệ cơ sở ở hai phía mà không ảnh hưởng đến giao tiếp.

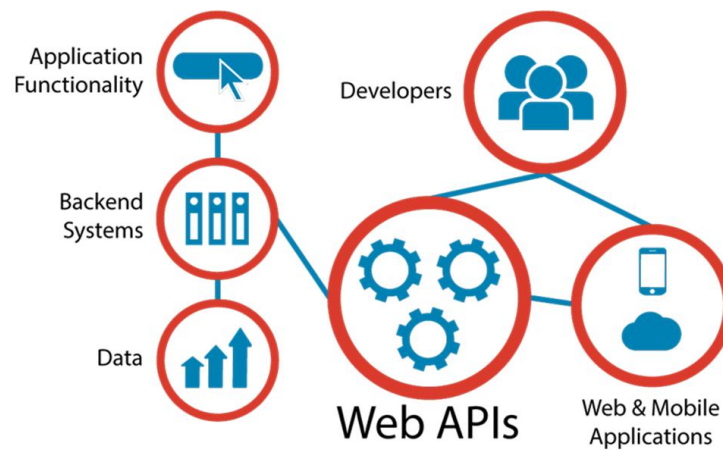
2.1.3 Phương thức hoạt động:

Client gửi một yêu cầu đến server. Client làm theo tài liệu API để định dạng yêu cầu theo cách mà server hiểu được.

Server xác thực và xác nhận máy khách có quyền đưa ra yêu cầu đó.

Server nhận yêu cầu và xử lý trong nội bộ.

Server trả về một phản hồi đến client. Phản hồi chứa thông tin cho client biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà client yêu cầu.



Hình 1. Mô hình hoạt động của API

2.2 Tổng quan về Node.js:

2.2.1 Giới thiệu Node.js:

Node.js là một môi trường thực thi (runtime environment) cho JavaScript với mã nguồn mở và đa nền tảng.

Node.js là mã nguồn mở: Mã nguồn của Node.js là công khai. Và nó được duy trì bởi những người đóng góp từ khắp nơi trên thế giới.

Node.js là môi trường thực thi JavaScript.

Các trình duyệt như Chrome và Firefox có môi trường thực thi. Đó là lí do tại sao các trình duyệt có thể chạy mã JavaScript. Trước khi Node.js được tạo ra thì JavaScript chỉ có thể được chạy trên trình duyệt. Và chỉ được sử dụng để xây dựng các ứng dụng front-end.

Node.js cung cấp một môi trường chạy JavaScript ngoài trình duyệt. Nó được xây dựng trên công cụ JavaScript Chrome V8. Điều này cho phép xây dựng các ứng dụng back-end bằng cách sử dụng cùng một ngôn ngữ lập trình JavaScript.

2.2.2 Cách hoạt động của Node.js:

Node.js được thiết kế dựa trên mô hình event-driven (hướng sự kiện) và kiến trúc non-blocking I/O (nhập/xuất không chặn), cho phép xử lý hiệu quả hàng ngàn kết nối đồng thời trên một luồng duy nhất. Trái tim của cơ chế này chính là sự kết hợp

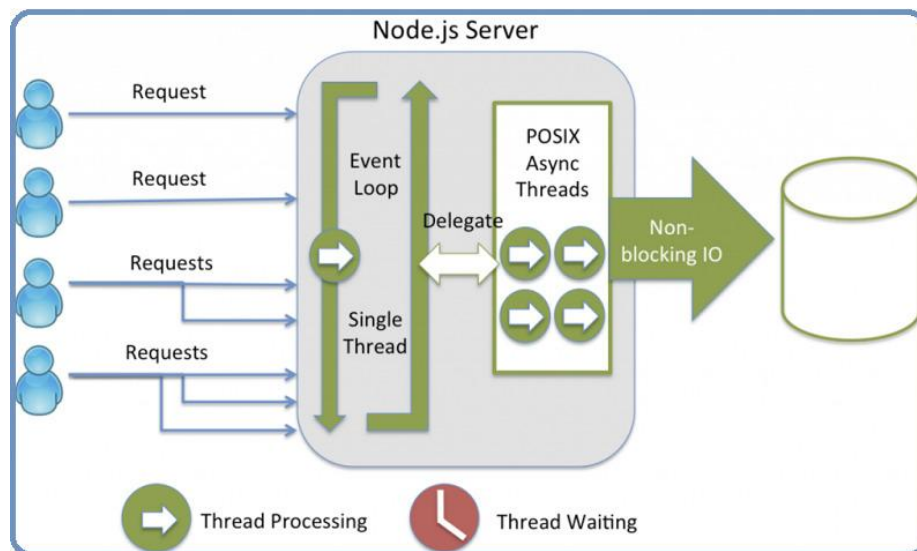
giữa event loop, libuv và mô hình thiết kế Reactor Pattern – ba thành phần then chốt giúp Node.js đạt được hiệu suất cao trong môi trường mạng hiện đại.

Node.js không tạo luồng mới cho mỗi yêu cầu như các nền tảng server truyền thống, mà sử dụng một event loop đơn luồng để theo dõi các sự kiện I/O. Khi một thao tác bất đồng bộ (chẳng hạn như đọc file hoặc truy vấn cơ sở dữ liệu) được gọi, nó không được thực thi ngay lập tức mà được ủy quyền cho hệ thống nền là libuv. Libuv là một thư viện C đa nền tảng, chịu trách nhiệm thực hiện các tác vụ I/O bất đồng bộ thông qua cơ chế event demultiplexer như epoll, kqueue, hoặc IOCP tùy vào hệ điều hành. Sau khi một sự kiện hoàn tất, libuv sẽ đưa kết quả vào hàng đợi sự kiện (event queue) và callback tương ứng sẽ được thực thi trong chu kỳ tiếp theo của event loop.

Cơ chế này phản ánh trực tiếp mô hình Reactor Pattern, trong đó ứng dụng đăng ký các trình xử lý (callback) với một bộ phân phối sự kiện trung tâm (event demultiplexer). Khi một sự kiện phát sinh (ví dụ, dữ liệu đến từ socket), hệ thống sẽ “kích hoạt” callback đã đăng ký. Cách tiếp cận này giúp Node.js tránh được việc sử dụng nhiều luồng, đồng thời loại bỏ sự phức tạp liên quan đến đồng bộ hóa, deadlock và race conditions.

Ngoài ra, đối với những thao tác mà hệ điều hành không hỗ trợ non-blocking trực tiếp (như thao tác file), libuv sử dụng một thread pool nội bộ để mô phỏng tính bất đồng bộ, đảm bảo rằng luồng chính vẫn không bị chặn trong quá trình xử lý.

Kết quả của kiến trúc này là một mô hình lập trình đơn giản nhưng cực kỳ hiệu quả cho các ứng dụng có tính chất I/O-intensive như API server, chat app hoặc hệ thống streaming. Node.js cung cấp khả năng mở rộng cao với chi phí bộ nhớ thấp, đặc biệt phù hợp với mô hình microservices và các kiến trúc thời gian thực hiện đại.^{[5][9]}



Hình 2. Mô hình hoạt động của Node.js

2.2.3 Ưu điểm của Node.js:

Hiệu suất cao: Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, cho phép biên dịch mã JavaScript thành mã máy nhanh chóng. Nhờ đó, thời gian thực thi của Node.js rất nhanh, làm tăng hiệu suất của các ứng dụng.

Hệ sinh thái phong phú: Với hơn 50,000 gói có sẵn trong Node Package Manager (NPM), các nhà phát triển có thể dễ dàng tìm và sử dụng các thư viện theo nhu cầu của họ mà không cần phải viết lại từ đầu, tiết kiệm đáng kể thời gian và công sức.

Xử lý bất đồng bộ và không chặn (Asynchronous and Non-blocking): Node.js hoạt động một cách bất đồng bộ và không chặn các hoạt động I/O, nghĩa là nó không cần chờ đợi API trả về dữ liệu trước khi tiếp tục xử lý yêu cầu tiếp theo. Điều này làm cho Node.js trở nên lý tưởng cho việc xây dựng các ứng dụng web thời gian thực và xử lý dữ liệu lớn.

Tính nhất quán trong mã nguồn: Node.js cho phép sử dụng cùng một ngôn ngữ lập trình (JavaScript) cho cả phía server và máy khách. Điều này không chỉ giúp giảm thiểu sự không đồng bộ giữa client và server mà còn làm cho việc bảo trì và quản lý mã nguồn trở nên dễ dàng hơn.

Khả năng mở rộng: Node.js hỗ trợ xây dựng các ứng dụng có khả năng mở rộng cao thông qua mô hình sự kiện và bất đồng bộ của mình. Điều này cho phép xử lý hàng ngàn kết nối đồng thời mà không làm giảm hiệu suất.

Ngôn ngữ quen thuộc: Vì Node.js là một khung làm việc JavaScript, nó trở thành lựa chọn lý tưởng cho những nhà phát triển đã quen thuộc với JavaScript. Điều này làm cho quá trình học tập và phát triển dự án với Node.js trở nên dễ dàng hơn nhiều. [5][9]

2.2.4 Nhược điểm của Node.js:

Đầu tiên phải kể đến chính là nó không có khả năng mở rộng. Do đó mà nó không thể tận dụng được lợi thế mô hình đa lõi ở các phần cứng cấp server trên thị trường hiện nay.

Không tối ưu cho CPU-intensive: Node.js chạy trên single-thread nên khi xử lý tác vụ như mã hóa, nén, hoặc xử lý ảnh sẽ làm chậm toàn bộ ứng dụng.

Callback hell: Gọi lồng nhau nhiều callback gây khó đọc, khó bảo trì. Mặc dù Promises và async/await đã cải thiện, nhưng vẫn cần cân trọng trong thiết kế.

Không đồng bộ mặc định (asynchronous by default): Đòi hỏi lập trình viên phải hiểu rõ mô hình bất đồng bộ, tránh lỗi race condition và callback nesting.

Quản lý lỗi phức tạp: Việc xử lý lỗi trong môi trường bất đồng bộ yêu cầu kỹ thuật rõ ràng.

Thiếu tính nhất quán trong thư viện cộng đồng: Dù npm lớn nhưng chất lượng không đồng đều, dễ gây lỗi bảo mật nếu không đánh giá kỹ.

Không phù hợp với xử lý đa tiến trình mặc định: Cần thêm module `cluster` hoặc `worker_threads` để tận dụng đa core, không đơn giản như trong Java hoặc C#.

API và nền tảng thay đổi nhanh: Cập nhật thường xuyên, nhiều thay đổi gây lỗi nếu không nâng cấp kịp hoặc kiểm thử kỹ. [5][9]

2.2.5 Ứng dụng của Node.js:

Node.js được sử dụng rộng rãi trong nhiều loại ứng dụng web và server do khả năng xử lý bất đồng bộ, hiệu suất cao và hệ sinh thái phong phú của nó. Dưới đây là một số ứng dụng phổ biến của Node.js:

Ứng dụng Web Thời Gian Thực (Real-time Web Applications): Node.js là lựa chọn lý tưởng cho các ứng dụng web thời gian thực như trò chuyện trực tuyến và trò chơi trực tuyến do khả năng xử lý các sự kiện I/O một cách nhanh chóng và hiệu quả.

APIs Server-side: Node.js thường được sử dụng để xây dựng RESTful APIs do khả năng xử lý đồng thời lớn và tốc độ phản hồi nhanh, làm cơ sở cho các ứng dụng di động và web.

Streaming Data: Node.js hỗ trợ xử lý dữ liệu dạng stream, cho phép ứng dụng xử lý các tệp video, âm thanh hoặc các dữ liệu khác trong khi chúng vẫn đang được truyền, thay vì phải chờ cho đến khi toàn bộ tệp được tải về.

Ứng dụng Một Trang (Single Page Applications): Node.js phù hợp với việc phát triển các ứng dụng một trang (SPA - Thông báo dịch vụ công cộng) như Gmail, Google Maps, hay Facebook, nơi mà nhiều tương tác xảy ra trên một trang duy nhất mà không cần tải lại trang.

Công cụ và Tự Động Hóa: Node.js cũng được sử dụng để phát triển các công cụ dòng lệnh và các script tự động hóa quy trình làm việc, nhờ vào các gói NPM hỗ trợ đa dạng và khả năng tích hợp dễ dàng với các công nghệ khác.

Microservices Architecture: Node.js là một lựa chọn phổ biến cho kiến trúc microservices, nơi các ứng dụng lớn được chia thành các dịch vụ nhỏ, độc lập và dễ quản lý hơn.

Ứng dụng IoT: Node.js thường được sử dụng trong các ứng dụng IoT, nơi cần xử lý một lượng lớn các kết nối đồng thời và các sự kiện từ các thiết bị IoT.

Dashboard và Monitoring: Node.js được sử dụng để xây dựng các dashboard hiển thị dữ liệu thời gian thực và các công cụ giám sát, giúp doanh nghiệp dễ dàng theo dõi hiệu suất và tình trạng của các hệ thống.

2.3 Tổng quan về ReactJs và React Vite:

2.3.1 Giới thiệu ReactJs và React Vite:

React, về bản chất, là một thư viện tập trung vào việc xây dựng các thành phần UI. Thay vì thao tác trực tiếp với DOM của trình duyệt, React sử dụng một "Virtual DOM" (DOM ảo). Khi có sự thay đổi dữ liệu, React sẽ so sánh Virtual DOM hiện tại với Virtual DOM mới, và chỉ cập nhật những phần thực sự khác biệt lên DOM thực. Cơ chế này giúp tối ưu hóa hiệu năng ứng dụng, giảm thiểu số lượng thao tác DOM tốn kém và mang lại trải nghiệm người dùng mượt mà hơn.^{[1][8]}

Một trong những điểm mạnh của React là tính "component-based". Ứng dụng React được xây dựng từ các thành phần độc lập, có thể tái sử dụng ở nhiều nơi khác nhau. Mỗi thành phần có thể chứa logic riêng, dữ liệu riêng và giao diện riêng. Cách tiếp cận này giúp đơn giản hóa việc quản lý mã nguồn, tăng tính bảo trì và khả năng mở rộng của ứng dụng.^{[1][2]}

JSX (JavaScript XML) là một phần không thể thiếu của React. Đây là một cú pháp cho phép viết mã HTML trực tiếp trong JavaScript. JSX giúp mã nguồn trở nên dễ đọc và dễ hiểu hơn, đồng thời cho phép các nhà phát triển tận dụng sức mạnh của JavaScript để tạo ra các giao diện người dùng động và linh hoạt.^{[1][3]}

Tuy nhiên, việc thiết lập một dự án React truyền thống có thể gặp phải một số thách thức. Các công cụ như Create React App (CRA) đã giúp đơn giản hóa quá trình này, nhưng vẫn còn một số hạn chế. CRA sử dụng Webpack để đóng gói (bundle) tất cả các module JavaScript thành một hoặc một vài file lớn. Quá trình này có thể tốn thời gian, đặc biệt đối với các ứng dụng lớn. Ngoài ra, CRA cũng khá "cứng nhắc" trong việc tùy chỉnh cấu hình bên dưới.

React Vite, được xây dựng dựa trên Vite, một công cụ xây dựng thế hệ mới, giải quyết những hạn chế này. Vite sử dụng cách tiếp cận "ES modules trực tiếp" (native ES modules). Thay vì đóng gói toàn bộ ứng dụng trước khi phục vụ, Vite cho phép trình duyệt tải các module JavaScript riêng lẻ khi chúng được yêu cầu. Điều này giúp giảm đáng kể thời gian khởi động và tải lại trang.^{[15][16]}

Một trong những ưu điểm lớn nhất của React Vite là tốc độ phát triển. Với Vite, thời gian khởi động một dự án React mới chỉ mất vài giây, so với vài phút với CRA. Khả năng HMR của Vite cũng cực kỳ nhanh chóng. Khi bạn thay đổi mã nguồn, Vite chỉ cập nhật các module bị ảnh hưởng mà không cần tải lại toàn bộ trang. Điều này giúp tiết kiệm thời gian và duy trì trạng thái ứng dụng, mang lại trải nghiệm phát triển liền mạch hơn.^[15]

React Vite cũng rất linh hoạt trong việc tùy chỉnh. Bạn có thể dễ dàng thêm các plugin, cấu hình các tùy chọn và tích hợp các công cụ khác theo nhu cầu của dự án. Điều này giúp bạn tạo ra một môi trường phát triển phù hợp với yêu cầu cụ thể của mình.^[8]

Ngoài ra, React Vite cũng hỗ trợ TypeScript một cách tuyệt vời. TypeScript là một ngôn ngữ siêu tập của JavaScript, bổ sung các tính năng kiểu tĩnh. Việc sử dụng TypeScript giúp phát hiện lỗi sớm hơn, cải thiện khả năng bảo trì mã nguồn và tăng tính tin cậy của ứng dụng.

Tóm lại, React và React Vite là một sự kết hợp mạnh mẽ để xây dựng các ứng dụng web hiện đại. React cung cấp một cách tiếp cận hiệu quả và linh hoạt để xây dựng giao diện người dùng, trong khi React Vite mang đến trải nghiệm phát triển nhanh chóng, linh hoạt và dễ dàng tùy chỉnh. Với sự kết hợp này, các nhà phát triển có thể tập trung vào việc xây dựng các tính năng tuyệt vời cho người dùng của họ, thay vì phải lo lắng về các vấn đề cấu hình và hiệu năng.

2.3.2 Cấu trúc của ReactJs:

React có cấu trúc rõ ràng, tuy nhiên không bắt buộc như các framework toàn diện như Angular,... React hoạt động theo triết lý "library over framework", tức là nó cung cấp các công cụ và nguyên tắc, nhưng người dùng được toàn quyền quyết định cách tổ chức mã nguồn. Trong thực tế phát triển, để đảm bảo tính bảo trì, mở rộng và dễ quản lý, các dự án React thường vẫn tuân theo những cấu trúc chuẩn:

src: Là thư mục cốt lõi chứa toàn bộ mã nguồn ứng dụng.

components/: Bao gồm các thành phần UI nhỏ và có khả năng tái sử dụng cao, ví dụ như Button, Input, Modal.

pages: Đại diện cho các trang chính trong ứng dụng, mỗi trang có thể là một điểm định tuyến trong hệ thống.

layouts: Định nghĩa cấu trúc bố cục tổng thể như thanh điều hướng, footer, hoặc sidebar được chia sẻ giữa nhiều trang.

hooks: Nơi lưu trữ các custom hook – những hàm có thể tái sử dụng logic với cú pháp `useSomething`, ví dụ `useDarkMode`, `useForm`.

contexts: Sử dụng React Context API để quản lý trạng thái chia sẻ giữa các component như Theme, Authentication, hoặc Language.

services: Bao gồm logic tương tác với hệ thống bên ngoài như API REST hoặc GraphQL, thường dùng thư viện Axios hoặc Fetch.

types: Chứa các định nghĩa kiểu dữ liệu nếu dự án sử dụng TypeScript, giúp kiểm tra tĩnh và phát hiện lỗi sớm.

App.jsx: Là component gốc của toàn bộ ứng dụng, thường chứa định tuyến chính và layout bao quanh.

main.jsx: Là điểm vào (entry point), nơi React được khởi tạo và render vào DOM thực qua lệnh `ReactDOM.createRoot`.

Khác với Angular – nơi kiến trúc ứng dụng được quy định nghiêm ngặt bởi framework – React chỉ là một thư viện giao diện nên nhà phát triển có quyền linh hoạt hoàn toàn trong việc định hình cấu trúc. Mức độ linh hoạt này tuy giúp React phù hợp với nhiều loại dự án, nhưng đồng thời cũng đặt ra yêu cầu về kỷ luật tổ chức trong nhóm phát triển.^{[1][2][8]}

2.3.3 Cách cài đặt và tạo dự án ReactJs:

Khởi tạo bằng Create React App (CRA):

- Create React App là một công cụ CLI chính thức do nhóm phát triển React duy trì. CRA giúp thiết lập một môi trường phát triển React chuẩn hóa với đầy đủ các công cụ cần thiết như Babel, Webpack, ESLint và hệ thống kiểm thử. Các bước thực hiện:

+ Bước 1: Cài đặt Node.js và npm. Trước tiên, cần cài đặt Node.js (kèm npm) từ trang chính thức <https://nodejs.org>. Sau khi cài đặt có thể kiểm tra phiên bản bằng cách chạy hai câu lệnh sau: *node -v*, *npm -v*.

+ Bước 2: Tạo dự án với CRA. Sử dụng lệnh sau để tạo dự án: *npx create-react-app my-app*. CRA sẽ tự động tạo cấu trúc thư mục, cài đặt các gói phụ thuộc và cấu hình sẵn Webpack, Babel, v.v.

+ Bước 3: Chạy dự án bằng cách di chuyển vào thư mục chứa dự án và chạy câu lệnh: *npm start*. Ứng dụng React mặc định sẽ chạy tại địa chỉ *http://localhost:3000/*.

- Ưu điểm của CRA:

+ Thiết lập nhanh chóng, không cần cấu hình thủ công.

+ Tích hợp sẵn testing, linting, và hot-reloading.

- Nhược điểm: Tùy chỉnh cấu hình bị giới hạn nếu không sử dụng eject (dẫn đến khó khăn khi mở rộng).

Khởi tạo bằng Vite:

- Vite là công cụ build thế hệ mới, do Evan You (tác giả Vue.js) phát triển, cung cấp trải nghiệm phát triển cực kỳ nhanh nhờ tận dụng ES Modules và HMR. Vite đang trở thành lựa chọn phổ biến thay thế CRA trong các dự án hiện đại. Các bước thực hiện:

+ Bước 1: Cài đặt Vite (qua npm hoặc yarn)

+ Bước 2: Tạo dự án mới bằng câu lệnh *npm create vite@latest my-vite-app*. Chọn framework là React khi được hỏi.

+ Bước 3: Cài đặt phụ thuộc. Sau khi tạo xong, di chuyển vào thư mục và chạy câu lệnh *cd my-vite-app*, *npm install*.

+ Bước 4: Chạy ứng dụng bằng câu lệnh *npm run dev*. Ứng dụng sẽ được chạy tại địa chỉ *http://localhost:5173/*.

- Ưu điểm của Vite:

- + Tốc độ khởi động nhanh nhờ sử dụng ES Modules.
- + Quá trình cập nhật mã cực kỳ nhanh (HMR mượt hơn).
- + Cấu trúc linh hoạt, dễ cấu hình và mở rộng.
- Nhược điểm:
 - + Chưa phổ biến bằng CRA ở một số môi trường doanh nghiệp cũ.
 - + Cần hiểu rõ hơn về cơ chế hoạt động nếu muốn cấu hình nâng cao. Tổng quan về Tailwind CSS. ^{[1][2][11]}

2.3.4 Giới thiệu Tailwind CSS:

Tailwind CSS là một framework CSS tiện ích (utility-first CSS framework) được thiết kế để giúp việc xây dựng giao diện người dùng trở nên nhanh chóng và hiệu quả. Thay vì định nghĩa các lớp CSS tùy chỉnh cho từng thành phần, Tailwind cung cấp một tập hợp các lớp được định nghĩa sẵn mô tả các thuộc tính CSS thường dùng như margin, padding, màu sắc, typography, flexbox, grid, v.v. Cách tiếp cận này cho phép các lập trình viên trực tiếp xây dựng giao diện thông qua các lớp tiện ích được áp dụng ngay trong phần tử HTML.

Ra mắt lần đầu vào năm 2017 bởi Adam Wathan, Tailwind CSS nhanh chóng thu hút được sự quan tâm rộng rãi nhờ triết lý thiết kế linh hoạt, dễ mở rộng và khả năng tùy chỉnh mạnh mẽ. Với hệ thống cấu hình dựa trên file `tailwind.config.js`, người dùng có thể dễ dàng điều chỉnh theme, breakpoint, spacing, font, hoặc thậm chí tạo ra các lớp tiện ích tùy chỉnh theo nhu cầu của dự án.



Hình 3. Giới thiệu Tailwind CSS

Một điểm nổi bật khác của Tailwind CSS là khả năng tích hợp tốt với các công cụ build hiện đại như PostCSS, Webpack, Vite hay các framework JavaScript như React, Vue, và Next.js. Ngoài ra, phiên bản mới nhất của Tailwind cũng hỗ trợ tính năng Just-in-Time (JIT) – cho phép sinh lớp CSS theo nhu cầu tại thời điểm biên dịch, giúp giảm kích thước file CSS và tăng hiệu năng tổng thể.

Tailwind không chỉ là một công cụ CSS, mà còn là một triết lý phát triển UI giúp các nhóm phát triển frontend tăng tốc độ làm việc và giảm thiểu sự phụ thuộc vào tệp stylesheet truyền thống. Với cộng đồng phát triển lớn và hệ sinh thái đa dạng (như Tailwind UI, Headless UI), framework này ngày càng được ưa chuộng trong các dự án thiết kế hiện đại.^{[17][12]}

2.3.5 Ưu điểm của Tailwin CSS:

Tùy biến cao, hiệu suất cao. Không giống như Bootstrap cung cấp những class gần như đồng gọi sẵn, chỉ cần gọi ra là dùng. Tailwind giúp bạn định nghĩa những phần phù hợp với dự án của bạn mà không bị gò bó.

Cho phép xây dựng responsive layout phức tạp.

Responsive và phát triển dễ dàng.

Tạo thành phần dễ dàng.

Hỗ trợ cài đặt với nhiều framework front-end khác như react, vuejs,...^[12]

2.3.6 Nhược điểm của Tailwin CSS:

Thiếu tiêu đề và thành phần điều hướng (navigation).

Cần có thời gian để làm quen, tìm hiểu và nhớ tên các class.

Có kiến thức về CSS thì mới sử dụng tốt được.^[12]

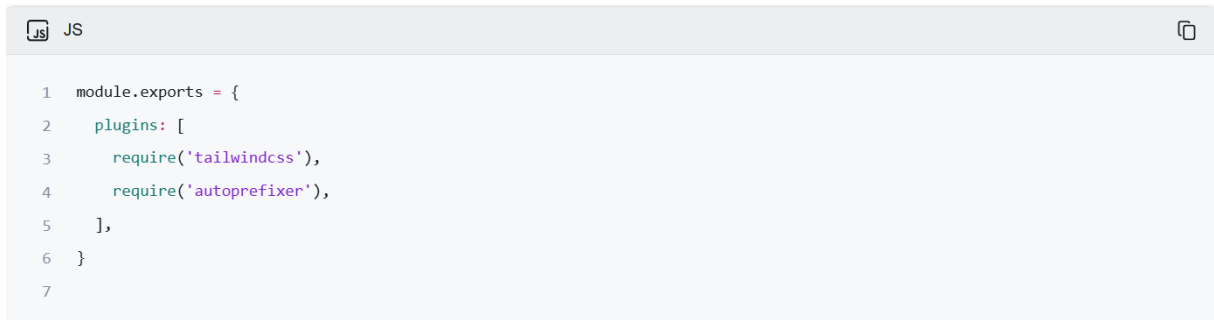
2.3.7 Cài đặt và sử dụng Tailwin CSS:

Hướng dẫn cài đặt Tailwind CSS:

Sử dụng lệnh sau để cài đặt: *npm install tailwindcss postcss autoprefixer* hoặc *yarn add tailwindcss postcss autoprefixer* (nếu dùng yarn)

Tạo tệp cấu hình: *npx tailwindcss init*

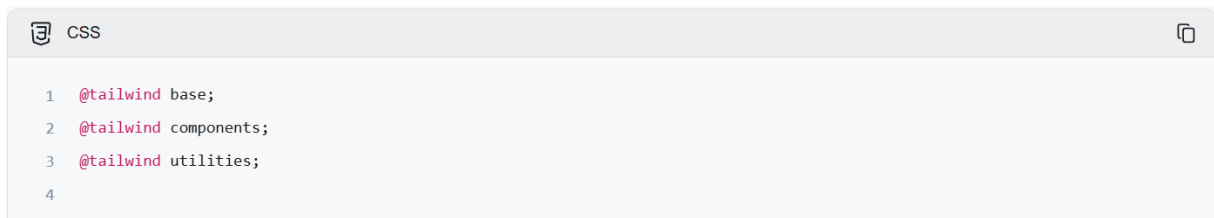
Tạo một *postcss.config.js* file trong thư mục gốc của thư mục dự án và thêm nội dung sau:

A screenshot of a code editor window titled 'JS'. The code is as follows:

```
1 module.exports = {
2   plugins: [
3     require('tailwindcss'),
4     require('autoprefixer'),
5   ],
6 }
7
```

Hình 4. Nội dung tệp *postcss.config.js*

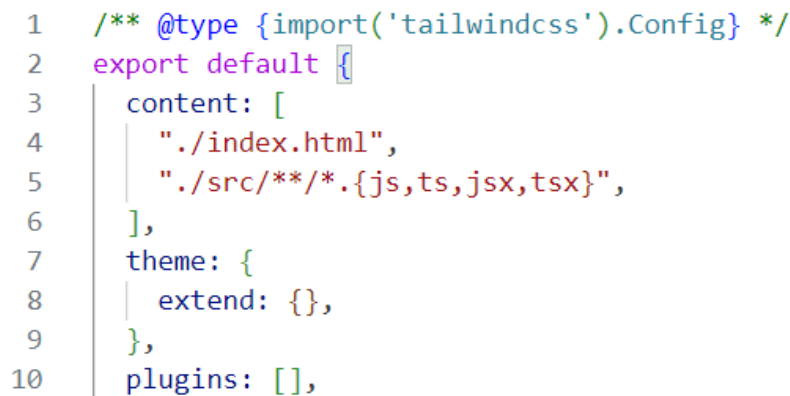
Mở thư mục *index.css* trong *src* và thêm nội dung sau:

A screenshot of a code editor window titled 'CSS'. The code is as follows:

```
1 @tailwind base;
2 @tailwind components;
3 @tailwind utilities;
4
```

Hình 5. Nội dung tệp *styles.css*

Mở file *tailwind.config.js* được tự động thêm vào thư mục gốc của dự án khi cài đặt tailwind và thay toàn bộ nội dung của file thành:

A screenshot of a code editor showing the updated content of *tailwind.config.js*. The code is as follows:

```
1 /** @type {import('tailwindcss').Config} */
2 export default {
3   content: [
4     './index.html',
5     './src/**/*..{js,ts,jsx,tsx}',
6   ],
7   theme: {
8     extend: {},
9   },
10  plugins: [],
11 }
```

Hình 6. Thay đổi nội dung của file *tailwind.config.js*

Khởi động lại dự án: *npm run dev* để áp dụng thay đổi và sử dụng tailwind.

2.4 Tổng quan về MongoDB:

2.4.1 Sơ lược về NoSQL:

NoSQL (Not Only SQL) là một nhóm các hệ quản trị cơ sở dữ liệu phi quan hệ (non-relational database management systems), được thiết kế để xử lý và lưu trữ lượng lớn dữ liệu phi cấu trúc, bán cấu trúc hoặc dữ liệu có cấu trúc linh hoạt. Khác với các hệ cơ sở dữ liệu quan hệ truyền thống (RDBMS) dựa trên mô hình bảng và lược đồ cố định (schema-based), NoSQL cho phép lưu trữ dữ liệu theo nhiều mô hình khác nhau như dạng tài liệu (document), cặp khóa–giá trị (key–value), cột (column-family), hoặc đồ thị (graph).

Sự ra đời của NoSQL xuất phát từ nhu cầu mở rộng quy mô hệ thống và xử lý dữ liệu lớn trong thời đại web 2.0 và điện toán đám mây, đặc biệt trong các ứng dụng yêu cầu khả năng xử lý song song cao, tính mở rộng ngang (horizontal scaling) và khả năng đáp ứng theo thời gian thực. Nhiều hệ thống NoSQL được xây dựng với mục tiêu hỗ trợ mô hình phân tán, cho phép lưu trữ dữ liệu trên nhiều máy chủ khác nhau mà vẫn đảm bảo tính toàn vẹn và hiệu năng.^{[14][13]}

2.4.2 Giới thiệu MongoDB:

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, được phát triển bởi công ty MongoDB Inc. (trước đây là 10gen), lần đầu ra mắt vào năm 2009. MongoDB thuộc nhóm cơ sở dữ liệu theo mô hình tài liệu (document-oriented database), trong đó dữ liệu được lưu trữ dưới dạng các tài liệu BSON (Binary JSON), cho phép biểu diễn các cấu trúc dữ liệu phức tạp như mảng, đối tượng lồng nhau, và các kiểu dữ liệu khác một cách tự nhiên và linh hoạt.

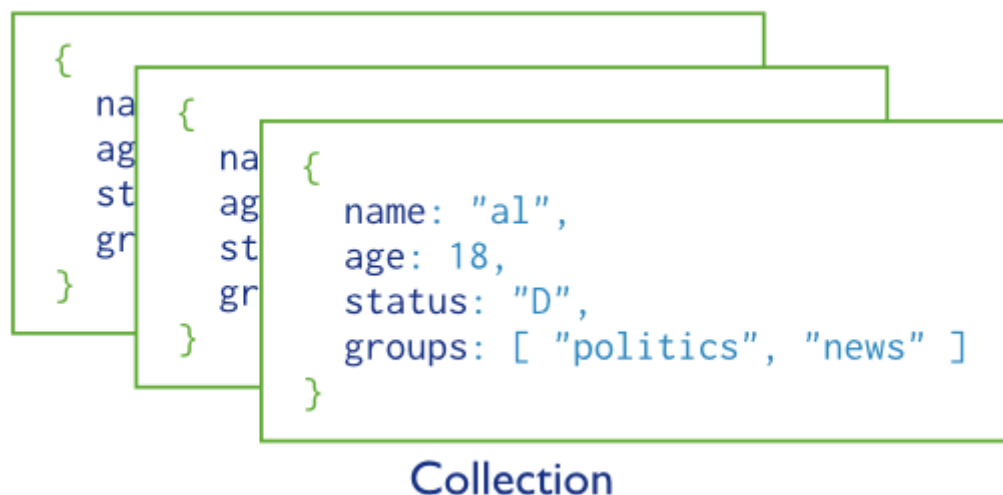
Một trong những điểm nổi bật của MongoDB là khả năng mở rộng theo chiều ngang (horizontal scalability) thông qua cơ chế phân mảnh dữ liệu (sharding), cùng với khả năng sao chép (replication) để đảm bảo tính sẵn sàng cao và chống mất dữ liệu. MongoDB được thiết kế để hoạt động hiệu quả trong các hệ thống phân tán, đồng thời hỗ trợ truy vấn động, lập chỉ mục linh hoạt và tích hợp dễ dàng với nhiều ngôn ngữ lập trình thông qua các driver chính thức.

Cơ sở dữ liệu MongoDB không yêu cầu một schema cố định. Các tài liệu trong cùng một collection có thể có cấu trúc khác nhau, điều này rất phù hợp với các ứng dụng có yêu cầu thay đổi cấu trúc dữ liệu thường xuyên hoặc có dữ liệu đến từ nhiều nguồn khác nhau. Chính sự linh hoạt này đã khiến MongoDB trở thành một trong những lựa chọn hàng đầu trong các ứng dụng web hiện đại, đặc biệt là các ứng dụng được xây dựng theo kiến trúc Microservices hoặc sử dụng Node.js.^{[4][10]}

2.4.3 Một số khái niệm trong MongoDB:

Database là một container vật lý cho các collection. Mỗi DB được thiết lập cho riêng nó một danh sách các files hệ thống files. Một server MongoDB đơn thường có nhiều DB.

Collection là một nhóm các documents của MongoDB. Nó tương đương với một table trong RDBMS. Một Collection tồn tại trong một cơ sở dữ liệu duy nhất. Các collection ko tạo nên một schema. Documents trong collection có thể có các fields khác nhau. Thông thường, tất cả các documents trong collections có mục đích khá giống nhau hoặc liên quan tới nhau.



Hình 7. Ví dụ về Collection trong MongoDB

Document là một tập hợp các cặp key-value. Documents có schema động. Schema động có nghĩa là documents trong cùng một collection không cần phải có cùng một nhóm các fields hay cấu trúc giống nhau và các fields phổ biến trong các documents của collection có thể chứa các loại dữ liệu khác nhau.

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```



A diagram illustrating the structure of a MongoDB document. It shows a JSON-like object with four fields: 'name', 'age', 'status', and 'groups'. To the right of each field, there is a horizontal arrow pointing left towards the field name, followed by the text 'field: value'. This indicates that each field in the document represents a key-value pair.

Hình 8. Ví dụ minh họa cho Document trong MongoDB

2.4.4 Ưu điểm của MongoDB:

Đầu tiên có thể nhắc đến là tính linh hoạt lưu trữ dữ liệu theo các kích cỡ khác nhau, dữ liệu dưới dạng hướng tài liệu JSON nên có thể chèn vào thoải mái bất cứ thông tin mong muốn.

Khác với RDBMS, dữ liệu trong đây không có sự ràng buộc và không có yêu cầu tuân theo khuôn khổ nhất định, điều này giúp tiết kiệm thời gian cho việc kiểm tra sự thỏa mãn về cấu trúc nếu muốn chèn, xóa, cập nhật hay thay đổi các dữ liệu trong bảng.

MongoDB dễ dàng mở rộng hệ thống bằng cách thêm node vào cluster (cụm các node chứa dữ liệu giao tiếp với nhau).

Ưu điểm thứ tư là tốc độ truy vấn nhanh hơn nhiều so với hệ quản trị cơ sở dữ liệu quan hệ RDBMS do dữ liệu truy vấn được cached lên bộ nhớ RAM để lượt truy vấn sau diễn ra nhanh hơn mà không cần đọc từ ổ cứng.

Cũng là một ưu điểm về hiệu suất truy vấn của MongoDB, trường dữ liệu "_id" luôn được tự động đánh chỉ mục để đạt hiệu suất cao nhất.

2.4.5 Nhược điểm của MongoDB

Dữ liệu trong MongoDB không bị ràng buộc như RDBMS nhưng người sử dụng lưu ý cẩn thận mọi thao tác để không xảy ra các kết quả ngoài ý muốn gây ảnh hưởng đến dữ liệu.

Một nhược điểm mà ”dân công nghệ” hay lo ngại là bộ nhớ của thiết bị. Chương trình này thường tốn bộ nhớ do dữ liệu được lưu dưới dạng key-value, trong khi các collection chỉ khác về value nên sẽ lặp lại key dẫn đến thừa dữ liệu.

Thông thường, dữ liệu thay đổi từ RAM xuống ổ cứng phải qua 60 giây thì chương trình mới thực hiện hoàn tất, đây là nguy cơ bị mất dữ liệu nếu bất ngờ xảy ra tình huống mất điện trong vòng 60 giây đó.

CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Phân tích và đặt tả yêu cầu hệ thống:

3.1.1 Phân tích chức năng:

Đối với người dùng (khách hàng):

Đăng nhập, đăng xuất khỏi hệ thống: Đăng nhập bằng số điện thoại và OTP để xác nhận tính khả dụng của số điện thoại, đăng xuất khỏi hệ thống khi không sử dụng.

Đặt hẹn khám bệnh: Khách hàng nhập đầy đủ thông tin cá nhân cơ bản, chọn các lựa chọn và đặt hẹn.

Lịch khám chữa: Cho phép chọn tuần, mỗi tuần sẽ hiển thị các ngày trong tuần và bác sĩ sẽ làm việc trong ngày.

Thông tin bác sĩ: Hiển thị thông tin bác sĩ khi đặt hẹn và khi kiểm tra lịch khám chữa.

Thông tin phòng khám: Hiển thị thông tin phòng khám tại trang chủ.

Theo dõi lịch hẹn: Hiển thị tất cả lịch hẹn của khách hàng.

Đơn hẹn: Hiển thị đầy đủ thông tin khách hàng điền vào đơn đăng ký hẹn khám, trạng thái xác nhận của đơn.

Đối với người quản trị:

Quản lý bác sĩ: Xem danh sách tài khoản và thông tin của bác sĩ, thêm hoặc xóa tài khoản nếu cần.

Quản lý danh sách đơn hẹn: Xem danh sách, phản hồi trạng thái xác nhận của đơn.

Quản lý thông tin phòng khám: Chỉnh sửa thông tin của phòng khám hiển thị tại trang chủ của khách hàng.

Thông kê đơn đặt: Xem báo cáo đơn đặt qua biểu đồ.

3.1.2 Yêu cầu phi chức năng

Khả năng mở rộng: Dễ nâng cấp, thêm tính năng mới trong tương lai.

Bảo mật: Sử dụng JWT để xác thực, phân quyền rõ ràng giữa user và admin.

Hiệu năng: Giao diện tải nhanh, truy vấn dữ liệu tối ưu.

Dễ sử dụng: Giao diện thân thiện với khách hàng và quản trị viên.

3.2 Thiết kế hệ thống

3.2.1 Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu gồm có các collection chính như sau:

1. Collection “users”: (Lưu trữ thông tin người dùng gồm: số điện thoại)

```
“users”: {  
  
  phone: { type: String, required: true },  
  
  email: { type: String, required: false, unique: true },  
  
  role: { type: Number, default: 9, require: true },  
  
  cartData: { type: Object, default: { } }  
  
}
```

2. Collection “otp”: (Lưu trữ số điện thoại và otp đã được mã hóa dùng để xác thực tính khả dụng của số điện thoại khi đăng nhập)

```
“otp”: {  
  
  phone: { type: String, required: true },  
  
  otp: { type: String, required: true },  
  
  createdAt: { type: Date, default: Date.now, expires: 60 }  
  
}
```

3. Collection “doctor”: (Lưu trữ thông tin tài khoản cung cấp cho bác sĩ và thông tin cá nhân của bác sĩ)

```
“doctors”: {  
  
  username: { type: String, required: true, unique: true },  
  
  password: { type: String, required: true },
```



```

    role: { type: Number, default: 0, require: true, max: 9 },
    image: { type: String, required: false },
    name: { type: String, required: true },
    phone: { type: String, required: true },
    email: { type: String, required: false, unique: true },
    description: { type: String, required: false },
    workDay: { type: Array, require: false },
    cartData: { type: Object, default: {} }
  }

```

4. Collection “clinic”: (Lưu trữ thông tin phòng khám sẽ hiển thị ở trang chủ người dùng)

```

“clinic”: {
  code: { type: String, required: true, unique: true },
  name: { type: String, required: true },
  phone: { type: String, required: true },
  open: { type: String, required: false },
  close: { type: String, required: false }
}

```

5. Collection “booking”: (Lưu trữ thông tin đơn hẹn của khách hàng)

```

“booking”: {
  userId: { type: String, required: true },
  name: { type: String, required: true },
  phone: { type: String, required: true },
  session: { type: String, required: true },
  note: { type: String, required: false },
}

```

doc: { type: String, required: true },

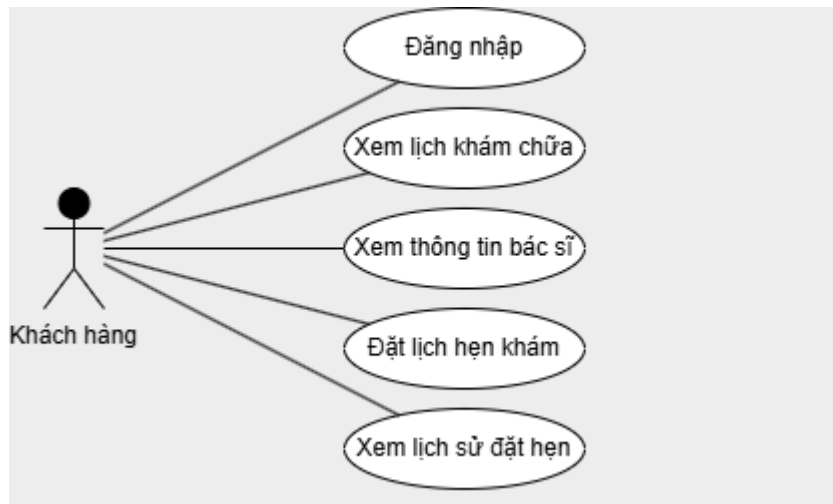
status: { type: Boolean, required: true, default: false },

date: { type: Date, required: true },

datebook: { type: Date, default: Date.now } }

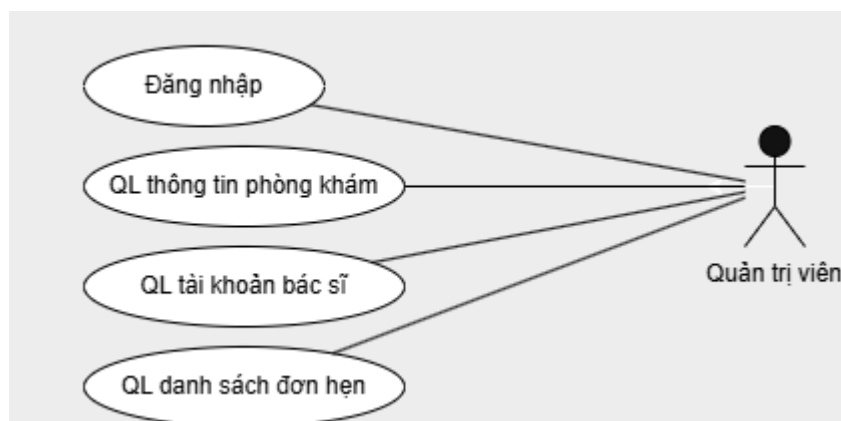
3.2.2 Sơ đồ Use Case:

Đối với khách hàng:



Hình 9. Sơ đồ usecase đối với khách hàng

Đối với quản trị viên:



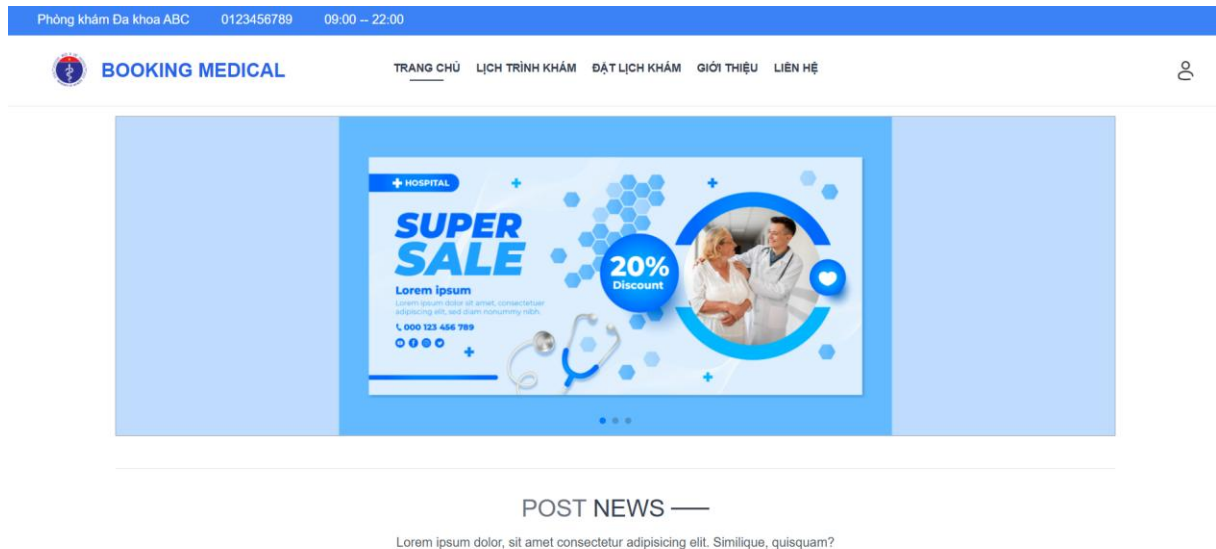
Hình 10. Sơ đồ usecase đối với quản trị viên

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

4.1 Giao diện khách hàng:

4.1.1 Trang chủ:

Trang chủ được thiết kế đơn giản, hiển thị thông tin về phòng khám.




Hình 11. Giao diện khách hàng: Trang chủ


4.1.2 Trang đăng nhập:

Trang đăng nhập đơn giản phù hợp với mọi đối tượng khách hàng từ trẻ đến cao tuổi.

Phòng khám Đa khoa ABC
0123456789
09:00 – 22:00



BOOKING MEDICAL

[TRANG CHỦ](#)
[LỊCH TRÌNH KHÁM](#)
[ĐẶT LỊCH KHÁM](#)
[GIỚI THIỆU](#)
[LIÊN HỆ](#)



Customer Login —

Bác sĩ đăng nhập


BOOKING MEDICAL

[Home](#)
[About](#)

CLINIC
[Home](#)
[About](#)

GET IN TOUCH
 +84 842-088-945
bookingmedical@gmail.com


Lorem ipsum dolor sit amet consectetur adipiscing elit. Voluptates, tempora.

Hình 12. Trang đăng nhập


4.1.3 Trang đặt hẹn khám:

Trang đặt hẹn hiển thị rõ các các lưu ý, nhắc nhở và thông tin bác sĩ muốn đặt hẹn.

Phòng khám Đa khoa ABC
0123456789
09:00 – 22:00



BOOKING MEDICAL

[TRANG CHỦ](#)
[LỊCH TRÌNH KHÁM](#)
[ĐẶT LỊCH KHÁM](#)
[GIỚI THIỆU](#)
[LIÊN HỆ](#)



PHIẾU ĐĂNG KÝ —

THÔNG TIN BÁC SĨ —



Bs. NGUYỄN MAI DUY KHOA
Giới thiệu về bác sĩ:
 Tôi là một bác sĩ đã được đào tạo bài bản, có hơn 5 năm kinh nghiệm làm việc trong lĩnh vực Nội khoa. Trong suốt quá trình công tác, tôi luôn nỗ lực không ngừng để mang đến chất lượng khám chữa bệnh tốt nhất cho bệnh nhân.

Lưu ý khi đặt lịch


- Vui lòng nhập đầy đủ và chính xác thông tin.
- Chọn đúng ngày, bác sĩ phù hợp với nhu cầu.
- Nhân viên sẽ liên hệ xác nhận sau khi đặt lịch.
- Đến trước giờ hẹn 10 phút để làm thủ tục.

Hình 13. Trang đặt hẹn khám

4.1.4 Trang lịch khám chữa:

Trang lịch khám chữa hiển thị một bảng biểu về lịch khám chữa của các bác sĩ trong tuần.

Phòng khám Đa khoa ABC012345678909:00 – 22:00



BOOKING MEDICAL


TRANG CHỦLỊCH TRÌNH KHÁMLỊCH TRÌNH

Hình 14 Trang lịch khám chữa


4.1.5 Trang lịch sử đặt hẹn:

Hiện thị một danh sách các đơn đã đặt hẹn và cho biết rõ trạng thái của đơn (đã xác nhận, chưa xác nhận, đã đến khám)


Phòng khám Đa khoa ABC012345678909:00 -- 22:00

BOOKING MEDICAL

TRANG CHỦLỊCH TRÌNH KHÁMĐẶT LỊCH KHÁMGỚI THIỆULIÊN HỆ



MY BOOKING



Nguyễn Mai Duy Khoa

Số điện thoại: 0842088945


Ngày hẹn khám: 13/06/2025

Sáng

Ngày đặt hẹn: 12/06/2025


Bác sĩ: NGUYỄN MAI DUY KHOA

Chờ xác nhận



Track Booking

HISTORY BOOKING

BOOKING MEDICAL

Lorem ipsum dolor sit amet consectetur adipiscing elit. Voluptates, tempora.

CLINIC

Home
About
Contact

GET IN TOUCH

+84 842-088-945
bookingmedical@gmail.com

Hình 15. Trang lịch sử đặt hẹn

4.1.6 Trang cá nhân:

Trang cá nhân chỉ áp dụng đối với tài khoản nhân sự của phòng khám và hiển thị thông tin cá nhân của người được cấp tài khoản.

Phòng khám Đa khoa ABC
0123456789
09:00 – 22:00

BOOKING MEDICAL
TRANG CHỦ
LỊCH TRÌNH KHÁM
ĐẶT LỊCH KHÁM
GIỚI THIỆU
LIÊN HỆ

Trang cá nhân
Danh sách đặt hẹn

THÔNG TIN CÁ NHÂN
Bs. NGUYỄN MAI DUY KHOA
Liên hệ:
SĐT: 0842068945 **Email:** nguyenkoahmt@gmail.com
Lịch làm việc:

	T2	T3	T4	T5	T6	T7	CN
Sáng	X	X	X				X
Chiều				X	X	X	

Giới thiệu
Tôi là một bác sĩ đã được đào tạo bài bản, có hơn 5 năm kinh nghiệm làm việc trong lĩnh vực Nội khoa. Trong suốt quá trình công tác, tôi luôn nỗ lực không ngừng để mang đến chất lượng khám chữa bệnh tốt nhất cho bệnh nhân.

Hình 16. Trang cá nhân

4.2 Giao diện quản trị:

4.2.1 Trang quản lý thông tin phòng khám:

Hiện thị thông tin phòng khám và form nhập để thay đổi thông tin.

BOOKING MEDICAL
Logout

Clinic
Article
Banner
Manage doctor
List Booking
Orders

Clinic code

Clinic name

Clinic phone

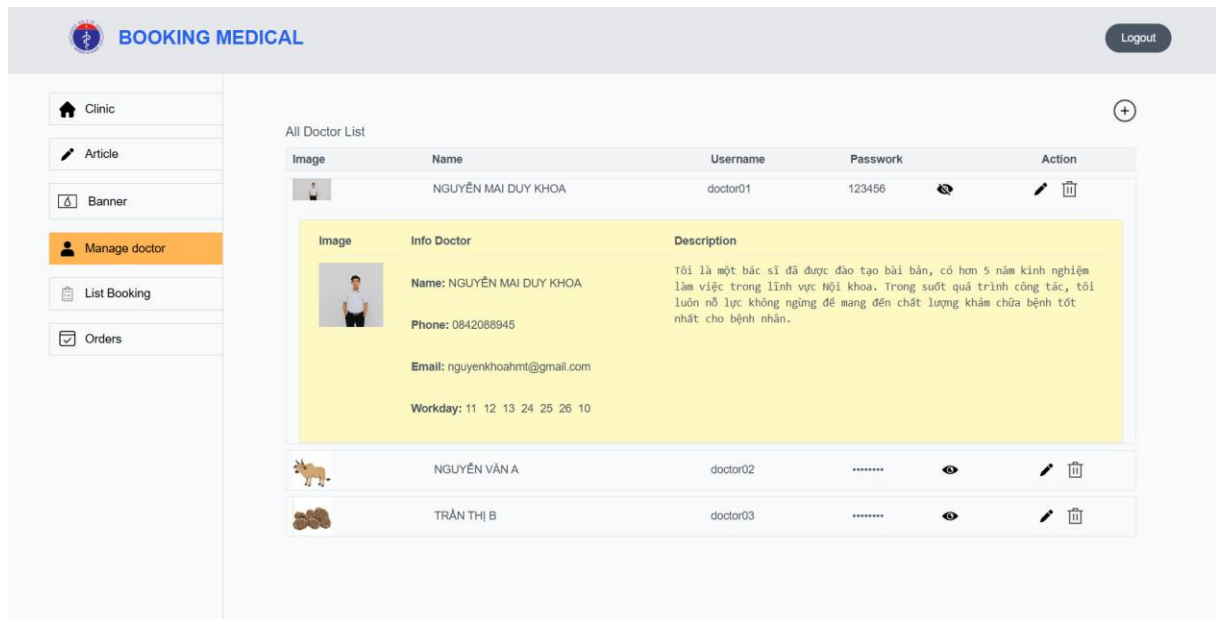
Opening hours
Open: -- Close:

Clinic code: ABC
Clinic name: Phòng khám Đa khoa ABC
Phone: 0123456789
Open: 09:00 -- Close: 22:00

Hình 17. Trang quản lý thông tin phòng khám

4.2.2 Trang quản lý tài khoản bác sĩ:

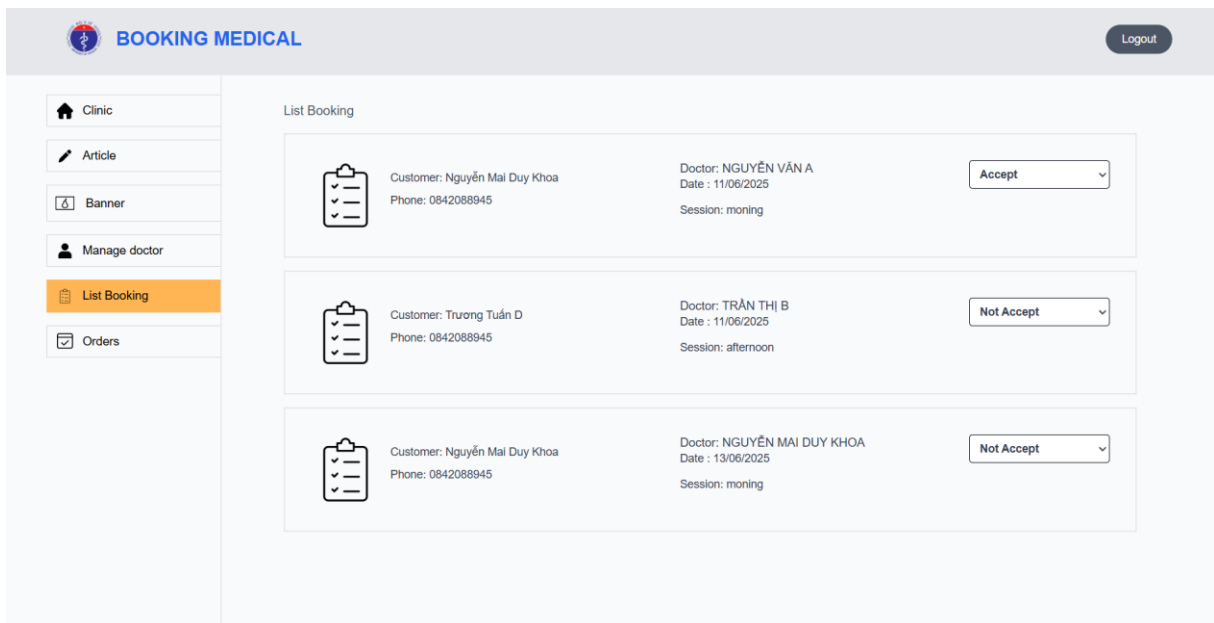
Hiện thị danh sách toàn bộ tài khoản đã cấp, cho phép xem chi tiết thông tin khi nhấp vào, cho phép sửa, xóa tài khoản.



Hình 18. Trang quản lý tài khoản bác sĩ

4.2.3 Trang quản lý đơn hẹn:

Hiện thị danh sách toàn bộ đơn hẹn của khách hàng, cho phép thay đổi và cập nhật trạng thái của đơn hẹn để phản hồi cho khách hàng.



Hình 19. Trang quản lý đơn hẹn

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận:

Về sản phẩm, em đã xây dựng được ứng dụng đặt hẹn khám bệnh cho phòng khám tư nhân. Hệ thống đáp ứng được các chức năng cốt lõi như đặt hẹn khám từ phía khách hàng và phản hồi đơn hẹn từ phía quản trị, một phần giúp quản lý nhân sự của phòng khám, góp phần làm tăng trải nghiệm người dùng khi đi khám bệnh.

Về kiến thức, quá trình triển khai giúp em hiểu sâu hơn và vận dụng thành thạo hơn các công nghệ hiện đại như ReactJS, Vite và NodeJs, thành thạo hơn trong việc sử dụng và làm việc với NoSQL và MongoDB. Qua đó, em cảm nhận được sự tiến bộ và sự cải thiện trên nhiều kỹ năng quan trọng trong quá trình thực hiện đề tài, phát triển tư duy trong phân tích và thiết kế hệ thống, tổ chức mã nguồn và kiểm thử tính năng. Những kiến thức này là nền tảng giúp em thành công trong dự án và là hành trang để bước tiếp vào một môi trường làm việc mới, một thị trường lao động đầy cạnh tranh và thách thức.

5.2 Hướng phát triển:

Trong thời gian tới, hệ thống có thể được mở rộng và nâng cấp với các tính năng hiện đại, kết hợp với các công nghệ mới làm tăng hiệu suất, luôn thu thập các ý kiến và phản hồi từ người dùng để cải thiện trải nghiệm người dùng. Một số thay đổi dự kiến như sau:

- Phát triển quy trình chuẩn đoán online giúp người dùng giảm chi phí đi lại, giảm tải cho các phòng khám.
- Hệ thống đánh giá bác sĩ giúp người dùng dễ dàng hơn trong việc lựa chọn bác sĩ phù hợp.
- Hệ thống chat box trực tiếp với bác sĩ thông qua ứng dụng giúp bác sĩ có thể theo dõi tình trạng của khách hàng sau khi đã thăm khám, khách hàng có thể kết nối với bác sĩ từ xa.
- Các tiện ích y tế tự hành khác như: dự đoán các bệnh có thể qua các thông số y tế có thể tự đo đạc tại nhà,...

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Accomazzo, A., Murray, N., Lerner, C., & Zepernick, T. (2017). *Fullstack React: The Complete Guide to ReactJS and Friends*. Fullstack.io.
- [2] Banks, A., & Porcello, E. (2020). *Learning React: Functional Web Development with React and Redux* (2nd ed.). O'Reilly Media.
- [3] Ceddia, D. (2017). *Pure React: A step-by-step guide to mastering React for beginners and experienced developers*.
- [4] Chodorow, K. (2019). *MongoDB: The definitive guide* (3rd ed.). O'Reilly Media.
- [5] Greg Lim (2019). *Beginning Node.js, Express & MongoDB Development*. Apress.
- [6] Leonard Richardson, Mike Amundsen & Sam Ruby (2013). What Is REST?, Your First RESTful Service. *RESTful Web APIs*, O'Reilly Media, Sebastopol, California.
- [7] JJ Geewax (2021). Introduction to APIs. *API Design Patterns*, Shelter Island, New York.
- [8] Mardan, A. (2017). *React Quickly: Painless web apps with React, JSX, Redux, and GraphQL*. Manning Publications.
- [9] Mario Casciaro & Luciano Mammino (2020). *Node.js Design Patterns* (3rd ed.). Packt Publishing.
- [10] MongoDB Inc. (2024). MongoDB Manual v7.0. <https://www.mongodb.com/docs/manual/>.
- [11] O'Reilly Media. (2022). Building React Applications with Vite. O'Reilly Publishing.
- [12] Pickering, B. (2022). Refactoring UI with Tailwind CSS. Tailwind Labs Press.
- [13] Redmond, E., & Wilson, J. R. (2012). Seven databases in seven weeks: A guide to modern databases and the NoSQL movement. Pragmatic Bookshelf.
- [14] Sadalage, P. J., & Fowler, M. (2013). NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Addison-Wesley.

- [15] Storybook. (2023). *Integrating Storybook with Vite and React*. <https://storybook.js.org/docs/react/builders/vite>.
- [16] Vite Documentation Team. (2024). *Vite Documentation*. <https://vitejs.dev>.
- [17] Wathan, A., & others. (2023). Tailwind CSS: Rapidly build modern websites without ever leaving your HTML (v3.3). Tailwind Labs. <https://tailwindcss.com/docs>.