

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

XÂY DỰNG WEBSITE BÁN THIẾT BỊ DI ĐỘNG BẰNG NODEJS

Giảng viên hướng dẫn:
ThS. Phạm Minh Dương

Sinh viên thực hiện:
Họ tên: Trần Khánh Duy
MSSV: 110121150
Lớp: DA21TTC

Trà Vinh, tháng 12 năm 2024

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ I, NĂM HỌC 2024-2025

XÂY DỰNG WEBSITE BÁN THIẾT BỊ DI ĐỘNG BẰNG NODEJS

Giảng viên hướng dẫn:
ThS. Phạm Minh Dương

Sinh viên thực hiện:
Họ tên: Trần Khánh Duy
MSSV: 110121150
Lớp: DA21TTC

Trà Vinh, tháng 12 năm 2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

[illegible]

Trà Vinh, ngày tháng năm

Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

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.

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết, tôi xin chân thành gửi lời cảm ơn đến quý thầy, cô trường Đại Học Trà Vinh nói chung và các thầy cô bộ môn trong khoa Kỹ thuật và Công nghệ nói riêng đã tạo điều kiện cho chúng tôi được tiếp xúc và thực hành, giúp chúng tôi định hướng và tránh những sai sót trong môi trường làm việc cho thời gian sắp tới.

Tôi xin được gửi lời cảm ơn chân thành và sâu sắc nhất đến người hướng dẫn tôi, Thầy Phạm Minh Dương, người đã tận tình giảng dạy, hướng dẫn và giúp đỡ tôi trong suốt quá trình thực hiện đồ án chuyên ngành.

Thầy đã dành thời gian quý báu của mình để chia sẻ những kiến thức quý giá, giúp tôi hoàn thành đồ án một cách xuất sắc nhất. Thầy cũng đã luôn động viên, khích lệ và nhắc nhở tôi trong suốt quá trình thực hiện đồ án, giúp tôi vượt qua những khó khăn và thử thách, hoàn thành đồ án đúng tiến độ và đặc biệt là giúp tôi học hỏi được nhiều kiến thức bổ ích.

Tôi cũng xin gửi lời cảm ơn tất cả các bạn đã giúp đỡ và hỗ trợ tôi khi gặp những trở ngại và khó khăn trong quá trình thực hiện đồ án. Nhờ có sự giúp đỡ của các bạn, tôi đã có thể hoàn thành đồ án một cách tốt nhất.

Mặc dù đã cố gắng hoàn thành đề tài tốt nhất nhưng do thời gian và kiến thức chuyên ngành còn hạn chế nên tôi vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm, góp ý của các thầy, cô giảng viên bộ môn để đề tài của tôi được hoàn chỉnh và đầy đủ hơn.

Tôi xin hứa sẽ tiếp tục nỗ lực học tập và phấn đấu, không làm phụ lòng tin của Thầy và các bạn.

Tôi xin chân thành cảm ơn!

Trần Khánh Duy

MỤC LỤC

LỜI CẢM ƠN.....	ii
DANH MỤC HÌNH ẢNH – BẢNG BIỂU	vii
MỞ ĐẦU	10
1. Lý do chọn đề tài	10
2. Mục tiêu.....	10
3. Nội dung	10
4. Đối tượng và phạm vi nghiên cứu	11
5. Phương pháp nghiên cứu	11
CHƯƠNG 1 TỔNG QUAN	12
1.1. Đặt vấn đề.....	12
1.2 Mục đích nghiên cứu	12
CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT.....	14
2.1 ReactJS.....	14
2.1.1 Khái niệm	14
2.1.2 Ưu điểm của ReactJs	15
2.1.3 Nhược điểm của ReactJS	16
2.2 NodeJS	16
2.2.1 Khái niệm	16
2.2.2 Ứng dụng	17
2.2.3 Ưu điểm của NodeJS	18
2.2.4 Nhược điểm của NodeJs	19
2.3 ExpressJS	20
2.3.1 Khái niệm	20
2.3.2 Tính năng của ExpressJS	21
2.3.3 Ưu điểm của ExpressJS	21
2.3.4 Nhược điểm của ExpressJS	22
2.4 RESTful API.....	23
2.4.1 Tổng quan về RESTful API.....	23
2.4.2 Các thành phần chính của RESTful API	23
2.4.3 Các nguyên tắc của RESTful API	23
2.4.4 Ưu điểm của RESTful API.....	25
2.4.5 Nhược điểm của RESTful API	26
2.5 JavaScript.....	26
2.5.1 Khái niệm	26
2.5.2 Ứng dụng	26
2.5.3 Ưu điểm của JavaScript	28
2.5.4 Nhược điểm của JavaScript	28
2.6 MongoDB.....	29
2.6.1 Khái niệm	29
2.6.2 Tính năng nổi bật của MongoDB	29
2.6.3 Ưu điểm của MongoDB	30
2.6.4 Nhược điểm của MongoDB.....	31
2.7 Các nghiệp vụ liên quan đến đề tài	31
2.7.1 Nghiệp vụ quản lý sản phẩm	31
2.7.2 Nghiệp vụ quản lý người dùng	33
2.7.3 Nghiệp vụ giỏ hàng	33

2.7.4	Nghiệp vụ đặt hàng.....	34
2.7.5	Thanh toán.....	34
2.7.6	Nghiệp vụ quản lý kho	35
2.7.7	Nghiệp vụ thống kê và báo cáo	35
2.8	Các công trình nghiên cứu liên quan đến đề tài	35
2.8.1	Báo cáo thực tập: Lập trình website bán hàng trực tuyến với ReactJS, NodeJS và ExpressJS.....	35
2.8.2	Báo cáo thực tập: Phát triển ứng dụng web với ReactJS và tích hợp NodeJS	36
2.8.3	Tài liệu thực hành: Hướng dẫn lập trình website bán hàng trực tuyến với NodeJS và ReactJS	36
CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU		37
3.1	Mô tả hệ thống.....	37
3.2	Xác định các yêu cầu chức năng của hệ thống	38
3.3	Thiết kế dữ liệu hệ thống.....	39
3.3.1	Mô hình dữ liệu mức quan niệm	39
3.3.2	Mô hình dữ liệu mức logic	40
3.3.3	Các thực thể.....	40
3.4	Thiết kế xử lý hệ thống.....	43
3.4.1	Biểu đồ Use Case tổng quát.....	43
3.4.2	Use Case tổng quan Admin	43
3.4.3	Use Case tổng quan của khách	44
3.4.4	Use Case tổng quan của người dùng	45
3.4.5	Quản lý người dùng	46
3.4.6	Quản lý sản phẩm	47
3.4.7	Quản lý giỏ hàng	48
3.4.8	Quản lý đơn hàng	49
3.4.9	Quản lý khuyến mãi.....	50
3.5	Mô tả.....	50
3.5.1	Mô tả các Actor	50
3.5.1	Mô tả các use case	51
CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU		53
4.1	Giao diện và chức năng của người dùng	53
4.1.1	Giao diện trang chủ.....	53
4.1.2	Giao diện Sản phẩm.....	54
4.1.3	Giao diện đăng nhập.....	55
4.1.4	Giao diện đăng ký.....	56
4.1.5	Giao diện giỏ hàng.....	57
4.1.6	Giao diện nhập thông tin vận chuyển	58
4.1.7	Giao diện xác nhận hóa đơn	58
4.1.8	Giao diện thanh toán.....	59
4.1.9	Giao diện đơn hàng.....	60
4.2	Giao diện và chức năng của quản trị viên	60
4.1.1	Giao diện quản lý chung.....	60
4.1.2	Giao diện quản lý sản phẩm	61
4.1.3	Giao diện quản lý đơn hàng.....	61
4.1.4	Giao diện quản lý người dùng	62
4.1.5	Giao diện quản lý khuyến mãi.....	62
4.1.6	Giao diện quản lý bình luận.....	63
CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN		64
5.1	Kết quả đạt được.....	64

5.2 Hướng phát triển.....	64
DANH MỤC TÀI LIỆU THAM KHẢO	65
PHỤ LỤC	67

DANH MỤC HÌNH ẢNH – BẢNG BIỂU

DANH MỤC HÌNH ẢNH

Hình 1. Mô hình dữ liệu mức quan niệm	39
Hình 2. Mô hình dữ liệu mức logic	40
Hình 3. Use Case tổng quát hệ thống	43
Hình 4. Use Case tổng quan Admin	43
Hình 5. Use Case tổng quan của khách	44
Hình 6. Use Case tổng quan của người dùng	45
Hình 7. Use Case quản lý người dùng.....	46
Hình 8. Use case quản lý sản phẩm.....	47
Hình 9. Use case quản lý giỏ hàng	48
Hình 10. Use case quản lý đơn hàng	49
Hình 11. Use case quản lý khuyến mãi	50
Hình 12. Giao diện trang chủ	53
Hình 13. Giao diện Sản phẩm.....	54
Hình 14. Giao diện đăng nhập.....	55
Hình 15. Giao diện đăng ký.....	56
Hình 16. Giao diện giỏ hàng.....	57
Hình 17. Giao diện nhập thông tin vận chuyển	58
Hình 18. Giao diện xác nhận hóa đơn	58
Hình 19. Giao diện thanh toán.....	59
Hình 20. Giao diện đơn hàng.....	60
Hình 21. Quản lý tổng quan	60
Hình 22. Giao diện quản lý sản phẩm	61
Hình 23. Quản lý đơn hàng	61
Hình 24. Giao diện quản lý người dùng	62
Hình 25. Giao diện quản lý khuyến mãi.....	62
Hình 26. Giao diện quản lý bình luận.....	63

DANH MỤC BẢNG

Bảng 1. Order	40
Bảng 2. Order_items.....	40
Bảng 3. Payment.....	41
Bảng 4. User	41
Bảng 5. Product	41
Bảng 6. Category	42
Bảng 7. Review.....	42
Bảng 8. Promotion.....	42

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Đề tài “Thiết kế website bán thiết bị di động” là một hành trình nghiên cứu và phát triển hệ thống thương mại điện tử chuyên về lĩnh vực kinh doanh các thiết bị di động. Dự án không chỉ tập trung vào việc xây dựng một giao diện thân thiện, tối ưu trải nghiệm người dùng mà còn chú trọng đến tính hiệu quả trong quản lý sản phẩm, đơn hàng và bảo mật hệ thống.

Mục tiêu chính

Nghiên cứu và phân tích yêu cầu của một website bán thiết bị di động để xác định các chức năng cần thiết, như tìm kiếm sản phẩm, quản lý giỏ hàng, thanh toán trực tuyến, khuyến mãi và đánh giá sản phẩm.

Phát triển hệ thống quản lý sản phẩm, khách hàng, và đơn hàng nhằm hỗ trợ doanh nghiệp trong việc quản trị hiệu quả.

Tích hợp các biện pháp bảo mật như xác thực người dùng, mã hóa dữ liệu, và quản lý phiên làm việc để đảm bảo an toàn thông tin của khách hàng.

Đưa sản phẩm lên môi trường trực tuyến, đảm bảo hiệu năng và khả năng mở rộng trong tương lai.

Hướng tiếp cận

Sử dụng **ReactJS** để phát triển giao diện người dùng, đảm bảo tính hiện đại, tối ưu trải nghiệm và tương thích đa nền tảng.

Tận dụng **NodeJS** và **ExpressJS** trong việc xây dựng hệ thống backend, tạo các API để xử lý dữ liệu, đồng thời sử dụng **MongoDB** làm cơ sở dữ liệu để quản lý thông tin một cách hiệu quả.

Áp dụng các mô hình thiết kế như RESTful API để đảm bảo tính linh hoạt và khả năng mở rộng của hệ thống.

Phát triển hệ thống quản trị với giao diện trực quan, dễ sử dụng, hỗ trợ quản lý các sản phẩm, đơn hàng, chương trình khuyến mãi, và đánh giá từ người dùng.

Cách giải quyết vấn đề

Phân tích yêu cầu hệ thống, thiết kế cơ sở dữ liệu và xây dựng sơ đồ ERD để đảm bảo các chức năng đáp ứng yêu cầu thực tế.

Xây dựng giao diện thân thiện, dễ dàng tìm kiếm và lọc sản phẩm, kết hợp với các tính năng quản lý tài khoản người dùng và giỏ hàng.

Tích hợp chức năng thanh toán trực tuyến an toàn thông qua các cổng thanh toán như VNPay hoặc PayPal.

Tối ưu hóa hiệu suất hệ thống thông qua cơ chế phân trang và quản lý bộ nhớ hiệu quả trên server.

Triển khai và kiểm tra hệ thống trên môi trường trực tuyến để đảm bảo hoạt động ổn định.

Kết quả đạt được

Xây dựng thành công website bán thiết bị di động với đầy đủ các chức năng cần thiết, bao gồm tìm kiếm, giỏ hàng, quản lý đơn hàng, thanh toán trực tuyến, và đánh giá sản phẩm.

Phát triển giao diện người dùng thân thiện, hỗ trợ cả khách hàng và người quản trị dễ dàng sử dụng.

Đảm bảo hiệu suất và bảo mật của hệ thống, đáp ứng nhu cầu thực tiễn của một trang thương mại điện tử hiện đại.

Nâng cao kiến thức và kỹ năng trong lập trình front-end, back-end, và quản lý dự án thực tế.

Hướng phát triển

Mở rộng chức năng như chatbot tư vấn tự động, theo dõi đơn hàng theo thời gian thực, và gợi ý sản phẩm dựa trên trí tuệ nhân tạo (AI).

Tích hợp thêm các phương thức thanh toán đa dạng và hỗ trợ nhiều ngôn ngữ để phục vụ người dùng quốc tế.

Tối ưu hóa bảo mật hệ thống, nâng cao trải nghiệm người dùng và mở rộng quy mô hệ thống cho các cửa hàng hoặc đối tác lớn hơn.

MỞ ĐẦU

1. Lý do chọn đề tài

Thương mại điện tử hiện nay đã trở thành một phần không thể thiếu trong đời sống hiện đại. Đặc biệt trong lĩnh vực kinh doanh các thiết bị di động đang bùng nổ mạnh mẽ với nhu cầu đa dạng từ khách hàng. Đề tài xây dựng website bán thiết bị di động mang lại nhiều ý nghĩa thực tiễn và lợi ích cụ thể nhằm đáp ứng nhu cầu của thị trường, giúp doanh nghiệp kinh doanh thiết bị di động mở rộng quy mô và dễ dàng tiếp cận khách hàng tiềm năng, áp dụng công nghệ mới tạo cơ hội học hỏi và sử dụng các công nghệ tiên tiến như NodeJS, ReactJS, và MongoDB, giúp phát triển kỹ năng thực hành khi thực hiện đề tài sinh viên rèn luyện được kỹ năng lập trình, thiết kế cơ sở dữ liệu và triển khai hệ thống thực tế, thúc đẩy sáng tạo và giải quyết vấn đề, tăng cường tư duy logic thông qua việc tối ưu hóa giao diện, hiệu năng và trải nghiệm người dùng.

2. Mục tiêu

Xây dựng một hệ thống website bán thiết bị di động hoàn chỉnh với giao diện thân thiện, tính năng phong phú và hoạt động ổn định.

Tích hợp các công nghệ hiện đại như NodeJS, ExpressJS, ReactJS và MongoDB.

Cung cấp các tính năng cơ bản như quản lý sản phẩm, giỏ hàng, thanh toán và đánh giá sản phẩm.

Tối ưu hóa bảo mật của hệ thống, đảm bảo trải nghiệm tốt nhất có thể cho người dùng.

3. Nội dung

Nghiên cứu các công nghệ liên quan đến phát triển web như NodeJS, ReactJS, MongoDB và ExpressJS.

Xây dựng mô hình cơ sở dữ liệu để quản lý thông tin sản phẩm, người dùng, đơn hàng và thanh toán.

Phát triển các tính năng chính của hệ thống: quản lý sản phẩm, giỏ hàng, thanh toán, quản lý đơn hàng và đánh giá sản phẩm.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Nghiên cứu các phương pháp và công nghệ phát triển website thương mại điện tử, cụ thể là các công nghệ hỗ trợ xây dựng backend (NodeJS), frontend (ReactJS) và cơ sở dữ liệu (MongoDB). Đối tượng nghiên cứu bao gồm các thành phần cấu trúc website, cơ chế xử lý dữ liệu, tối ưu hóa hiệu năng và bảo mật thông tin.

Phạm vi nghiên cứu: Đề tài tập trung vào việc phát triển một website bán thiết bị di động trực tuyến với các tính năng cơ bản như quản lý sản phẩm, giỏ hàng, thanh toán và quản lý đơn hàng.

5. Phương pháp nghiên cứu

Phương pháp thu thập tài liệu: Tiến hành tìm kiếm và nghiên cứu các tài liệu liên quan đến công nghệ NodeJS, ReactJS và MongoDB thông qua sách, bài báo khoa học, diễn đàn chuyên ngành và tài liệu chính thức từ các nhà phát triển công cụ. Việc này giúp hiểu rõ nguyên lý hoạt động, tính năng và cách áp dụng công nghệ vào thực tế.

Phương pháp phân tích và thiết kế: Dựa trên các yêu cầu cụ thể của hệ thống, tiến hành phân tích và thiết kế mô hình dữ liệu bằng ERD (Entity Relationship Diagram) để đảm bảo cơ sở dữ liệu được xây dựng chặt chẽ, tối ưu. Song song đó, thực hiện phân tích yêu cầu hệ thống và xây dựng kiến trúc tổng thể để tạo nên một ứng dụng linh hoạt, dễ dàng mở rộng.

Phương pháp thực nghiệm: Triển khai xây dựng hệ thống thực tế với đầy đủ các thành phần bao gồm backend, frontend và cơ sở dữ liệu. Kiểm thử từng chức năng trong quá trình phát triển để phát hiện và khắc phục kịp thời các lỗi, đảm bảo hệ thống hoạt động đúng như mong đợi. Sử dụng các công cụ hỗ trợ như Postman để kiểm tra API, trình duyệt để kiểm tra giao diện và các công cụ phân tích hiệu năng.

CHƯƠNG 1 TỔNG QUAN

1.1. Đặt vấn đề

Trong thời đại công nghệ phát triển mạnh mẽ, thương mại điện tử đã và đang khẳng định vai trò quan trọng trong việc thay đổi cách thức giao dịch và kinh doanh của con người hiện nay. Đây không chỉ là xu hướng quan trọng mà còn là nền tảng hỗ trợ các doanh nghiệp mở rộng thị trường và tiếp cận khách hàng một cách nhanh chóng và hiệu quả. Đặc biệt, lĩnh vực kinh doanh các thiết bị di động đang tăng trưởng mạnh mẽ khi các sản phẩm công nghệ ngày càng trở nên phổ biến và không thể thiếu trong cuộc sống hiện đại. [1]

Tuy nhiên, thực trạng hiện nay cho thấy nhiều website thương mại điện tử vẫn còn tồn tại những hạn chế như tốc độ tải trang chậm, giao diện thiếu thân thiện hoặc chưa tối ưu trải nghiệm người dùng. Điều này không chỉ làm giảm sức cạnh tranh mà còn ảnh hưởng tiêu cực đến mức độ hài lòng của khách hàng.

Vì vậy, việc phát triển một website bán các thiết bị di động với các tính năng hiện đại và tối ưu, là một yêu cầu cấp thiết. Một hệ thống được xây dựng tốt không chỉ đáp ứng nhu cầu ngày càng cao của thị trường mà còn góp phần nâng cao hiệu quả kinh doanh, tạo dựng uy tín cho doanh nghiệp, và cải thiện trải nghiệm mua sắm của người dùng.

1.2 Mục đích nghiên cứu

Đồ án hướng tới việc xây dựng một website bán thiết bị di động hiện đại, ứng dụng các công nghệ tiên tiến để giải quyết những hạn chế mà các hệ thống hiện tại đang gặp phải. Mục tiêu chính bao gồm:

Quản lý sản phẩm: Cung cấp các công cụ để sử dụng cho phép thêm mới, chỉnh sửa, xóa và cập nhật thông tin sản phẩm, đảm bảo dữ liệu được tổ chức và truy cập hiệu quả.

Quản lý người dùng: Phát triển hệ thống đăng ký và đăng nhập an toàn, hỗ trợ quản lý tài khoản cá nhân, tạo điều kiện để người dùng tương tác và theo dõi thông tin giao dịch.

Giỏ hàng và thanh toán: Xây dựng quy trình giỏ hàng linh hoạt, kết hợp phương thức thanh toán như thẻ tín dụng mang lại sự tiện lợi tối đa cho người mua.

Quản lý đơn hàng: Cho phép khách hàng theo dõi trạng thái đơn hàng theo thời gian thực, đồng thời hỗ trợ quản trị viên giám sát và xử lý đơn hàng nhanh chóng.

Đánh giá sản phẩm: Tạo không gian để khách hàng chia sẻ nhận xét, đánh giá chất lượng sản phẩm sau khi sử dụng, qua đó nâng cao uy tín và cải thiện dịch vụ.

Ngoài ra, đề án còn chú trọng vào các khía cạnh như bảo mật, hiệu năng và tính thân thiện với người dùng. Sản phẩm cuối cùng không chỉ là một công cụ hỗ trợ kinh doanh mà còn là một nền tảng tương tác hiện đại, đáp ứng nhu cầu của cả người bán và người mua trong môi trường cạnh tranh ngày càng khốc liệt.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

2.1 ReactJS

2.1.1 Khái niệm

React (ReactJS) là một thư viện JavaScript mã nguồn mở, được dùng để xây dựng giao diện người dùng (frontend) cho web. React chỉ tập trung vào phần hiển thị giao diện (view), chứ không can thiệp vào cách sắp xếp logic nghiệp vụ hoặc cấu trúc ứng dụng

React là không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM [4][5]

Một số khái niệm cơ bản:

Virtual DOM: (Document Object Model ảo) là một bản sao nhẹ hơn của DOM thật. DOM thật là cấu trúc cây chứa tất cả các thành phần HTML trong trang web. Khi người dùng tương tác với ứng dụng thì ứng dụng sẽ thay đổi nội dung và DOM thật phải được cập nhật. Công nghệ tối ưu hiệu năng vượt trội trong ReactJS.

Một trong những yếu tố cốt lõi giúp ReactJS đạt được hiệu năng cao là việc sử dụng Virtual DOM (DOM ảo). Công nghệ này được thiết kế để giải quyết bài toán về hiệu suất khi giao diện người dùng cần thay đổi liên tục.

Với việc sử dụng Virtual DOM, ReactJS không chỉ giải quyết vấn đề hiệu suất mà còn mở ra hướng đi mới trong việc xây dựng các giao diện hiện đại và tối ưu hóa quy trình phát triển. Tuy nhiên, việc cập nhật trực tiếp DOM thật có thể gây ra tình trạng chậm chạp và kém hiệu quả, đặc biệt trong các ứng dụng lớn với nhiều phần tử. Để giải quyết vấn đề này, React sử dụng Virtual DOM, với mục đích chỉ những phần có sự thay đổi mới được cập nhật lên DOM thật, giúp tiết kiệm thời gian và tài nguyên.

JSX (JavaScript XML): Cầu nối giữa HTML và JavaScript trong ReactJS. JSX là một cú pháp mở rộng được sử dụng trong React, cho phép viết mã giống như HTML trực tiếp trong JavaScript. Nó kết hợp sức mạnh của cả hai ngôn ngữ, giúp bạn xây dựng giao diện một cách trực quan và dễ dàng.

JSX không phải là HTML mà là một cú pháp đặc biệt của JavaScript. Khi trình duyệt xử lý, JSX sẽ được chuyển đổi thành các lệnh JavaScript thông qua Babel hoặc công cụ tương tự, sau đó tạo ra các phần tử DOM thật

JSX không chỉ giúp việc viết giao diện trở nên trực quan mà còn tăng năng suất lập trình bằng cách kết hợp logic JavaScript vào thiết kế giao diện. Đây là một trong những yếu tố quan trọng làm nên sức mạnh của ReactJS.

Component: Đơn vị cơ bản trong ReactJS Trong React, Component là nền tảng cốt lõi để xây dựng giao diện người dùng (UI). Mỗi component đại diện cho một phần nhỏ của giao diện, và việc chia giao diện thành các component nhỏ giúp việc phát triển trở nên dễ dàng, gọn gàng và có tổ chức.

Component có các đặc điểm như mỗi component được thiết kế để hoạt động độc lập, không phụ thuộc vào các thành phần khác, giúp dễ dàng quản lý và bảo trì, Một component nên có kích thước nhỏ và chỉ tập trung vào một nhiệm vụ cụ thể điều này giúp cải thiện tính dễ đọc và dễ tái sử dụng. Các component được viết một cách tổng quát và linh hoạt để có thể dùng lại ở nhiều nơi trong ứng dụng hoặc thậm chí ở các dự án khác.

Component không chỉ giúp cấu trúc giao diện trở nên rõ ràng mà còn mang lại sự linh hoạt và hiệu quả khi phát triển ứng dụng. Đây chính là điểm mạnh làm nên sự khác biệt của ReactJS trong việc xây dựng giao diện người dùng hiện đại.[4][5][7][8]

2.1.2 Ưu điểm của ReactJs

Hiệu suất cao nhờ DOM ảo (Virtual DOM): ReactJS sử dụng cơ chế Virtual DOM, giúp lưu trữ cấu trúc dữ liệu trong bộ nhớ và chỉ cập nhật những thay đổi cần thiết lên DOM thật. Điều này giúp giảm thiểu việc thao tác trực tiếp với DOM trình duyệt, từ đó cải thiện hiệu suất và tăng tốc độ phản hồi của ứng dụng.

Tái sử dụng và quản lý dễ dàng với Component: ReactJS cho phép chia ứng dụng thành các component độc lập, mỗi component đại diện cho một chức năng cụ thể. Các component này có thể được tái sử dụng trong nhiều phần của ứng dụng, giúp giảm thiểu việc lặp lại mã nguồn. Việc tổ chức mã nguồn theo từng component không chỉ cải thiện khả năng bảo trì mà còn giúp lập trình viên dễ dàng mở rộng hoặc sửa đổi ứng dụng khi cần.

Dễ tiếp cận và mã nguồn mở: Là một thư viện mã nguồn mở, ReactJS cung cấp tài liệu phong phú và cộng đồng lớn, tạo điều kiện thuận lợi cho người mới bắt đầu học lập trình. Các nhà phát triển có thể dễ dàng truy cập tài liệu chính thức, hướng dẫn, và các ví dụ thực tiễn để nhanh chóng làm quen với ReactJS

Được hỗ trợ bởi các công ty lớn: ReactJS được phát triển và duy trì bởi Meta (Facebook), với sự tham gia của các công ty lớn như Instagram, Netflix, và Apple. Điều này đảm bảo sự ổn định, cập nhật thường xuyên và tính đáng tin cậy của thư viện

Xây dựng giao diện người dùng đa nền tảng: ReactJS có thể được sử dụng để phát triển giao diện người dùng không chỉ cho các ứng dụng web mà còn cho các ứng dụng di động thông qua React Native. Điều này giúp lập trình viên sử dụng cùng một kiến thức để xây dựng ứng dụng trên nhiều nền tảng khác nhau.

Dễ dàng gỡ lỗi: Với ReactJS, mã nguồn chủ yếu được viết bằng JavaScript thay vì HTML, giúp dễ dàng hơn trong việc tìm kiếm và sửa lỗi. Công cụ như React Developer Tools cũng hỗ trợ lập trình viên kiểm tra và theo dõi trạng thái của các component trong ứng dụng một cách trực quan.[5][6]

2.1.3 Nhược điểm của ReactJS

Khó khăn với người mới làm quen: ReactJS sử dụng JSX, một cú pháp kết hợp JavaScript với HTML và CSS. Điều này khác biệt với cách tiếp cận truyền thống khi HTML, CSS, và JavaScript được tách biệt. Khi mới học lập trình hoặc chưa quen với JavaScript XML (JSX), việc hiểu và viết mã ReactJS có thể gây khó khăn và dễ nhầm lẫn giữa JSX và HTML.

Dung lượng file lớn khi phát triển: Trong giai đoạn phát triển, ReactJS yêu cầu sử dụng nhiều thư viện và công cụ hỗ trợ, khiến dung lượng file trở nên khá lớn. Điều này có thể ảnh hưởng đến tốc độ tải ban đầu của ứng dụng, đặc biệt khi làm việc với các dự án lớn. Tuy nhiên, nhược điểm này thường được giảm thiểu bằng các công cụ nén và tối ưu hóa trong quá trình triển khai (deployment). [5][6]

2.2 NodeJS

2.2.1 Khái niệm

Node.js là một nền tảng mã nguồn mở, dựa trên công cụ JavaScript V8 của Google Chrome, cho phép thực thi mã JavaScript trên máy chủ. Với kiến trúc không

đồng bộ và hướng sự kiện, Node.js tối ưu hóa hiệu suất xử lý đa nhiệm, đồng thời hỗ trợ xây dựng các ứng dụng real-time một cách hiệu quả.

Điểm đặc biệt của Node.js là khả năng chia sẻ mã giữa máy chủ và trình duyệt, giúp đơn giản hóa quá trình phát triển. Kèm theo đó là hệ thống quản lý gói npm mạnh mẽ, cho phép dễ dàng quản lý và tích hợp các thư viện, dependencies.

Với sự hỗ trợ từ cộng đồng lớn và ngày càng phát triển, Node.js không chỉ là một công cụ phổ biến mà còn trở thành sự lựa chọn hàng đầu trong việc phát triển ứng dụng hiệu suất cao, đặc biệt là các ứng dụng yêu cầu tính thời gian thực. Các framework như Express.js và công nghệ như Socket.io đã mở rộng khả năng của Node.js, giúp các nhà phát triển xây dựng ứng dụng web và ứng dụng streaming phức tạp một cách dễ dàng và hiệu quả.[9]

2.2.2 Ứng dụng

Ứng dụng thời gian thực (Real-time Applications): Node.js là lựa chọn lý tưởng để xây dựng các ứng dụng yêu cầu kết nối và cập nhật liên tục giữa máy chủ và máy khách, nhờ kiến trúc event-driven và hỗ trợ mạnh mẽ cho WebSocket. Các ứng dụng như hệ thống trò chuyện trực tuyến, game đa người chơi, hoặc bảng thông báo thời gian thực đều tận dụng được tối đa sức mạnh của Node.js.

API RESTful và GraphQL: Node.js thường được sử dụng để phát triển các API RESTful và GraphQL, đóng vai trò là cầu nối cung cấp dữ liệu cho các ứng dụng frontend hoặc mobile. Tốc độ xử lý nhanh cùng khả năng quản lý lượng lớn yêu cầu đồng thời giúp Node.js trở thành một công cụ lý tưởng cho việc xây dựng API hiệu quả và linh hoạt.

Ứng dụng Single Page Application (SPA): Trong các ứng dụng SPA, việc xử lý yêu cầu và dữ liệu từ máy chủ là yếu tố quan trọng. Node.js đảm bảo khả năng xử lý nhanh chóng và mượt mà, cung cấp nền tảng mạnh mẽ để tạo ra trải nghiệm người dùng liền mạch trên các trang web đơn trang, nơi tất cả thao tác đều diễn ra mà không cần tải lại trang.

Ứng dụng Internet of Things (IoT): Node.js là một trong những lựa chọn hàng đầu khi phát triển hệ thống IoT, nhờ vào khả năng xử lý lượng lớn sự kiện trong thời gian ngắn và sử dụng tài nguyên một cách tối ưu. Các thiết bị IoT như cảm biến, thiết

bị thông minh và các hệ thống giám sát thời gian thực đều có thể vận hành hiệu quả trên nền Node.js.

Ứng dụng truyền phát dữ liệu và nội dung (Streaming Applications): Node.js hỗ trợ truyền phát dữ liệu theo luồng (streaming) rất hiệu quả, làm cho việc xây dựng các ứng dụng phát trực tiếp, truyền tải nội dung video, hoặc xử lý tệp dữ liệu lớn trở nên dễ dàng. Với khả năng xử lý từng phần dữ liệu khi nhận được, Node.js giảm thiểu độ trễ và tăng hiệu suất ứng dụng.

Ứng dụng thương mại điện tử (E-commerce Applications): Đối với các trang web thương mại điện tử, khả năng xử lý đồng thời hàng nghìn yêu cầu từ người dùng là yếu tố then chốt. Node.js, với kiến trúc không đồng bộ, đảm bảo tốc độ và khả năng mở rộng, giúp cải thiện trải nghiệm khách hàng ngay cả khi lưu lượng truy cập tăng cao.

Ứng dụng máy chủ proxy (Proxy Server Applications): Node.js có thể được triển khai như một máy chủ proxy, đảm nhiệm việc xử lý các yêu cầu giữa máy khách và máy chủ đích hoặc API bên thứ ba. Điều này đặc biệt hữu ích khi cần quản lý nhiều kết nối đồng thời hoặc hợp nhất dữ liệu từ nhiều nguồn khác nhau trong thời gian thực.

Ứng dụng đa nền tảng (Cross-platform Applications): Khi kết hợp với các công cụ như Electron, Node.js có khả năng phát triển các ứng dụng desktop đa nền tảng, chạy trên cả Windows, macOS và Linux. Điều này mở ra cơ hội tạo ra các phần mềm dễ sử dụng, tích hợp sâu với hệ điều hành mà không cần phát triển riêng lẻ cho từng nền tảng.[9][10]

2.2.3 Ưu điểm của NodeJS

Xử lý đồng thời mạnh mẽ (Concurrency): Node.js được xây dựng dựa trên kiến trúc không đồng bộ và mô hình hướng sự kiện, cho phép xử lý hàng nghìn kết nối đồng thời mà không cần tạo luồng mới cho mỗi kết nối. Điều này giúp tối ưu hóa khả năng mở rộng và đảm bảo hiệu suất vượt trội. Node.js đặc biệt phù hợp với các ứng dụng thời gian thực như trò chuyện trực tuyến, trò chơi nhiều người chơi, cập nhật dữ liệu theo thời gian thực hoặc ứng dụng streaming

Hiệu suất vượt trội: Nhờ sử dụng V8 JavaScript engine của Google, Node.js biên dịch mã JavaScript thành mã máy ngay trong lúc chạy (just-in-time compilation), giúp

giảm độ trễ và tăng tốc độ xử lý. Khả năng tối ưu hóa hiệu suất này làm cho Node.js trở thành một lựa chọn lý tưởng cho các ứng dụng đòi hỏi xử lý nhanh và hiệu quả.

Khả năng chia sẻ mã giữa máy chủ và trình duyệt: Node.js mang đến sự linh hoạt bằng cách cho phép sử dụng JavaScript ở cả phía máy chủ và trình duyệt. Điều này giúp các nhà phát triển dễ dàng chia sẻ và tái sử dụng mã giữa hai môi trường, giảm thời gian phát triển và tăng cường hiệu quả trong việc duy trì mã nguồn.

Hệ sinh thái mô-đun nguồn mở mạnh mẽ: Node.js đi kèm với npm (Node Package Manager), một hệ thống quản lý gói lớn và đa dạng, cung cấp hàng triệu thư viện mã nguồn mở. Với npm, việc quản lý dependencies, tích hợp và chia sẻ mã nguồn trở nên đơn giản và hiệu quả. Sự phát triển nhanh chóng của cộng đồng npm mang lại nhiều lựa chọn linh hoạt cho các nhà phát triển khi xây dựng ứng dụng.

Nhẹ và linh hoạt: Là một môi trường thực thi nhẹ, Node.js giúp giảm tải tài nguyên hệ thống và tối ưu hóa việc triển khai ứng dụng. Tính linh hoạt của Node.js cho phép nó được áp dụng vào nhiều mục đích khác nhau, từ xây dựng các ứng dụng web đơn giản đến phát triển hệ thống IoT hoặc dịch vụ API phức tạp.

Tăng tốc quá trình phát triển: Node.js hỗ trợ sử dụng cùng một ngôn ngữ - JavaScript - cho cả phía máy chủ và máy khách, giúp loại bỏ sự phức tạp trong việc chuyển đổi ngôn ngữ giữa hai môi trường. Điều này không chỉ rút ngắn thời gian phát triển mà còn giúp các đội ngũ lập trình tận dụng tối đa kỹ năng sẵn có của mình.

Cộng đồng hỗ trợ mạnh mẽ: Node.js được hỗ trợ bởi một cộng đồng lớn, năng động và đam mê, giúp các nhà phát triển dễ dàng tìm kiếm tài liệu, giải pháp hoặc hỗ trợ kỹ thuật. Các bản cập nhật và cải tiến thường xuyên từ cộng đồng đảm bảo Node.js luôn được nâng cấp để đáp ứng nhu cầu hiện đại, đồng thời giữ được sự an toàn và ổn định cho các ứng dụng.[9][11]

2.2.4 Nhược điểm của NodeJs

Hiệu suất hạn chế với tác vụ CPU nặng: Node.js hoạt động dựa trên single-thread, khiến nó không phù hợp với các tác vụ yêu cầu xử lý phức tạp về CPU, như các thuật toán tính toán nặng hoặc xử lý dữ liệu lớn. Khi gặp các tác vụ này, event loop có thể bị chặn, làm giảm hiệu suất của toàn bộ ứng dụng.

Không phù hợp cho các ứng dụng yêu cầu đa luồng: Mặc dù Node.js có thể xử lý lượng lớn kết nối đồng thời, nhưng đối với các ứng dụng cần xử lý đa luồng phức tạp hoặc chạy nhiều tiến trình song song, các nền tảng như Java hay .NET lại mang lại hiệu quả tốt hơn nhờ hỗ trợ multi-threading mạnh mẽ.

Quản lý callback phức tạp: Dù các công cụ như `async/await` đã cải thiện đáng kể việc xử lý bất đồng bộ, nhưng Node.js vẫn gặp khó khăn trong việc quản lý các callback phức tạp, đặc biệt với mã nguồn lớn. Điều này có thể dẫn đến tình trạng callback hell, khiến mã nguồn trở nên khó đọc và khó bảo trì, đặc biệt đối với các lập trình viên mới.

Tính nhất quán thấp trong hệ sinh thái npm: Với hàng triệu module và thư viện trong npm, việc chọn sai thư viện có thể dẫn đến rủi ro. Một số thư viện không được duy trì hoặc kém chất lượng, gây ra lỗi trong quá trình phát triển và triển khai. Lập trình viên cần cẩn thận trong việc đánh giá và chọn lựa các module phù hợp.

Rủi ro bảo mật từ hệ sinh thái: Node.js có hệ sinh thái mở, cho phép mọi người đóng góp module lên npm. Tuy nhiên, nếu không kiểm tra kỹ lưỡng, việc sử dụng các thư viện không an toàn hoặc chứa mã độc có thể dẫn đến lỗ hổng bảo mật nghiêm trọng, ảnh hưởng tiêu cực đến hệ thống. Điều này đòi hỏi các lập trình viên phải có quy trình kiểm tra bảo mật cẩn thận trong quá trình phát triển.[9][11]

2.3 ExpressJS

2.3.1 Khái niệm

Express.js là một framework mạnh mẽ và phổ biến, được xây dựng trên nền tảng Node.js để phát triển ứng dụng web và API. Với khả năng xử lý các yêu cầu HTTP, định tuyến (routing), tích hợp middleware và mở rộng linh hoạt, Express.js mang lại một giải pháp hiệu quả cho việc phát triển ứng dụng hiện đại.

Framework này tập trung vào việc đơn giản hóa quá trình xây dựng ứng dụng bằng cách cung cấp cấu trúc rõ ràng và dễ hiểu. Các nhà phát triển có thể dễ dàng tổ chức mã nguồn, quản lý yêu cầu/ phản hồi và tích hợp các middleware bên ngoài để mở rộng tính năng của ứng dụng. Với cộng đồng lớn và sự hỗ trợ rộng rãi, Express.js đã trở thành lựa chọn hàng đầu trong việc xây dựng ứng dụng web và API dựa trên Node.js.[12]

2.3.2 Tính năng của ExpressJS

Hệ thống Middleware mạnh mẽ: Middleware là một trong những tính năng cốt lõi của Express.js, cho phép xử lý yêu cầu (request) và phản hồi (response) một cách linh hoạt và có tổ chức. Các middleware hỗ trợ: Thêm, chỉnh sửa hoặc loại bỏ chức năng trong từng tầng xử lý và dễ dàng mở rộng khả năng của ứng dụng thông qua việc tích hợp tùy chỉnh hoặc bên thứ ba.

Router tích hợp sẵn: Express.js cung cấp hệ thống định tuyến (routing) linh hoạt và có tổ chức, hỗ trợ các phương thức HTTP phổ biến như GET, POST, PUT, DELETE. Giúp xây dựng các ứng dụng với cấu trúc URL rõ ràng, dễ bảo trì và mở rộng

Quản lý API hiệu quả: Express.js là công cụ lý tưởng để phát triển API, đặc biệt là API RESTful hoặc GraphQL. Framework cung cấp: Cú pháp đơn giản để định nghĩa các endpoints và Khả năng tích hợp dễ dàng với cơ sở dữ liệu và các thư viện liên quan để tối ưu hóa API.

Hỗ trợ Template Engines: Express.js tương thích với nhiều công cụ template phổ biến như Pug, EJS, và Handlebars, giúp dễ dàng xây dựng giao diện người dùng động. Điều này đặc biệt hữu ích trong các ứng dụng cần hiển thị dữ liệu từ máy chủ một cách linh hoạt.

Tích hợp mượt mà với Node.js và các công nghệ khác: Là một phần mở rộng của Node.js, Express.js hoạt động trơn tru với các thư viện bên ngoài thông qua npm. Framework này cũng hỗ trợ tích hợp dễ dàng với cơ sở dữ liệu như MongoDB, PostgreSQL hoặc Redis, giúp xây dựng các ứng dụng phức tạp trở nên đơn giản hơn.[13]

2.3.3 Ưu điểm của ExpressJS

Nhẹ và linh hoạt: Express.js không áp đặt cấu trúc cố định, cho phép lập trình viên tự do thiết kế ứng dụng theo nhu cầu cụ thể. Kích thước nhỏ gọn và tính linh hoạt giúp framework này phù hợp với nhiều dự án, từ đơn giản đến phức tạp.

Dễ học và sử dụng: Với cú pháp đơn giản, Express.js rất dễ tiếp cận đối với các lập trình viên đã quen thuộc với JavaScript hoặc Node.js. Hơn nữa, tài liệu chi tiết cùng sự hỗ trợ từ cộng đồng lớn giúp giảm thiểu rào cản học tập.

Tốc độ phát triển nhanh: Express.js cung cấp nhiều tính năng tích hợp sẵn, giúp tiết kiệm thời gian khi phát triển ứng dụng. Ngoài ra, hệ sinh thái npm mang đến hàng ngàn module và plugin hữu ích, giúp mở rộng khả năng của ứng dụng mà không cần viết lại từ đầu.

Hiệu suất vượt trội: Dựa trên Node.js, Express.js tận dụng kiến trúc không đồng bộ để xử lý các yêu cầu một cách nhanh chóng và hiệu quả. Điều này giúp ứng dụng duy trì hiệu suất cao ngay cả khi xử lý nhiều kết nối đồng thời.

Hỗ trợ đa dạng ứng dụng: Express.js phù hợp với nhiều loại ứng dụng khác nhau: Phát triển API RESTful hoặc GraphQL, ứng dụng web tĩnh hoặc động, các dự án phức tạp như hệ thống quản lý nội dung hoặc thương mại điện tử.[12][13]

2.3.4 Nhược điểm của ExpressJS

Hạn chế trong các dự án lớn: Express.js không áp đặt cấu trúc chặt chẽ, điều này có thể gây khó khăn trong việc quản lý mã nguồn khi dự án mở rộng hoặc khi làm việc với đội ngũ lập trình đông người.

Phụ thuộc nhiều vào middleware: Việc sử dụng quá nhiều middleware có thể làm mã nguồn trở nên phức tạp và khó bảo trì. Hơn nữa, việc lạm dụng middleware có thể ảnh hưởng đến hiệu suất của ứng dụng.

Yêu cầu nhiều mã nguồn thủ công: So với các framework như Django hay Laravel, Express.js đòi hỏi lập trình viên tự xây dựng nhiều chức năng từ đầu, tăng khối lượng công việc và yêu cầu kinh nghiệm cao hơn.

Thiếu tính năng bảo mật tích hợp: Express.js không có các tính năng bảo mật tích hợp như chống tấn công Cross Site Scripting (XSS) hoặc Cross-Site Request Forgery (CSRF). Lập trình viên cần tự triển khai hoặc sử dụng các thư viện bên ngoài để đảm bảo an toàn cho ứng dụng.

Quản lý lỗi không hiệu quả: Express.js không cung cấp cơ chế quản lý lỗi mạnh mẽ. Lập trình viên cần tự xử lý lỗi trong ứng dụng, điều này có thể phức tạp và tốn thời gian đối với các dự án lớn.[12][13]

2.4 RESTful API

2.4.1 Tổng quan về RESTful API

RESTful API (Representational State Transfer API) là một kiểu kiến trúc thiết kế dành cho việc xây dựng các dịch vụ web linh hoạt, đơn giản và dễ dàng tích hợp. Được giới thiệu bởi Roy Fielding vào năm 2000, RESTful API hoạt động dựa trên các nguyên tắc của REST - một mô hình giao tiếp không trạng thái (stateless) và có khả năng mở rộng cao.

RESTful API sử dụng các URL thân thiện và dễ hiểu để xác định tài nguyên, cùng với các định dạng dữ liệu phổ biến như JSON hoặc XML để trao đổi thông tin giữa máy chủ và máy khách. Điều này giúp RESTful API trở nên khả dụng và tương thích với nhiều ứng dụng khác nhau, tạo điều kiện thuận lợi cho việc tích hợp và phát triển phần mềm hiện đại.[14]

2.4.2 Các thành phần chính của RESTful API

API (Application Programming Interface): API là tập hợp các quy tắc và giao thức giúp một ứng dụng hoặc thành phần có thể tương tác với một ứng dụng hoặc thành phần khác. Trong RESTful API, các dữ liệu trả về thường được định dạng dưới dạng JSON hoặc XML, giúp các ứng dụng dễ dàng xử lý và hiển thị.

REST (Representational State Transfer) REST là một phong cách kiến trúc API dựa trên giao thức HTTP, sử dụng các phương thức HTTP như GET, POST, PUT, DELETE, và PATCH để tương tác với tài nguyên. Thay vì sử dụng các URL phức tạp, RESTful API đơn giản hóa bằng cách sử dụng các yêu cầu HTTP tiêu chuẩn để xử lý dữ liệu.

2.4.3 Các nguyên tắc của RESTful API

RESTful API được thiết kế dựa trên một tập hợp các nguyên tắc cốt lõi, đảm bảo hiệu quả, tính linh hoạt và khả năng mở rộng trong quá trình phát triển và triển khai.

Kiến trúc Client-Server (Client-Server Architecture):

RESTful API phân tách hoàn toàn giữa giao diện người dùng (**client**) và logic xử lý dữ liệu (**server**).

Việc phân tách này giúp client và server có thể phát triển độc lập, nâng cao khả năng bảo trì, nâng cấp và mở rộng hệ thống.

Ví dụ: Một ứng dụng mobile có thể thay thế giao diện mà không ảnh hưởng đến logic xử lý trên server.

Không lưu trạng thái (Statelessness):

Mỗi yêu cầu từ client đến server phải chứa đầy đủ thông tin cần thiết để xử lý yêu cầu đó.

Server không lưu trữ trạng thái của client giữa các yêu cầu, giúp tăng khả năng mở rộng và giảm tải bộ nhớ cho server.

Điều này cho phép các hệ thống phân tán và cân bằng tải hoạt động hiệu quả hơn.

Có thể lưu cache (Cacheable):

RESTful API hỗ trợ đánh dấu dữ liệu từ server là có thể lưu cache hoặc không.

Khi được lưu cache, client có thể tái sử dụng dữ liệu đã nhận mà không cần gửi yêu cầu mới tới server, giảm tải hệ thống và cải thiện hiệu suất.

Ví dụ: Thông tin về sản phẩm thường xuyên truy cập có thể được lưu cache để tăng tốc độ phản hồi.

Hệ thống phân lớp (Layered System):

RESTful API cho phép client giao tiếp với server mà không cần biết về số lượng lớp trung gian giữa chúng (ví dụ: proxy, load balancer, các server khác).

Điều này tăng cường bảo mật, khả năng mở rộng và tính linh hoạt trong quản lý hệ thống.

Ví dụ: Một proxy có thể được sử dụng để xử lý xác thực trước khi chuyển yêu cầu đến server chính.

Giao diện thống nhất (Uniform Interface):

API phải tuân theo một giao diện thống nhất để dễ dàng hiểu và sử dụng. Giao diện này bao gồm:

Resource Identification: Tài nguyên được định danh qua URL duy nhất. Ví dụ: `https://api.example.com/products/123` để truy xuất thông tin sản phẩm có ID 123.

Manipulation of Resources: Tài nguyên được thao tác thông qua biểu diễn như JSON hoặc XML.

Self-Descriptive Messages: Mỗi yêu cầu và phản hồi phải bao gồm đầy đủ thông tin cần thiết. Ví dụ: Tiêu đề HTTP (Content-Type, Authorization, ...) xác định định dạng dữ liệu hoặc quyền truy cập.

HATEOAS (Hypermedia as the Engine of Application State): Kết quả trả về từ server nên chứa các liên kết tới các thao tác liên quan. Ví dụ: Phản hồi từ server khi lấy thông tin sản phẩm có thể chứa liên kết để chỉnh sửa hoặc xóa sản phẩm đó.

Mã được tải về (Code on Demand) (*Không bắt buộc*):

Server có thể gửi mã (chẳng hạn như JavaScript) để client thực thi, mở rộng khả năng động của hệ thống.

Ví dụ: Server cung cấp một đoạn mã để client thực hiện xử lý giao diện mà không cần gửi thêm yêu cầu.[15]

2.4.4 Ưu điểm của RESTful API

Đơn giản hóa việc phát triển: RESTful API sử dụng các phương thức HTTP tiêu chuẩn, giúp lập trình viên dễ dàng triển khai và tập trung vào việc phát triển các tính năng chính của ứng dụng.

Khả năng mở rộng cao: Với RESTful API, các ứng dụng có thể dễ dàng mở rộng và thêm tính năng mới. Việc kết nối và trao đổi dữ liệu giữa các ứng dụng cũng trở nên thuận tiện, giúp hệ thống linh hoạt hơn trong việc tích hợp.

Tính độc lập của ứng dụng: RESTful API cho phép các ứng dụng hoạt động độc lập với nhau. Điều này giúp tối ưu hóa quá trình bảo trì và nâng cấp mà không làm ảnh hưởng đến toàn bộ hệ thống.

Khả năng tương thích đa dạng: RESTful API dễ dàng tích hợp với nhiều ứng dụng khác nhau, giúp các ứng dụng sử dụng cùng một API có thể tương tác với nhiều loại dữ liệu mà không cần viết lại mã nguồn.[14][15]

2.4.5 Nhược điểm của RESTful API

Chi phí phát triển và bảo trì: Việc thiết kế, triển khai và duy trì một API có thể yêu cầu nguồn lực lớn. Các nhà phát triển cần đầu tư nhiều thời gian và công sức để đảm bảo API hoạt động ổn định và đáp ứng các yêu cầu bảo mật, hiệu suất.

Vấn đề bảo mật: API mở ra một lớp giao tiếp mới trong kiến trúc hệ thống, dễ trở thành mục tiêu của các cuộc tấn công. RESTful API thường phải đối mặt với các rủi ro như XSS, CSRF, hoặc các cuộc tấn công man-in-the-middle. Việc bảo mật API đòi hỏi sử dụng các biện pháp bổ sung như mã hóa, xác thực và kiểm tra đầu vào.

Phụ thuộc vào giao thức HTTP: RESTful API hoạt động chủ yếu trên giao thức HTTP, điều này có thể hạn chế trong các trường hợp yêu cầu giao tiếp phức tạp hoặc khi cần hiệu suất cao hơn mà các giao thức khác như gRPC hoặc WebSocket có thể đáp ứng tốt hơn.[14][15]

2.5 JavaScript

2.5.1 Khái niệm

JavaScript là một ngôn ngữ lập trình mạnh mẽ, chủ yếu được sử dụng để tạo tính năng tương tác và động trên các trang web. Là một ngôn ngữ kịch bản phía client (client-side scripting language), JavaScript cho phép thực hiện các tác vụ như kiểm tra dữ liệu người dùng, thay đổi nội dung trang web mà không cần tải lại toàn bộ, và tương tác động với các phần tử trên trang.

Ban đầu, JavaScript được thiết kế để xử lý các chức năng cơ bản trong trình duyệt. Tuy nhiên, theo thời gian, ngôn ngữ này đã phát triển thành một công cụ đa năng, đủ khả năng xây dựng các ứng dụng web phức tạp, điều khiển giao diện người dùng và thực hiện giao tiếp không đồng bộ với máy chủ thông qua công nghệ AJAX (Asynchronous JavaScript and XML).

2.5.2 Ứng dụng

JavaScript là một ngôn ngữ linh hoạt với nhiều ứng dụng trong lập trình, từ frontend đến backend và cả phát triển ứng dụng di động hoặc desktop:

Tạo giao diện người dùng động (Dynamic User Interface):

Cho phép thay đổi nội dung trang web mà không cần tải lại toàn bộ trang, thường thấy trong các ứng dụng một trang (Single Page Applications - SPA).

Hỗ trợ tạo các hiệu ứng trực quan như hiển thị/ẩn nội dung, hoạt ảnh, hoặc chuyển động.

Xử lý các sự kiện từ người dùng, như nhấp chuột, nhập liệu, hoặc di chuột.

Kiểm tra và xử lý dữ liệu phía client (Client-side Validation):

Kiểm tra dữ liệu đầu vào từ người dùng trên biểu mẫu trước khi gửi tới máy chủ, giảm tải cho server và cải thiện trải nghiệm người dùng.

Đảm bảo dữ liệu được nhập chính xác và đầy đủ ngay tại trình duyệt.

Giao tiếp không đồng bộ với máy chủ (Asynchronous Communication):

Sử dụng AJAX hoặc Fetch API để trao đổi dữ liệu với máy chủ mà không cần tải lại trang.

Tích hợp các dữ liệu từ dịch vụ bên thứ ba hoặc API RESTful.

Hiển thị dữ liệu theo thời gian thực, chẳng hạn như thông tin thời tiết hoặc cập nhật trạng thái.

Xây dựng ứng dụng web phức tạp:

Sử dụng các framework như React, Angular, hoặc Vue.js để phát triển SPA hoặc ứng dụng quản lý phức tạp.

Xây dựng các ứng dụng thời gian thực như trò chuyện trực tuyến, công cụ cộng tác, hoặc bảng điều khiển (dashboard) thông qua WebSockets.

Phát triển ứng dụng di động và desktop:

Xây dựng ứng dụng di động đa nền tảng bằng React Native, Ionic, hoặc Flutter for Web.

Tạo ứng dụng desktop bằng Electron (như Slack hoặc Visual Studio Code).

Phát triển trò chơi (Game Development):

Dùng các thư viện như Phaser.js hoặc Babylon.js để tạo trò chơi chạy trên trình duyệt.

Xây dựng trò chơi 2D/3D bằng HTML5 Canvas hoặc WebGL.

Xây dựng hệ thống backend:

Sử dụng Node.js để phát triển máy chủ hoặc API backend.

Tạo các dịch vụ web hiệu năng cao, xử lý dữ liệu nhanh chóng và tiết kiệm tài nguyên.

2.5.3 Ưu điểm của JavaScript

Dễ học và sử dụng: Cú pháp đơn giản, gần gũi với người mới học lập trình.

JavaScript được hỗ trợ sẵn trên tất cả các trình duyệt hiện đại, không cần cài đặt phần mềm bổ sung.

Tính linh hoạt và đa dụng: Đa nền tảng: Chạy trên mọi trình duyệt, hệ điều hành và cả trên máy chủ thông qua Node.js. Đa mục đích: Dùng để phát triển giao diện frontend, backend, ứng dụng di động, desktop và cả trò chơi.

Hiệu suất cao: Thực thi mã trực tiếp trên trình duyệt, giảm tải cho máy chủ và tăng tốc độ xử lý. Hỗ trợ giao tiếp không đồng bộ với máy chủ, cải thiện trải nghiệm người dùng.

Cộng đồng lớn và tài nguyên phong phú: Có một hệ sinh thái khổng lồ với nhiều thư viện và framework phổ biến như React, Angular, Vue.js. Cộng đồng hỗ trợ mạnh mẽ, giúp lập trình viên dễ dàng tìm kiếm tài liệu, giải pháp và học hỏi từ các ví dụ thực tế.

Tích hợp dễ dàng: Tương thích với hầu hết các công nghệ và ngôn ngữ lập trình khác. Hỗ trợ tích hợp API và dịch vụ từ bên thứ ba một cách nhanh chóng

2.5.4 Nhược điểm của JavaScript

Bảo mật không cao: JavaScript chạy trực tiếp trên trình duyệt, dễ bị lợi dụng để thực hiện các cuộc tấn công XSS (Cross-Site Scripting) nếu không xử lý đúng cách. Dễ bị tấn công CSRF (Cross-Site Request Forgery), cho phép hacker thực hiện các hành động trái phép.

Phụ thuộc vào trình duyệt: Hiệu suất của JavaScript bị giới hạn bởi khả năng xử lý của trình duyệt, đặc biệt là trên các thiết bị cũ hoặc cấu hình thấp. Một số người dùng có thể tắt JavaScript trên trình duyệt, làm các tính năng động không hoạt động.

Quản lý lỗi khó khăn: Là một ngôn ngữ không biên dịch, JavaScript không phát hiện lỗi trong quá trình viết mã, khiến lỗi chỉ được phát hiện khi mã được chạy, gây khó khăn trong việc kiểm tra trước khi triển khai.

Thiếu chuẩn hóa: Không phải tất cả các trình duyệt đều hỗ trợ JavaScript giống nhau, buộc lập trình viên phải kiểm tra và tối ưu hóa mã cho các nền tảng khác nhau.

Quá phụ thuộc vào trình duyệt: Một số tính năng chỉ khả dụng trên các trình duyệt mới, đòi hỏi lập trình viên phải cung cấp giải pháp thay thế cho các trình duyệt cũ.

2.6 MongoDB

2.6.1 Khái niệm

MongoDB là một cơ sở dữ liệu NoSQL hướng tài liệu, cho phép lưu trữ dữ liệu dưới dạng JSON hoặc BSON với cấu trúc linh hoạt. Khác với các cơ sở dữ liệu quan hệ (Relational Database), MongoDB không sử dụng cấu trúc bảng cứng nhắc, mà thay vào đó lưu trữ dữ liệu theo tài liệu, giúp xử lý hiệu quả các cấu trúc dữ liệu phức tạp và phi cấu trúc.

Ngoài ra, MongoDB Atlas, một giải pháp Database-as-a-Service (DBaaS) của MongoDB, cung cấp nền tảng đám mây mạnh mẽ và chi phí hợp lý, phù hợp với các doanh nghiệp từ nhỏ đến lớn. Atlas được đánh giá cao nhờ khả năng quản lý, bảo mật và tích hợp dễ dàng.

2.6.2 Tính năng nổi bật của MongoDB

Cấu trúc dữ liệu linh hoạt (Document-Oriented): MongoDB lưu trữ dữ liệu dưới dạng document JSON/BSON, cho phép biểu diễn các cấu trúc dữ liệu đa dạng mà không cần sử dụng bảng hoặc mối quan hệ cứng nhắc như cơ sở dữ liệu SQL.

Không có schema cố định (Schema-less): Không yêu cầu một schema cố định, MongoDB cho phép dễ dàng thêm, sửa hoặc xóa các trường dữ liệu mà không cần thay đổi cấu trúc cơ sở dữ liệu, giúp linh hoạt trong việc phát triển và mở rộng.

Hỗ trợ phân mảnh dữ liệu (Sharding): MongoDB cho phép phân mảnh dữ liệu trên nhiều máy chủ để xử lý lượng dữ liệu lớn hoặc lưu lượng truy cập cao, đảm bảo hiệu suất và khả năng mở rộng.

Sao chép dữ liệu (Replication): Thông qua Replica Sets, MongoDB hỗ trợ sao chép dữ liệu tự động, tăng cường độ tin cậy và khả năng phục hồi sau sự cố.

Truy vấn và phân tích mạnh mẽ: MongoDB cung cấp các công cụ như Aggregation Framework để thực hiện các truy vấn linh hoạt, tìm kiếm toàn văn (Full-text Search), và phân tích dữ liệu phức tạp một cách hiệu quả.

Tích hợp đa dạng: Hỗ trợ nhiều ngôn ngữ lập trình phổ biến như JavaScript, Python, Java, Node.js, và các công cụ phân tích dữ liệu như Tableau, MongoDB dễ dàng tích hợp vào hệ thống hiện có.

Hỗ trợ dữ liệu không gian địa lý (Geospatial Data): MongoDB cung cấp các tính năng xử lý dữ liệu địa lý, rất phù hợp cho các ứng dụng liên quan đến bản đồ, giao vận, hoặc tìm kiếm địa điểm.

2.6.3 Ưu điểm của MongoDB

Hiệu suất cao và dễ mở rộng: Thiết kế theo chiều ngang (Horizontal Scaling) giúp MongoDB xử lý tốt dữ liệu lớn và lưu lượng truy cập cao, phù hợp cho các hệ thống cần hiệu suất cao.

Dễ học và triển khai nhanh: Sử dụng JSON/BSON gần gũi với lập trình viên, MongoDB giúp giảm thời gian học tập và dễ dàng triển khai ngay cả đối với người mới.

Linh hoạt trong quản lý dữ liệu: Không yêu cầu schema cố định, MongoDB cho phép thêm hoặc sửa dữ liệu một cách tự do mà không cần thay đổi toàn bộ cấu trúc cơ sở dữ liệu.

Thích hợp cho dữ liệu phi cấu trúc và bán cấu trúc: MongoDB là lựa chọn lý tưởng cho các ứng dụng lưu trữ nội dung đa dạng như hồ sơ người dùng, dữ liệu IoT, hoặc tài liệu không có định dạng cố định.

Cộng đồng lớn và tài liệu phong phú: Cộng đồng phát triển đông đảo cùng tài liệu chi tiết giúp nhà phát triển dễ dàng tìm kiếm hỗ trợ và giải pháp cho các vấn đề gặp phải.

Khả năng xử lý dữ liệu phức tạp: Thông qua Aggregation Framework, MongoDB cho phép thực hiện các thao tác phức tạp như nhóm, sắp xếp, và tổng hợp dữ liệu ngay trong cơ sở dữ liệu mà không cần xử lý trên ứng dụng.

2.6.4 Nhược điểm của MongoDB

Tiêu thụ nhiều bộ nhớ: Do sử dụng định dạng BSON, MongoDB thường tiêu tốn nhiều bộ nhớ hơn so với các cơ sở dữ liệu quan hệ, đặc biệt khi lưu trữ dữ liệu lớn.

Không tối ưu cho giao dịch phức tạp: Dù đã hỗ trợ giao dịch nhiều tài liệu từ phiên bản 4.0, MongoDB vẫn chưa phù hợp cho các ứng dụng yêu cầu giao dịch phức tạp hoặc tính nhất quán dữ liệu cao như ngân hàng.

Thiếu tính toàn vẹn dữ liệu: Với cấu trúc không ràng buộc schema, việc đảm bảo tính toàn vẹn dữ liệu phụ thuộc vào ứng dụng, dễ dẫn đến lỗi nếu không kiểm soát tốt.

Không hỗ trợ JOIN trực tiếp: MongoDB không có chức năng JOIN như SQL, việc xử lý các mối quan hệ phức tạp giữa dữ liệu đòi hỏi thiết kế hợp lý và đôi khi phải thực hiện nhiều truy vấn thủ công.

Chi phí cao khi triển khai sharding: Cấu hình và quản lý phân mảnh (sharding) yêu cầu kiến thức chuyên sâu, có thể gây phức tạp và tốn kém khi triển khai trong môi trường lớn.

Hạn chế trong bảo mật mặc định: Các phiên bản cũ của MongoDB thường có cấu hình bảo mật mặc định yếu, yêu cầu nhà phát triển cấu hình bổ sung để đảm bảo an toàn dữ liệu.[16]

2.7 Các nghiệp vụ liên quan đến đề tài

2.7.1 Nghiệp vụ quản lý sản phẩm

Quản lý sản phẩm là một trong những chức năng cốt lõi của bất kỳ hệ thống thương mại điện tử nào, đặc biệt là trong lĩnh vực bán thiết bị di động. Các nghiệp vụ dưới đây đảm bảo rằng sản phẩm được hiển thị chính xác, dễ quản lý và phù hợp với nhu cầu kinh doanh.

2.7.1.1 Thêm mới sản phẩm

Quản trị viên được phép thêm sản phẩm mới vào hệ thống. Trong quá trình thêm sản phẩm, quản trị viên cần cung cấp đầy đủ thông tin chi tiết, bao gồm:

Tên sản phẩm: Đây là trường bắt buộc, giúp người dùng dễ dàng nhận biết và tìm kiếm sản phẩm.

Mô tả chi tiết: Bao gồm các thông tin về cấu hình kỹ thuật, tính năng nổi bật, kích thước, dung lượng pin, và các thông tin khác.

Hình ảnh minh họa: Hệ thống hỗ trợ tải lên nhiều hình ảnh để minh họa rõ ràng về thiết kế, kích thước, và màu sắc của sản phẩm.

Giá cả: Bao gồm giá gốc, giá khuyến mãi (nếu có), hoặc chính sách giá đặc biệt như giảm giá khi mua số lượng lớn.

Danh mục: Sản phẩm được phân loại vào các danh mục như “Điện thoại”, “Phụ kiện”, “Máy tính bảng”.

Thương hiệu: Như Apple, Samsung, Xiaomi, giúp khách hàng lọc theo thương hiệu.

2.7.1.2 Cập nhật sản phẩm

Sau khi sản phẩm đã được thêm vào hệ thống, quản trị viên có thể thực hiện các chỉnh sửa cần thiết như:

Thay đổi thông tin mô tả nếu có cải tiến sản phẩm.

Điều chỉnh giá bán trong các chiến dịch khuyến mãi.

Cập nhật số lượng tồn kho khi nhận hàng bổ sung.

2.7.1.3 Xóa hoặc ẩn sản phẩm

Nếu một sản phẩm không còn kinh doanh, quản trị viên có thể:

Xóa vĩnh viễn: Khi không cần giữ lại thông tin sản phẩm.

Ẩn khỏi hiển thị: Khi muốn lưu trữ thông tin sản phẩm cho mục đích thống kê hoặc tham khảo.

2.7.1.4 Hiển thị và tìm kiếm sản phẩm

Người dùng cuối có thể xem danh sách sản phẩm với các tính năng hỗ trợ như:

Tìm kiếm nâng cao: Tìm kiếm theo từ khóa, danh mục, hoặc thương hiệu.

Bộ lọc: Lọc sản phẩm theo giá cả, tính năng, hoặc đánh giá từ khách hàng.

Hiển thị nổi bật: Các sản phẩm mới, sản phẩm bán chạy được ưu tiên hiển thị để thu hút khách hàng.

2.7.2 Nghiệp vụ quản lý người dùng

Quản lý người dùng không chỉ đảm bảo tính an toàn và cá nhân hóa trải nghiệm khách hàng mà còn hỗ trợ quản lý quyền hạn trong hệ thống.

2.7.2.1 Đăng ký tài khoản

Khách hàng khi đăng ký tài khoản cần cung cấp:

Email hợp lệ: Dùng để xác thực danh tính và gửi thông báo đơn hàng.

Mật khẩu bảo mật: Đảm bảo có độ phức tạp để tránh rủi ro bảo mật.

Số điện thoại và địa chỉ nhận hàng: Thông tin này có thể được nhập bổ sung trong lần đặt hàng đầu tiên.

2.7.2.2 Đăng nhập và phân quyền

Hệ thống sử dụng JWT (JSON Web Token) để quản lý quyền truy cập:

Khách hàng: Có thể duyệt sản phẩm, thêm vào giỏ hàng, đặt hàng và theo dõi đơn hàng.

Quản trị viên: Có quyền truy cập các chức năng như thêm sản phẩm, quản lý đơn hàng, thống kê doanh thu.

2.7.2.3 Cập nhật thông tin cá nhân

Người dùng có thể cập nhật các thông tin: Đặt lại mật khẩu hoặc khôi phục tài khoản.

2.7.3 Nghiệp vụ giỏ hàng

Giỏ hàng là nơi khách hàng quản lý danh sách sản phẩm muốn mua, từ lúc thêm sản phẩm đến khi hoàn tất thanh toán.

2.7.3.1 Thêm sản phẩm vào giỏ

Người dùng có thể thêm bất kỳ sản phẩm nào vào giỏ hàng với số lượng mong muốn. Hệ thống kiểm tra xem số lượng đặt hàng có vượt quá số lượng tồn kho hay không.

2.7.3.2 *Chỉnh sửa giỏ hàng*

Trong giỏ hàng, khách hàng có thể thực hiện các thao tác sau: Tăng/giảm số lượng sản phẩm. Xóa sản phẩm nếu không còn nhu cầu mua.

2.7.3.3 *Hiển thị tổng chi phí*

Hệ thống tự động tính toán tổng chi phí, bao gồm: Tổng tiền sản phẩm. Thuế (nếu có). Phí vận chuyển, dựa trên địa chỉ giao hàng của khách hàng.

2.7.4 *Nghiệp vụ đặt hàng*

Quy trình đặt hàng là bước quan trọng để chuyển đổi từ khách hàng tiềm năng thành khách hàng thực sự.

2.7.4.1 *Tạo đơn hàng*

Khi người dùng xác nhận giỏ hàng, hệ thống tự động tạo đơn hàng với các thông tin:

Sản phẩm: Chi tiết về từng sản phẩm trong đơn hàng.

Địa chỉ giao hàng: Người dùng có thể chọn địa chỉ mặc định hoặc thêm địa chỉ mới.

Phương thức thanh toán: Thanh toán trực tuyến

2.7.4.2 *Quản lý trạng thái đơn hàng*

Mỗi đơn hàng được gán trạng thái để dễ dàng theo dõi:

Đã tạo: Đơn hàng vừa được đặt.

Đang xử lý: Đơn hàng đang được kiểm tra hoặc đóng gói.

Đang vận chuyển: Đơn hàng đã được gửi đến đơn vị vận chuyển.

Hoàn thành: Giao hàng thành công.

Hủy: Đơn hàng bị hủy bởi khách hàng hoặc hệ thống.

2.7.5 *Thanh toán*

Hệ thống hỗ trợ tích hợp các cổng thanh toán phổ biến: Thanh toán trực tuyến: Sử dụng Stripe hoặc VNPAY.

2.7.6 Nghiệp vụ quản lý kho

Đảm bảo số lượng tồn kho được cập nhật chính xác và liên tục là điều kiện tiên quyết để vận hành hệ thống hiệu quả.

2.7.6.1 Theo dõi tồn kho

Hệ thống hiển thị số lượng tồn kho theo từng sản phẩm, tổng số lượng hiện tại.

2.7.6.2 Tự động cập nhật sau bán hàng

Khi đơn hàng được xác nhận, số lượng sản phẩm trong kho tự động giảm, tránh tình trạng bán quá số lượng.

2.7.7 Nghiệp vụ thống kê và báo cáo

Hệ thống cung cấp công cụ hỗ trợ quản trị viên trong việc phân tích và đánh giá hiệu quả kinh doanh. Thống kê doanh thu và hiển thị doanh thu theo tháng hoặc năm. Liệt kê các danh sách sản phẩm bán chạy.

2.8 Các công trình nghiên cứu liên quan đến đề tài

2.8.1 Báo cáo thực tập: Lập trình website bán hàng trực tuyến với ReactJS, NodeJS và ExpressJS

Vào cuối năm 2023, nhóm sinh viên chuyên ngành Công nghệ Thông tin, trong quá trình thực tập tại Công ty TNHH Ingreetech, đã hoàn thành một báo cáo nghiên cứu tập trung vào việc xây dựng hệ thống thương mại điện tử hiện đại. Dự án sử dụng ReactJS cho giao diện frontend, NodeJS làm backend, và ExpressJS làm framework chính để phát triển API. Hệ thống được thiết kế tích hợp các chức năng quan trọng như quản lý sản phẩm (thêm, sửa, xóa), quản lý giỏ hàng, phân quyền người dùng, và đặc biệt là xử lý thanh toán trực tuyến thông qua các cổng thanh toán phổ biến như VNPay và PayPal. Nghiên cứu cũng chú trọng vào việc tối ưu hóa hiệu năng hệ thống, đảm bảo khả năng xử lý hơn 500 yêu cầu đồng thời một cách ổn định mà không xảy ra lỗi. Điểm nổi bật của dự án là việc đề cao tính bảo mật dữ liệu giao dịch và khả năng ứng dụng thực tế, hướng đến phục vụ các doanh nghiệp vừa và nhỏ. Nhờ vào những thành tựu đạt được, báo cáo đã nhận được sự đánh giá cao từ phía công ty thực tập và được khuyến nghị phát triển thêm để triển khai trong môi trường sản xuất thực tế.

2.8.2 Báo cáo thực tập: Phát triển ứng dụng web với ReactJS và tích hợp NodeJS

Vào tháng 8 năm 2023, một sinh viên chuyên ngành Khoa học Máy tính thuộc Đại học FPT, trong thời gian thực tập tại Công ty Codosa Tech, đã hoàn thành một công trình nghiên cứu tập trung vào việc phát triển giao diện người dùng hiện đại, thân thiện và tối ưu hóa trải nghiệm khách hàng bằng ReactJS. Báo cáo cũng đề cập đến việc sử dụng NodeJS để xây dựng một API cơ bản hỗ trợ các chức năng như quản lý sản phẩm, tìm kiếm nhanh, và bộ lọc nâng cao. Đặc biệt, công trình đã tích hợp thành công bản đồ giao hàng, cho phép hiển thị và tối ưu hóa lộ trình giao nhận. Nhờ vào việc tối ưu hóa giao diện, hệ thống đạt điểm hiệu suất cao trên Google PageSpeed với điểm số vượt 90/100, đảm bảo khả năng hoạt động mượt mà trên cả thiết bị di động và desktop. Nghiên cứu được đánh giá là một mẫu ứng dụng xuất sắc, cung cấp nền tảng mạnh mẽ để phát triển thêm các tính năng phức tạp hơn trong tương lai.

2.8.3 Tài liệu thực hành: Hướng dẫn lập trình website bán hàng trực tuyến với NodeJS và ReactJS

Vào đầu năm 2024, nhóm sinh viên chuyên ngành Công nghệ Phần mềm thuộc Đại học Khoa học Tự nhiên TP.HCM đã thực hiện một nghiên cứu tập trung vào việc phát triển website bán hàng trực tuyến. Công trình này sử dụng ReactJS làm công cụ chính để xây dựng giao diện frontend, NodeJS đảm nhận vai trò backend, và MongoDB để quản lý và lưu trữ dữ liệu. Nghiên cứu đặc biệt chú trọng vào việc tích hợp các tính năng thời gian thực, bao gồm cập nhật trạng thái đơn hàng, phân quyền người dùng (khách hàng và quản trị viên), và quản lý giỏ hàng. Ngoài ra, nhóm thực hiện cũng quan tâm đến các yếu tố bảo mật cơ bản và tối ưu hóa hiệu năng để đảm bảo hệ thống hoạt động ổn định và an toàn. Kết quả nghiên cứu là một website hoàn thiện, đáp ứng đầy đủ yêu cầu triển khai trong thực tế, đặc biệt phù hợp với các doanh nghiệp vừa và nhỏ. Tài liệu nghiên cứu không chỉ được đánh giá cao bởi tính ứng dụng mà còn được chọn làm giáo trình tham khảo cho các khóa học thực hành lập trình web của sinh viên trong các năm tiếp theo.

CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả hệ thống

Website bán thiết bị di động được thiết kế và xây dựng nhằm tạo ra một nền tảng thương mại điện tử hiện đại, cung cấp trải nghiệm mua sắm toàn diện cho người dùng. Mục tiêu của hệ thống là đáp ứng các nhu cầu quan trọng trong việc tìm kiếm, so sánh và mua sắm thiết bị di động, đồng thời hỗ trợ các doanh nghiệp quản lý sản phẩm và giao dịch một cách hiệu quả.

Hệ thống này không chỉ tập trung vào giao diện trực quan, dễ sử dụng mà còn chú trọng vào hiệu năng, bảo mật và khả năng mở rộng, đảm bảo đáp ứng cả nhu cầu cá nhân lẫn doanh nghiệp trong môi trường cạnh tranh khốc liệt. Hệ thống mang lại những lợi ích nổi bật:

Đối với người dùng(user):

Duyệt danh mục sản phẩm dễ dàng, với các bộ lọc và công cụ tìm kiếm.

Xem chi tiết sản phẩm bao gồm mô tả, thông số kỹ thuật và đánh giá từ khách hàng khác.

Thêm sản phẩm vào giỏ hàng, thanh toán trực tuyến an toàn, và theo dõi trạng thái đơn hàng.

Đối với quản trị viên(admin):

Quản lý toàn bộ danh mục sản phẩm: thêm, chỉnh sửa, hoặc xóa sản phẩm.

Theo dõi và xử lý đơn hàng trong hệ thống, đảm bảo mọi giao dịch được thực hiện hiệu quả.

Kiểm duyệt và quản lý các đánh giá từ người dùng để duy trì chất lượng.

Hệ thống được chia thành hai phần chính:

Frontend: Xây dựng bằng ReactJS, đảm bảo giao diện trực quan và tương tác tốt. Giao diện được thiết kế với tính thích ứng (responsive) cao, hoạt động mượt mà trên cả thiết bị di động và máy tính để bàn.

Backend: Được triển khai bằng NodeJS và ExpressJS, tích hợp với MongoDB để quản lý dữ liệu. Hệ thống API RESTful được xây dựng để đảm bảo giao tiếp giữa frontend và backend nhanh chóng, bảo mật với sự hỗ trợ của JWT.

3.2 Xác định các yêu cầu chức năng của hệ thống

Website được thiết kế để đáp ứng các nhu cầu sử dụng cơ bản và nâng cao của người dùng và quản trị viên. Các chức năng chính bao gồm:

Quản lý người dùng:

Đăng ký tài khoản mới với các thông tin như email, tên người dùng và mật khẩu.

Đăng nhập và xác thực phiên làm việc thông qua JSON Web Token (JWT).

Quản lý thông tin cá nhân, bao gồm thay đổi mật khẩu và cập nhật thông tin liên lạc.

Quản lý sản phẩm:

Hiển thị danh sách sản phẩm với khả năng phân loại và sắp xếp theo các tiêu chí như giá, tên hoặc đánh giá.

Tìm kiếm nâng cao với bộ lọc theo danh mục, giá và các thuộc tính khác.

Xem chi tiết sản phẩm bao gồm hình ảnh, mô tả, thông số kỹ thuật và đánh giá từ người dùng.

Hệ thống khuyến mãi và mã giảm giá, hỗ trợ áp dụng mã khuyến mãi hoặc giảm giá khi thanh toán.

Giỏ hàng và thanh toán:

Thêm sản phẩm vào giỏ hàng, chỉnh sửa số lượng hoặc xóa sản phẩm khỏi giỏ hàng.

Hỗ trợ phương thức thanh trực tuyến Stripe.

Xử lý thanh toán an toàn với hệ thống mã hóa dữ liệu và xác thực giao dịch.

Quản lý đơn hàng:

Theo dõi trạng thái đơn hàng từ lúc đặt hàng đến khi giao hàng thành công.

Cập nhật trạng thái đơn hàng bởi quản trị viên, đảm bảo thông tin luôn được cập nhật kịp thời.

Đánh giá và phản hồi:

Người dùng có thể để lại nhận xét và đánh giá về sản phẩm đã mua.

Quản trị viên có quyền kiểm duyệt các đánh giá để đảm bảo chất lượng nội dung.

Bảo mật và hiệu năng:

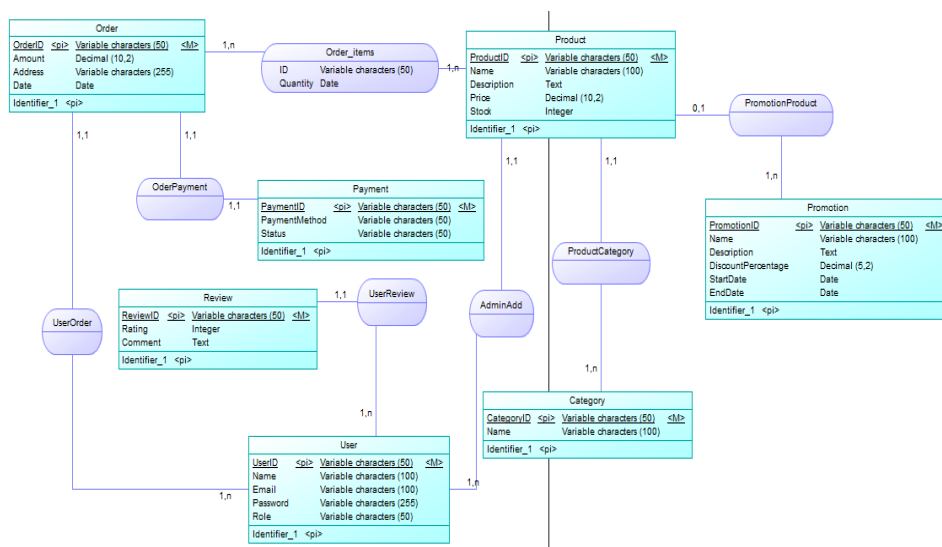
Sử dụng HTTPS để mã hóa giao tiếp giữa client và server.

Tích hợp các biện pháp chống tấn công XSS, CSRF để bảo vệ dữ liệu người dùng.

Tối ưu hóa tốc độ tải trang và truy vấn cơ sở dữ liệu để đảm bảo trải nghiệm tốt nhất cho người dùng.

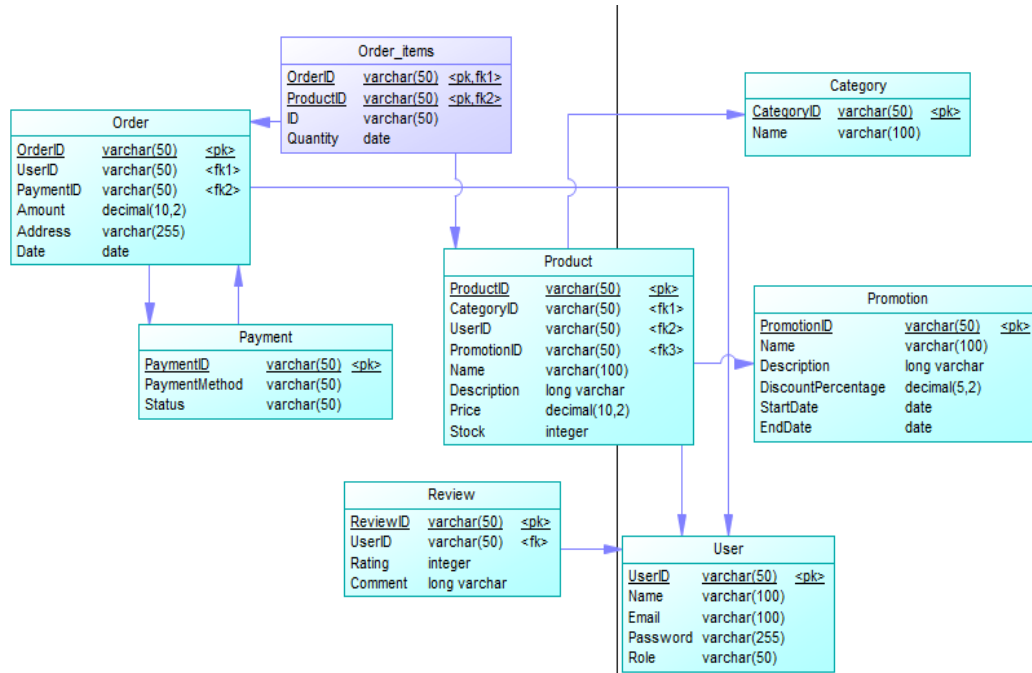
3.3 Thiết kế dữ liệu hệ thống

3.3.1 Mô hình dữ liệu mức quan niệm



Hình 1. Mô hình dữ liệu mức quan niệm

3.3.2 Mô hình dữ liệu mức logic



Hình 2. Mô hình dữ liệu mức logic

3.3.3 Các thực thể

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	OrderID	Mã đơn hàng(PK)	VARCHAR(50)
2.	UserID	Mã người dùng liên kết (FK)	VARCHAR(50)
3.	PaymentID	Mã thanh toán liên kết(FK)	VARCHAR(50)
4.	Amount	Tổng số tiền của đơn hàng	DECIMAL(10, 2)
5.	Address	Địa chỉ giao hàng	VARCHAR(255)
6.	Date	Ngày đặt hàng	DATE

Bảng 1. Order

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	ID	Mã đơn hàng(PK)	VARCHAR(50)
2.	OrderID	Mã đơn hàng liên kết (FK)	VARCHAR(50)
3.	ProductID	Mã sản phẩm trong đơn hàng (FK)	VARCHAR(50)
4.	Quantity	Ngày đặt hàng	DATA

Bảng 2. Order_items

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	PaymentID	Mã giao dịch thanh toán (PK)	VARCHAR(50)
2.	OrderID	Mã đơn hàng liên kết (FK)	VARCHAR(50)
3.	PaymentMethod	Phương thức thanh toán (Stripe)	VARCHAR(50)
4.	Status	Trạng thái thanh toán (Pending, Completed, Failed)	VARCHAR(50)

Bảng 3. Payment

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	UserID	Mã người dùng (PK)	VARCHAR(50)
2.	Name	Tên người dùng	VARCHAR(100)
3.	Email	Email đăng nhập của người dùng	VARCHAR(100)
4.	Password	Mật khẩu người dùng	VARCHAR(255)
5.	Role	Vai trò của người dùng (Admin, Customer)	VARCHAR(50)

Bảng 4. User

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	ProductID	Mã sản phẩm (PK)	VARCHAR(50)
2.	Name	Tên sản phẩm	VARCHAR(100)
3.	Description	Mô tả sản phẩm	TEXT
4.	Price	Giá sản phẩm	DECIMAL(10, 2)
5.	CategoryID	Mã danh mục sản phẩm liên kết(FK)	VARCHAR(50)
6.	Stock	Số lượng tồn của sản phẩm	INT

Bảng 5. Product

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	CategoryID	Mã danh mục (PK)	VARCHAR(50)
2.	Name	Tên danh mục	VARCHAR(100)

Bảng 6. Category

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	ReviewID	Mã đánh giá (PK)	VARCHAR(50)
2.	ProductID	Mã sản phẩm liên kết(FK)	VARCHAR(50)
3.	UserID	Mã người dùng liên kết(FK)	VARCHAR(50)
4.	Rating	Điểm đánh giá (1-5 sao)	INT
5.	Comment	Nhận xét của người dùng về sản phẩm	TEXT

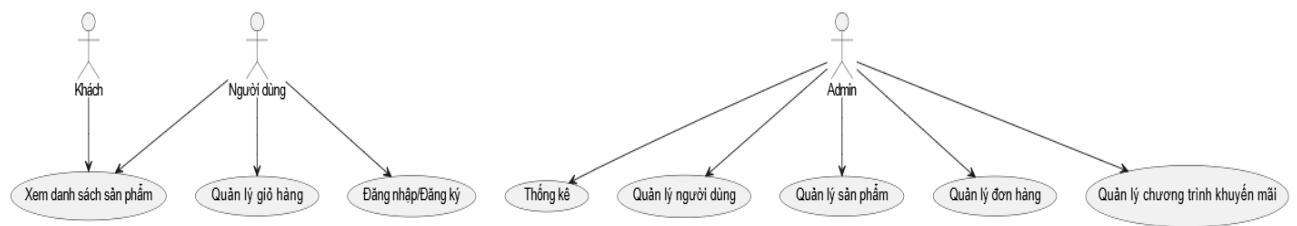
Bảng 7. Review

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1.	PromotionID	Mã khuyến mãi (PK)	VARCHAR(50)
2.	Name	Tên chương trình khuyến mãi	VARCHAR(100)
3.	Description	Mô tả chương trình khuyến mãi	TEXT
4.	DiscountPercentage	Phần trăm giảm giá	DECIMAL(5, 2)
5.	StartDate	Ngày bắt đầu áp dụng khuyến mãi	DATE
6.	EndDate	Ngày kết thúc áp dụng khuyến mãi	DATE

Bảng 8. Promotion

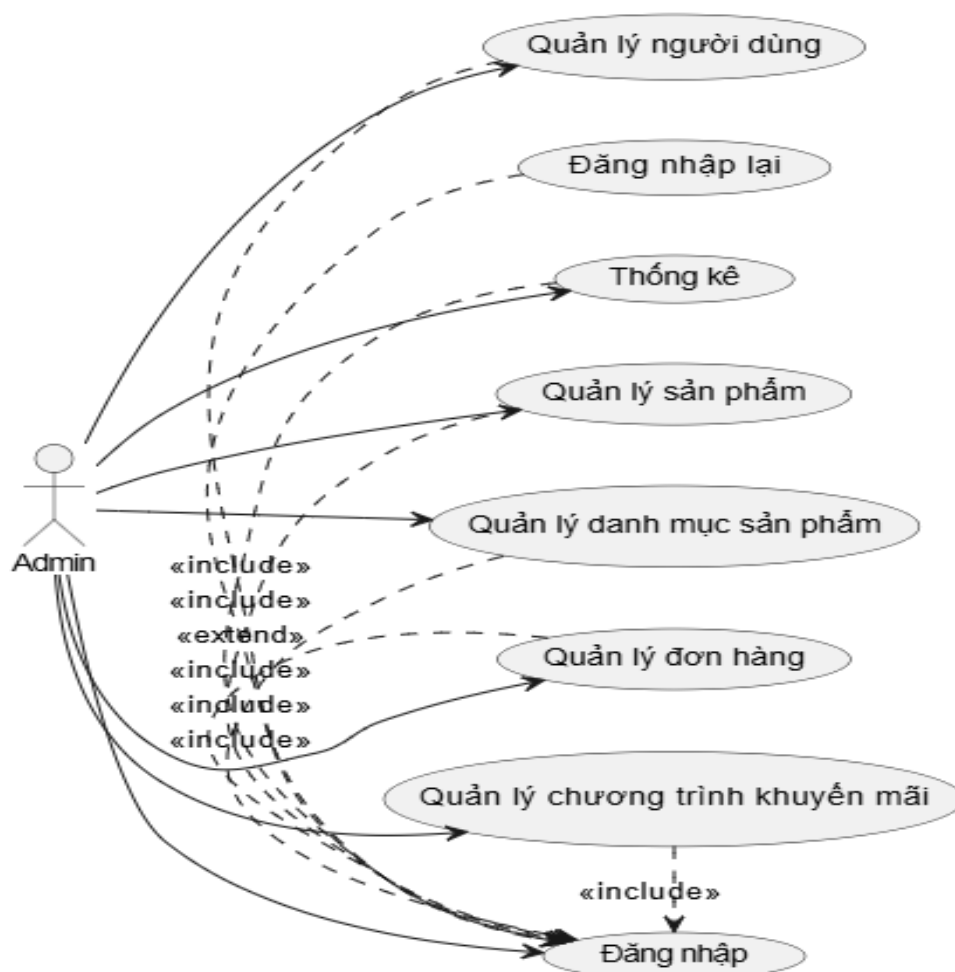
3.4 Thiết kế xử lý hệ thống

3.4.1 Biểu đồ Use Case tổng quát



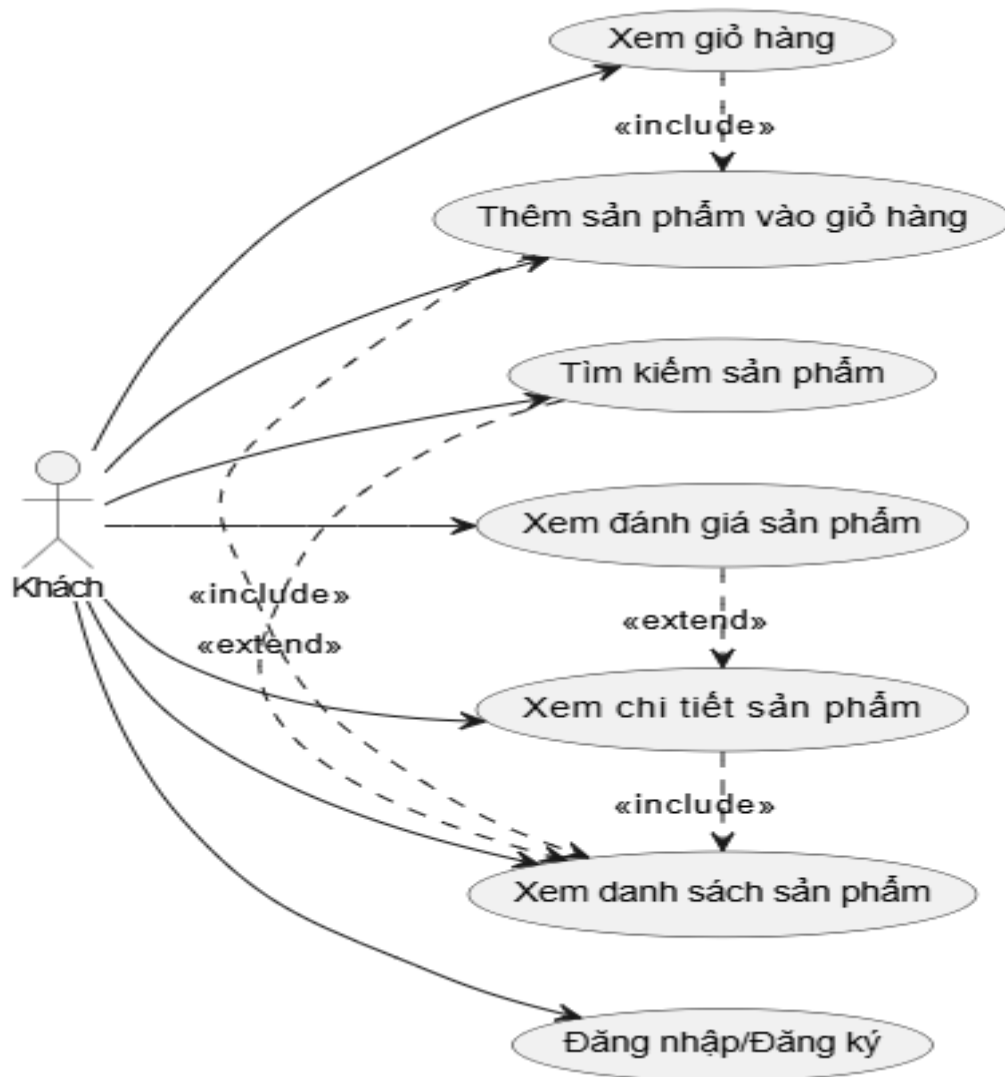
Hình 3. Use Case tổng quát hệ thống

3.4.2 Use Case tổng quan Admin



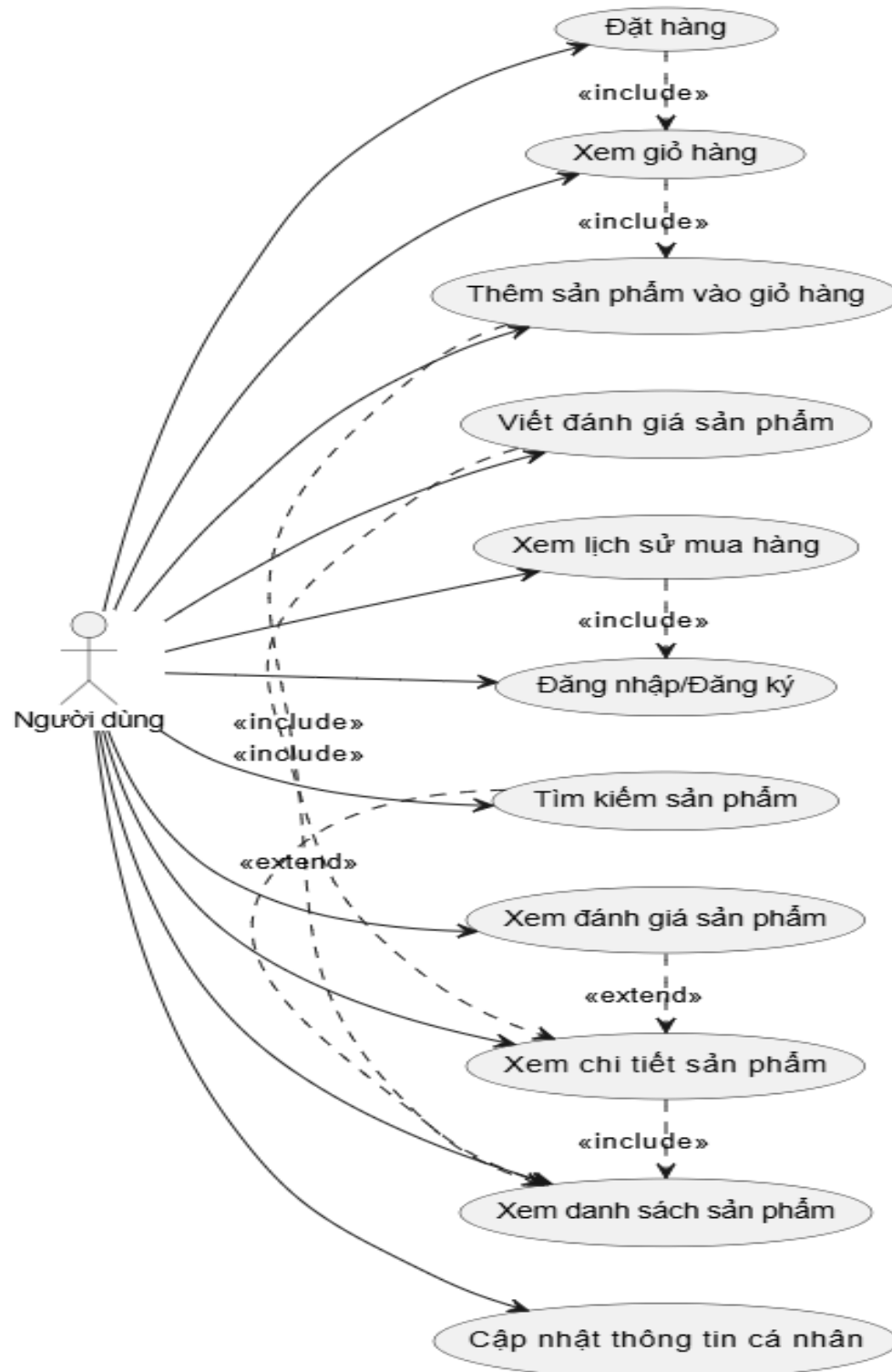
Hình 4. Use Case tổng quan Admin

3.4.3 Use Case tổng quan của khách



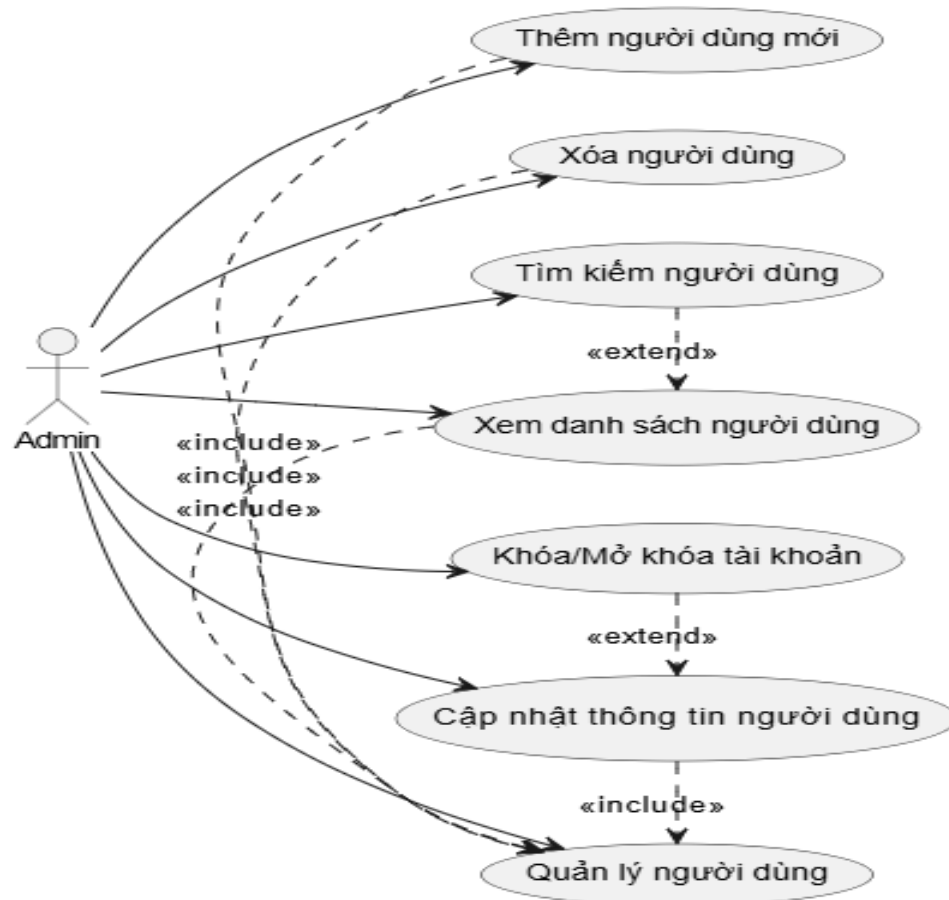
Hình 5. Use Case tổng quan của khách

3.4.4 Use Case tổng quan của người dùng



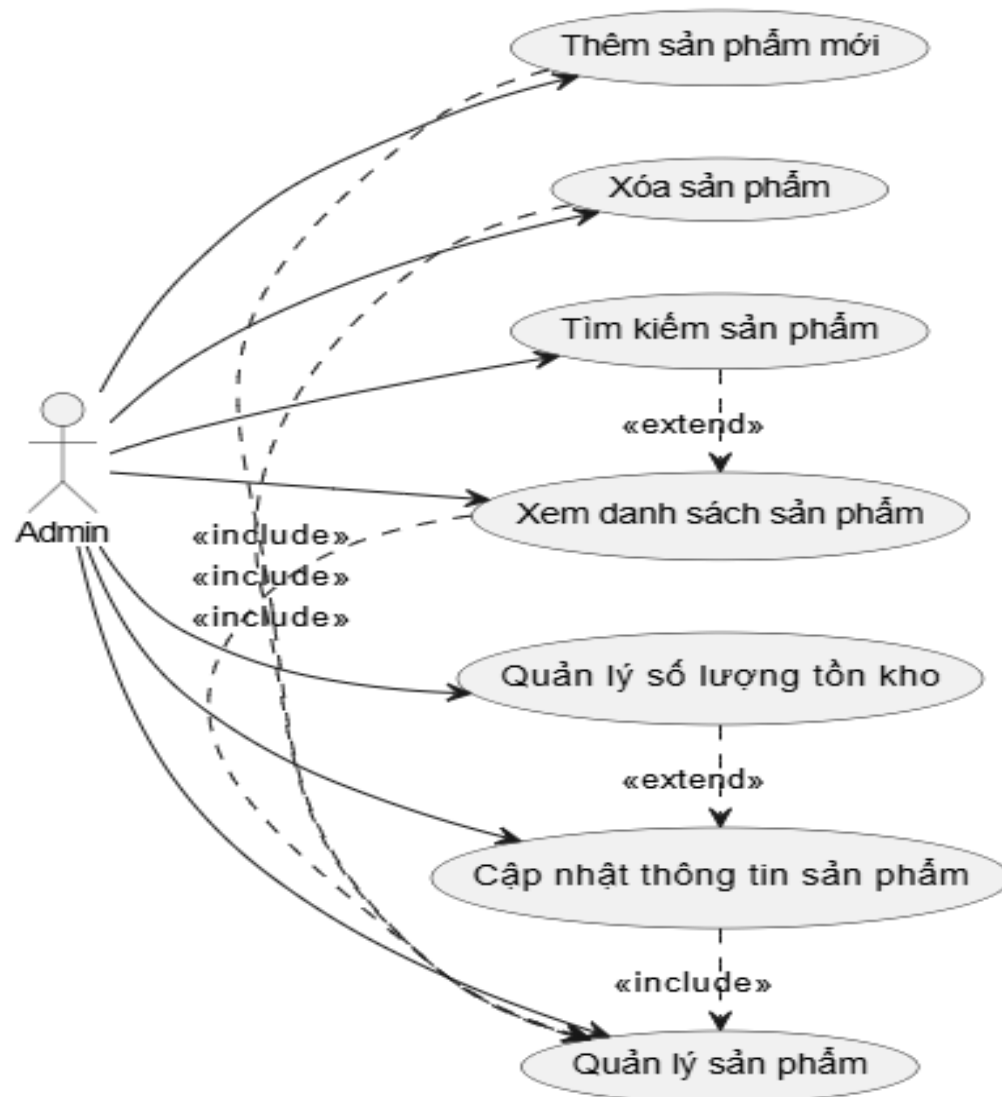
Hình 6. Use Case tổng quan của người dùng

3.4.5 Quản lý người dùng



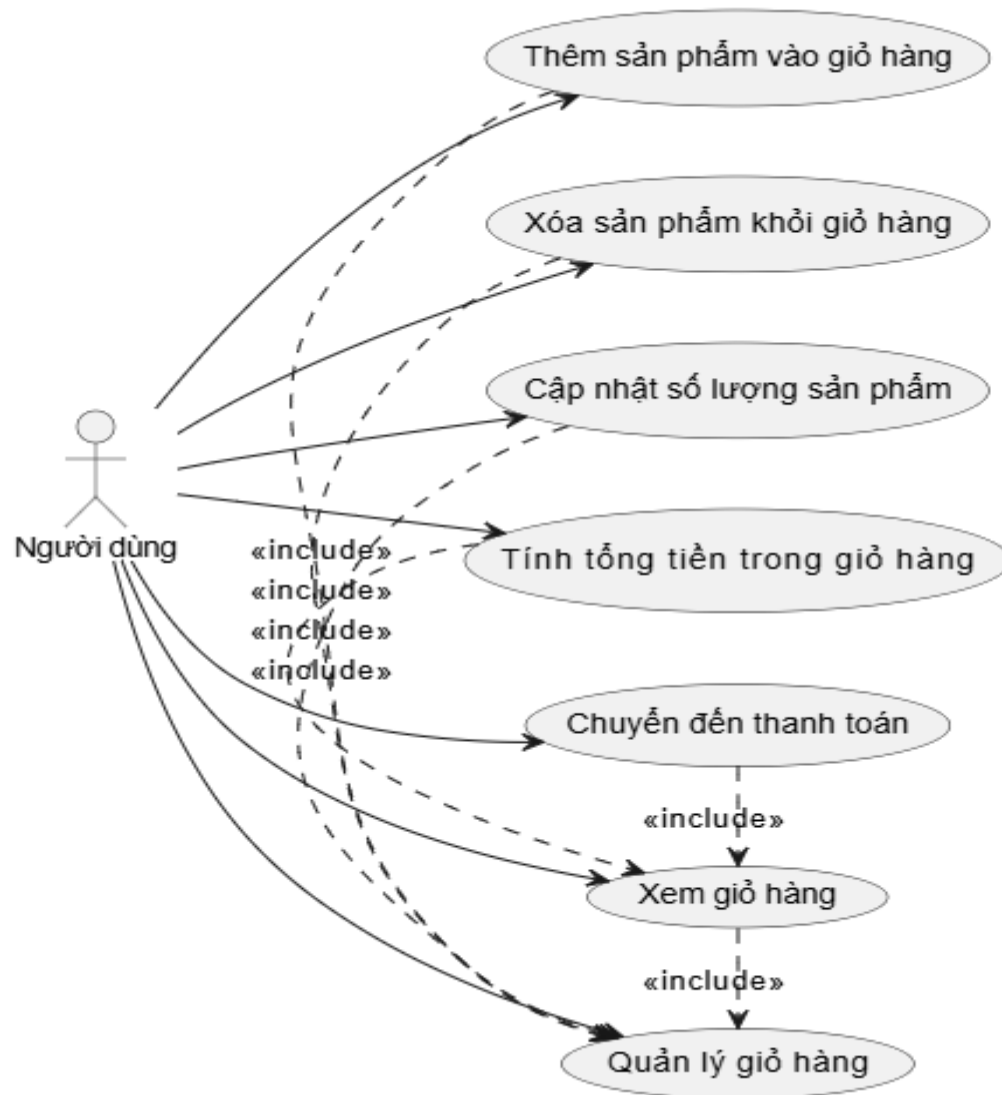
Hình 7. Use Case quản lý người dùng

3.4.6 Quản lý sản phẩm



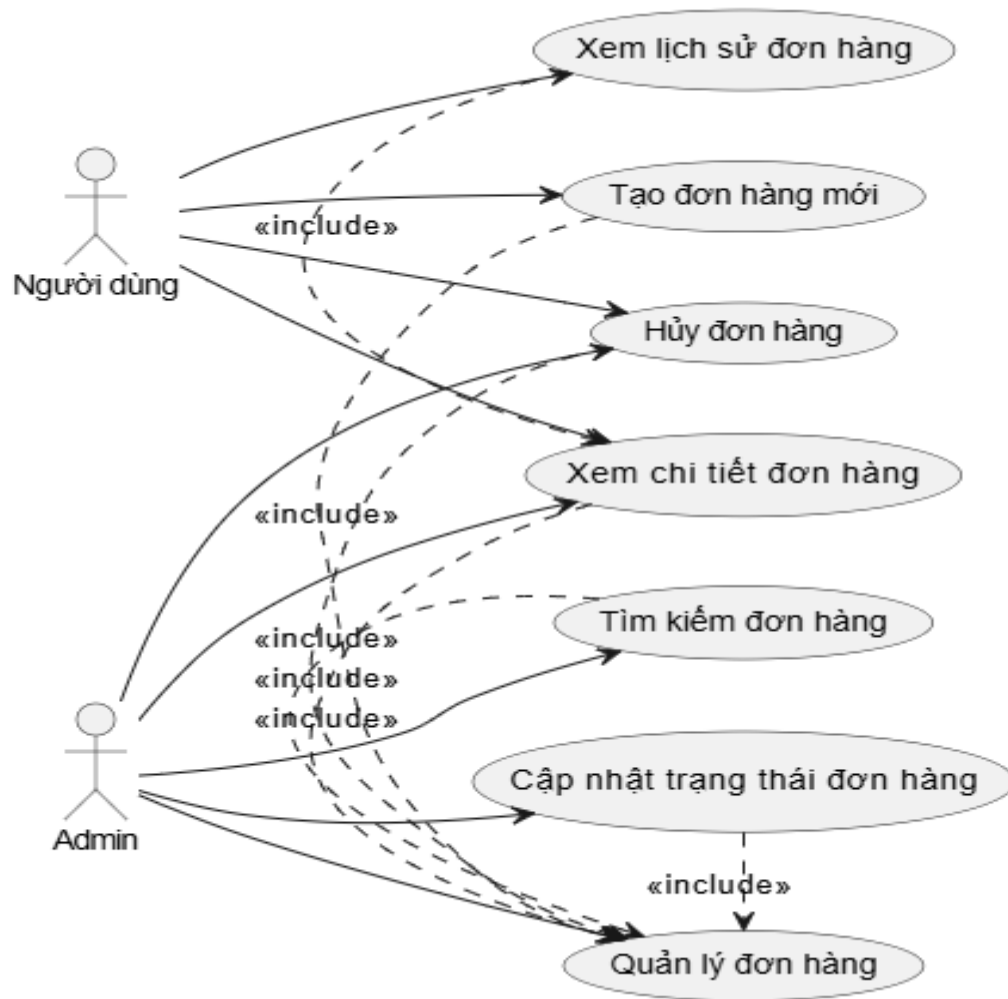
Hình 8. Use case quản lý sản phẩm

3.4.7 Quản lý giỏ hàng



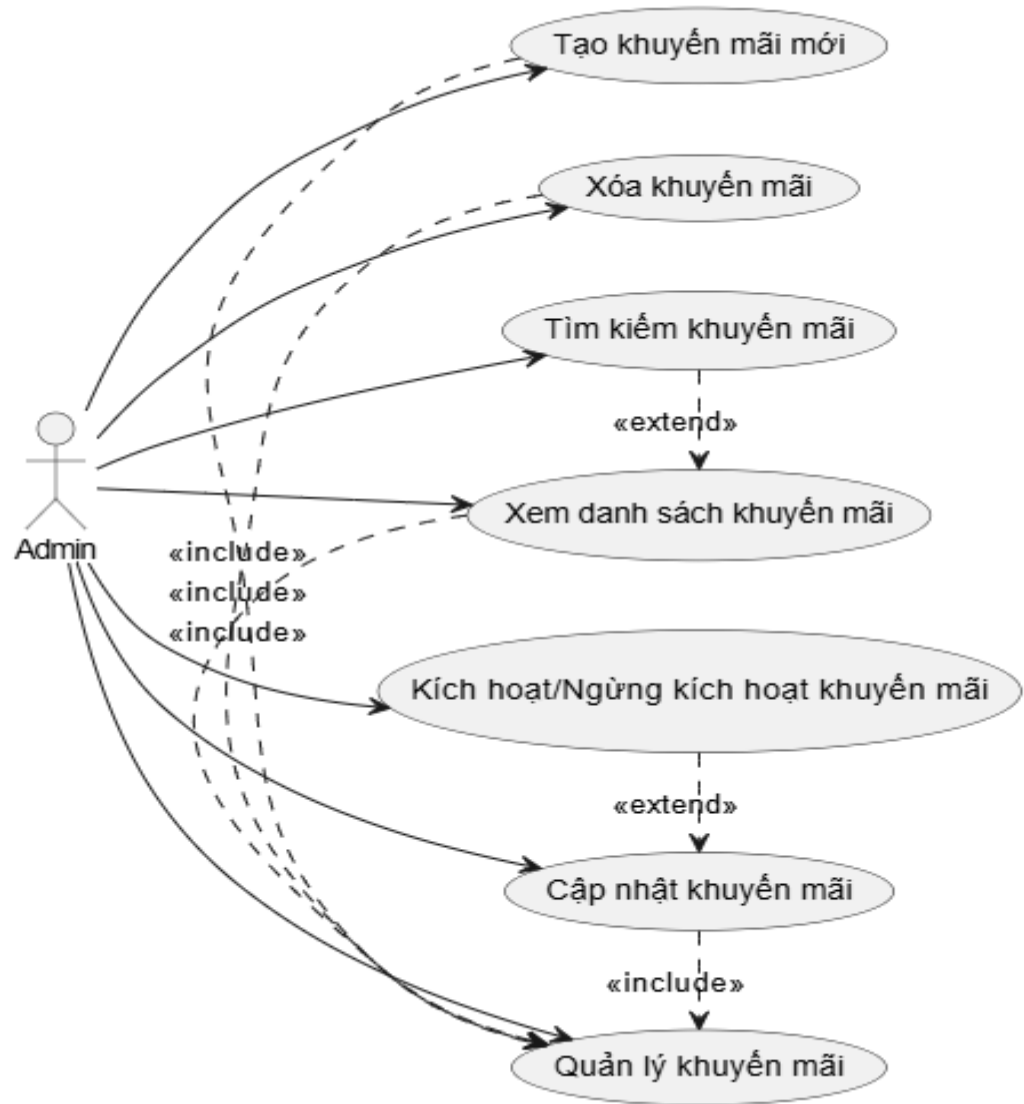
Hình 9. Use case quản lý giỏ hàng

3.4.8 Quản lý đơn hàng



Hình 10. Use case quản lý đơn hàng

3.4.9 Quản lý khuyến mãi



Hình 11. Use case quản lý khuyến mãi

3.5 Mô tả

3.5.1 Mô tả các Actor

Actor	Mô tả
Người dùng (User)	Bao gồm cả khách chưa đăng nhập (Guest) và người dùng đã đăng nhập Người dùng tương tác với hệ thống để tìm kiếm, duyệt, và đặt hàng.
Quản trị viên (Admin)	Người có quyền quản lý toàn bộ hệ thống, bao gồm sản phẩm, đơn hàng, người dùng, và các chương trình khuyến mãi.

3.5.1 Mô tả các use case

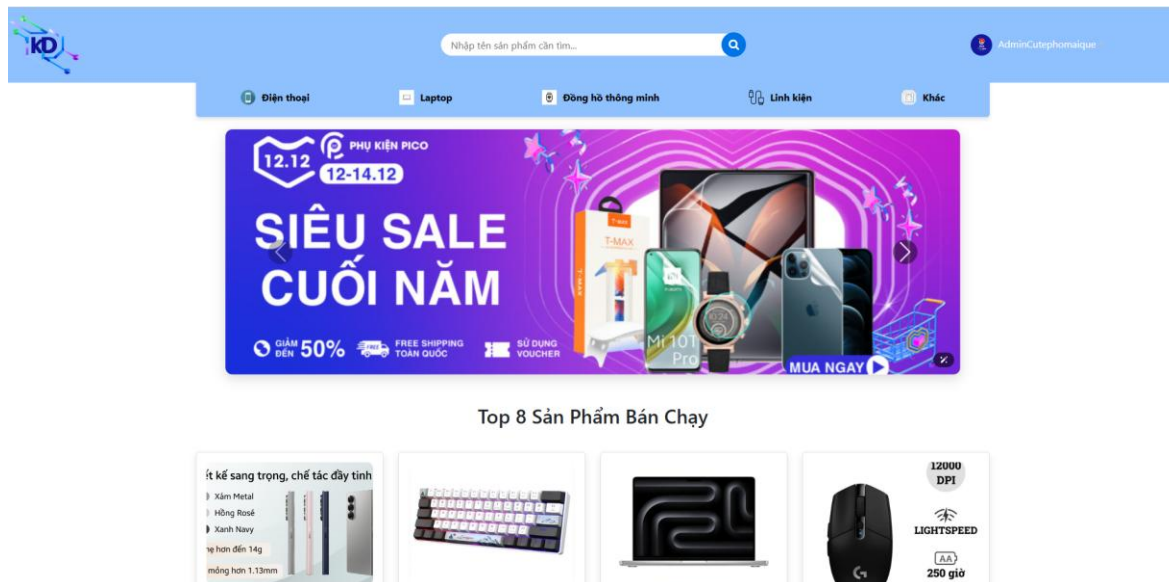
Use case	Mô tả
Quản lý người dùng	Admin có thể thực hiện các thao tác liên quan đến người dùng: Thêm người dùng mới (ví dụ: thêm tài khoản nhân viên). Cập nhật thông tin người dùng (chỉnh sửa email, số điện thoại, hoặc vai trò). Xóa người dùng không còn sử dụng. Xem danh sách người dùng hiện có. Tìm kiếm người dùng theo tên hoặc email. Khóa hoặc mở khóa tài khoản của người dùng trong trường hợp cần thiết.
Quản lý sản phẩm	Admin có thể quản lý danh mục sản phẩm bán hàng: Thêm sản phẩm mới với các thông tin như tên, giá, mô tả, hình ảnh. Cập nhật thông tin sản phẩm (giá cả, mô tả, hình ảnh). Xóa sản phẩm không còn kinh doanh. Xem danh sách sản phẩm hiện có. Tìm kiếm sản phẩm theo danh mục hoặc từ khóa. Quản lý số lượng tồn kho, bao gồm tăng giảm số lượng sản phẩm.
Quản lý giỏ hàng	Người dùng có thể quản lý giỏ hàng cá nhân: Thêm sản phẩm vào giỏ hàng từ danh sách sản phẩm hoặc chi tiết sản phẩm. Xóa sản phẩm không còn nhu cầu khỏi giỏ hàng. Cập nhật số lượng sản phẩm trong giỏ hàng. Xem toàn bộ giỏ hàng bao gồm tên sản phẩm, giá, và số lượng. Tính tổng tiền trong giỏ hàng (bao gồm các khoản giảm giá, nếu có). Chuyển đến thanh toán sau khi hoàn tất việc chọn sản phẩm.
Quản lý đơn hàng	User: Tạo đơn hàng mới sau khi chọn sản phẩm trong giỏ hàng. Xem chi tiết đơn hàng đã đặt, bao gồm sản phẩm, tổng tiền, trạng thái. Hủy đơn hàng trong một số điều kiện nhất định (trước khi vận chuyển). Xem lịch sử đơn hàng đã đặt, bao gồm cả các đơn đã hoàn thành

	<p>hoặc bị hủy.</p> <p>Admin: Cập nhật trạng thái đơn hàng (Đang xử lý, Đang giao, Đã hoàn thành). Hủy đơn hàng theo yêu cầu từ khách hàng hoặc do lỗi vận hành. Xem chi tiết đơn hàng và tìm kiếm đơn hàng theo trạng thái, thời gian, hoặc mã đơn hàng.</p>
Quản lý khuyến mãi	<p>Admin có thể quản lý các chương trình khuyến mãi như sau: Tạo chương trình khuyến mãi mới với các thông tin như tên, ngày bắt đầu, ngày kết thúc, giá trị giảm giá. Cập nhật chương trình khuyến mãi (chỉnh sửa thời gian, giá trị giảm giá). Xóa chương trình khuyến mãi đã hết hạn hoặc không còn áp dụng. Xem danh sách chương trình khuyến mãi hiện tại. Tìm kiếm chương trình khuyến mãi theo tên hoặc trạng thái. Kích hoạt hoặc ngừng chương trình khuyến mãi tùy theo kế hoạch kinh doanh.</p>

CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

4.1 Giao diện và chức năng của người dùng

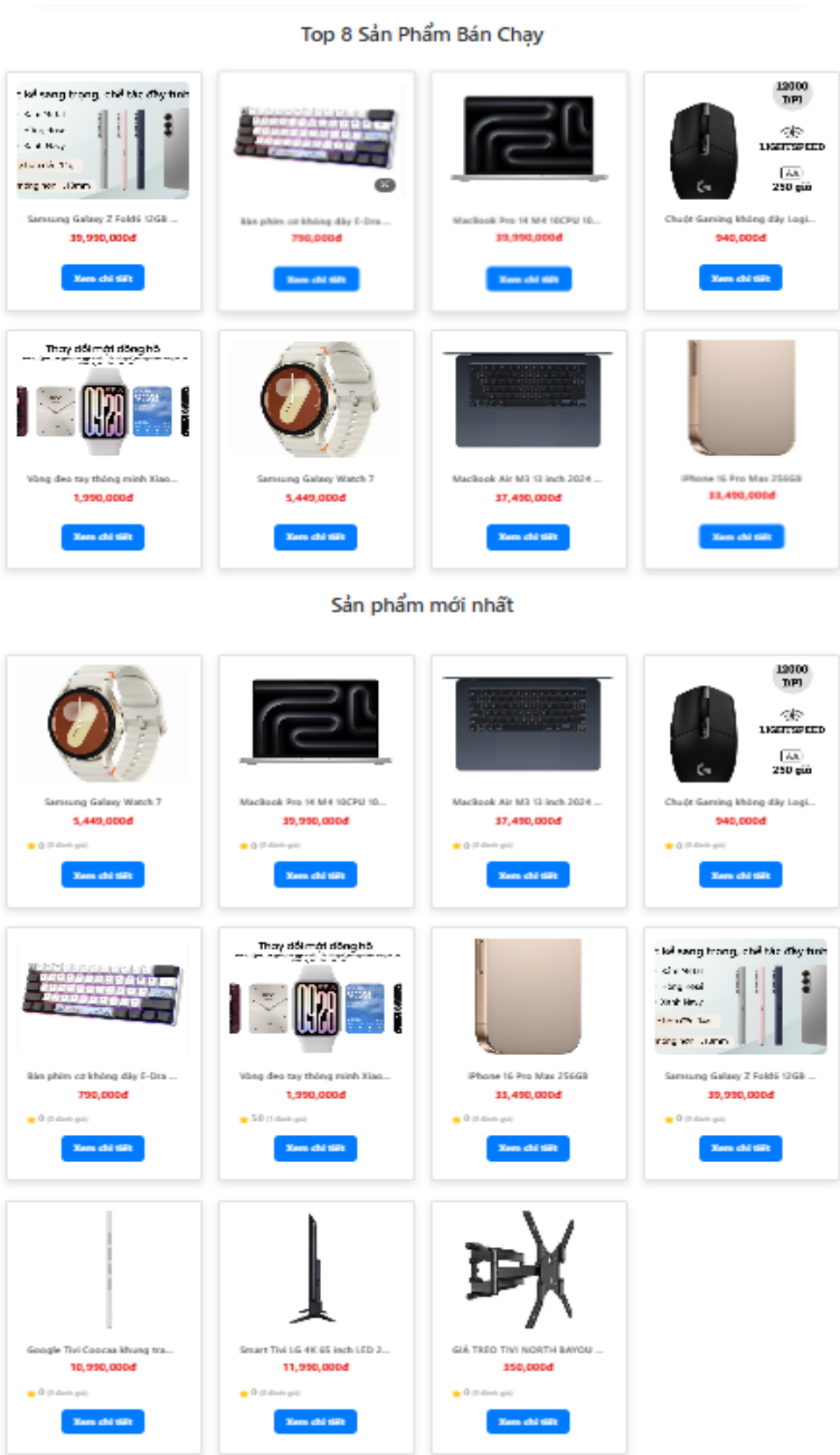
4.1.1 Giao diện trang chủ



Hình 12. Giao diện trang chủ

Trang chủ là giao diện đầu tiên mà người dùng sẽ thấy khi truy cập vào website, đóng vai trò là điểm tiếp xúc ban đầu giữa khách hàng và hệ thống. Được thiết kế với giao diện thân thiện, hiện đại và tối ưu hóa trải nghiệm người dùng, giúp khách hàng dễ dàng tiếp cận các sản phẩm nổi bật, khám phá những mặt hàng mới nhất, và nhanh chóng định hướng đến các danh mục của sản phẩm. Bên cạnh đó, chức năng tìm kiếm được tích hợp nhằm hỗ trợ người dùng tìm kiếm sản phẩm một cách nhanh chóng và chính xác. Mục tiêu chính của trang chủ không chỉ là tạo ấn tượng ban đầu mà còn hướng tới việc tối ưu hóa hành trình mua sắm, giữ chân người dùng và khuyến khích họ khám phá sâu hơn vào các tính năng của website.

4.1.2 Giao diện Sản phẩm



Hình 13. Giao diện Sản phẩm

Giao diện sản phẩm được thiết kế tối ưu để người dùng có thể dễ dàng tiếp cận và tương tác với các mặt hàng trên website. Đây là thành phần trọng tâm của hệ thống,

đóng vai trò quan trọng trong việc giới thiệu các sản phẩm đến khách hàng một cách trực quan và hiệu quả.

Mỗi sản phẩm được trình bày trong một khung hiển thị riêng, bao gồm hình ảnh, tên sản phẩm, giá bán và các nút chức năng như “Xem chi tiết” để cung cấp thêm thông tin chi tiết hoặc thực hiện các thao tác mua hàng. Bố cục được tổ chức với “Sản phẩm bán chạy” và “Sản phẩm mới nhất”, giúp người dùng nhanh chóng khám phá những mặt hàng nổi bật hoặc mới được cập nhật.

Ngoài ra, việc sử dụng biểu tượng đánh giá (rating) giúp người dùng nhận diện mức độ phổ biến của sản phẩm. Mỗi giao diện được thiết kế đáp ứng (responsive) để hiển thị tốt trên nhiều loại thiết bị khác nhau, từ máy tính đến điện thoại di động. Chức năng phân trang và sắp xếp sản phẩm cũng được tích hợp nhằm hỗ trợ người dùng dễ dàng tìm thấy sản phẩm phù hợp trong số lượng lớn các mặt hàng.

Mục tiêu chính của giao diện sản phẩm là đảm bảo sự tiện lợi, tính thẩm mỹ, và tăng cường trải nghiệm người dùng, từ đó thúc đẩy hành vi mua sắm và tạo sự hài lòng cho khách hàng khi sử dụng hệ thống.

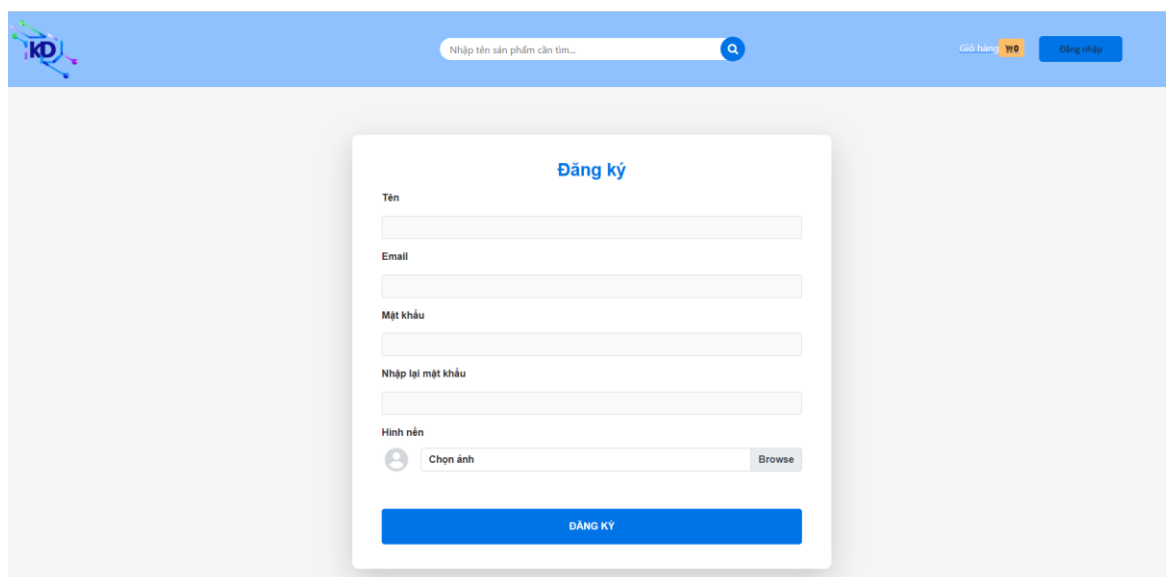
4.1.3 Giao diện đăng nhập

Hình 14. Giao diện đăng nhập

Giao diện đăng nhập được thiết kế tối giản và trực quan, giúp người dùng dễ dàng truy cập vào tài khoản của họ. Bao gồm các trường thông tin cơ bản như email và mật khẩu, cùng với nút “Đăng nhập” nổi bật. Phần liên kết “Bạn chưa có tài khoản?” hỗ trợ người dùng mới nhanh chóng chuyển sang giao diện đăng ký. Thanh điều

hướng phía trên tích hợp chức năng tìm kiếm, biểu tượng giỏ hàng và tùy chọn đăng nhập, đảm bảo trải nghiệm liền mạch và thuận tiện.

4.1.4 Giao diện đăng ký

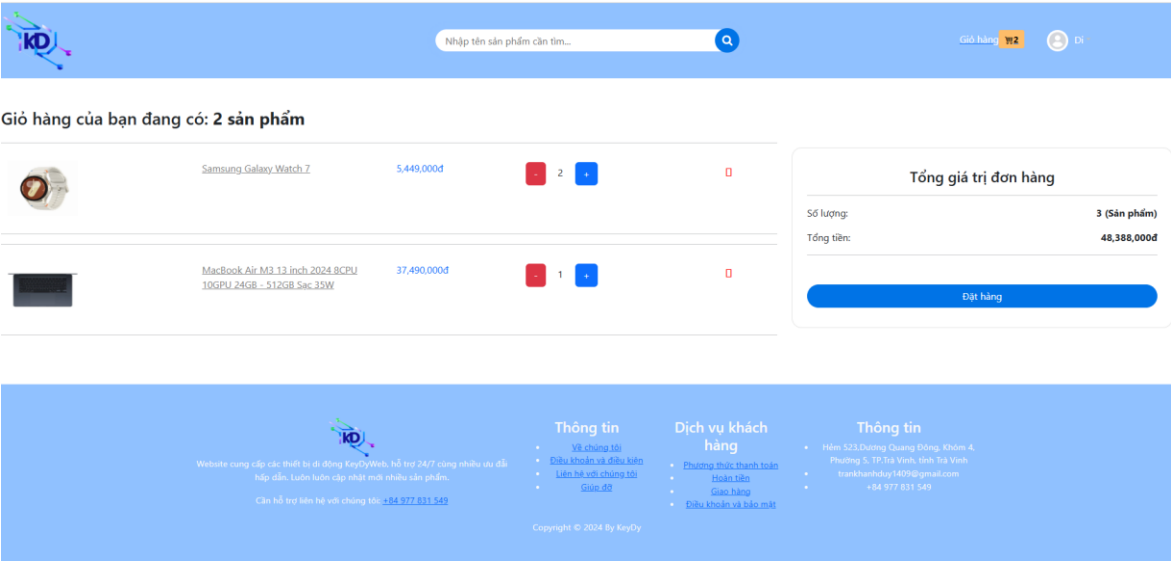


Hình 15. Giao diện đăng ký

Giao diện đăng ký được thiết kế đơn giản, thân thiện và trực quan để tối ưu hóa trải nghiệm người dùng. Tại đây, người dùng cần cung cấp các thông tin cơ bản như tên, email, mật khẩu và xác nhận mật khẩu nhằm đảm bảo tính chính xác trước khi tài khoản được tạo.

Phần nhập lại mật khẩu giúp kiểm tra và tránh sai sót khi thiết lập thông tin bảo mật. Ngoài ra, hệ thống còn cho phép người dùng tải lên hình đại diện cá nhân thông qua chức năng chọn file, với hình ảnh xem trước để đảm bảo chọn chính xác.

4.1.5 Giao diện giỏ hàng





Hình 16. Giao diện giỏ hàng

Giao diện giỏ hàng được thiết kế thân thiện và trực quan nhằm mang đến trải nghiệm mua sắm thuận tiện nhất cho người dùng. Tại đây, khách hàng có thể xem danh sách các sản phẩm đã chọn với thông tin chi tiết như tên sản phẩm, hình ảnh, giá bán, số lượng và tổng giá trị đơn hàng.

Người dùng có thể dễ dàng điều chỉnh số lượng sản phẩm bằng các nút tăng (+) hoặc giảm (-) trực tiếp trên giao diện, hoặc xóa sản phẩm khỏi giỏ hàng chỉ với một thao tác. Bảng tổng giá trị đơn hàng bên phải cung cấp thông tin tổng quan về số lượng sản phẩm và số tiền cần thanh toán, giúp khách hàng theo dõi chi phí một cách nhanh chóng và rõ ràng.


4.1.6 Giao diện nhập thông tin vận chuyển





Giỏ hàng

0

 Ds

Vận chuyển

Xác nhận hóa đơn

Thanh toán

Thông tin vận chuyển

Địa chỉ

Thành phố

Số điện thoại

Số nhà/Số hẻm

Quốc gia

NẾP TỤC

Hình 17. Giao diện nhập thông tin vận chuyển

Giao diện thông tin vận chuyển được thiết kế đơn giản và trực quan để giúp người dùng dễ dàng nhập thông tin cần thiết cho việc giao hàng. Tại đây, người dùng cần cung cấp các thông tin quan trọng như địa chỉ, thành phố, số điện thoại, số nhà/số hẻm và quốc gia.

4.1.7 Giao diện xác nhận hóa đơn

Giỏ hàng
Đã đăng nhập

Vận chuyển

Xác nhận hóa đơn

Thanh toán

Thông tin vận chuyển

Tên khách hàng: Di
Số điện thoại: 0977831549
Địa chỉ: 123, TP. Hồ Chí Minh, 123, Việt Nam

Các mặt hàng trong giỏ hàng của bạn:

	Samsung Galaxy Watch 7	2 x 5.449.000đ = 10,898,000đ
	Macbook Air M3 13 inch 2024 8CPU 10GPU 24GB - s12GB Sạc 35W	1 x 37.490.000đ = 37,490,000đ

Tổng giá trị đơn hàng

Thành tiền:	48,388,000đ
Phí vận chuyển:	0đ
Thuế:	483,880đ
Tổng tiền:	48,871,880đ

Tiến hành thanh toán

Website cung cấp các thiết bị di động KoryTech, hỗ trợ 24/7 cùng nhiều ưu đãi hấp dẫn. Luôn luôn cập nhật mức nhiều sản phẩm.

Gần hỗ trợ liên hệ với chúng tôi: [+84 977 831 549](tel:+84977831549)

Thông tin

- Về chúng tôi
- Điều khoản và điều kiện
- Liên hệ với chúng tôi
- Giới thiệu

Dịch vụ khách hàng

- Phương thức thanh toán
- Chính sách
- Giáo dục
- Điều khoản và bảo mật

Thông tin

- Hẻm 523 Đường Quang Đông, Khóm 4, Phường 5, TP Trà Vinh, Vĩnh Trà Vinh
- tranhanhquy1409@gmail.com
- +84 977 831 549

Copyright © 2004 by Korytech

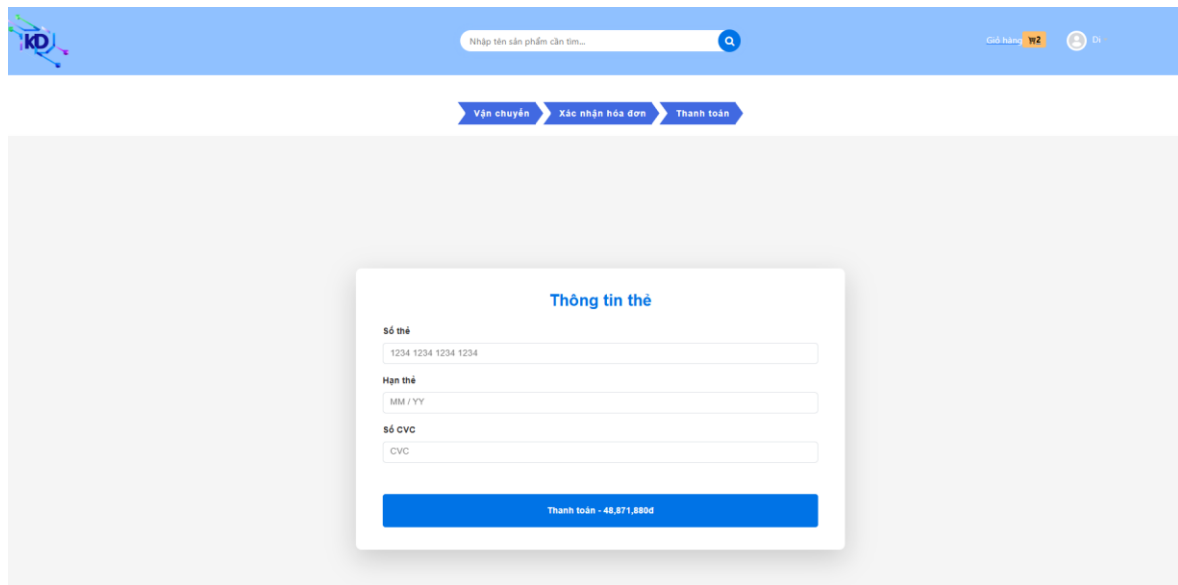
Hình 18. Giao diện xác nhận hóa đơn

Giao diện xác nhận hóa đơn được thiết kế nhằm tổng hợp thông tin chi tiết về đơn hàng trước khi người dùng tiến hành thanh toán. Tại đây, thông tin vận chuyển

của khách hàng như tên, số điện thoại, địa chỉ được hiển thị rõ ràng để người dùng kiểm tra và xác nhận.

Bên dưới là danh sách các sản phẩm trong giỏ hàng, bao gồm hình ảnh, tên sản phẩm, số lượng, giá bán và tổng giá trị từng sản phẩm, giúp người dùng dễ dàng kiểm tra chi tiết các mặt hàng đã chọn. Phần bên phải hiển thị bảng tổng kết đơn hàng với các thông tin quan trọng

4.1.8 Giao diện thanh toán

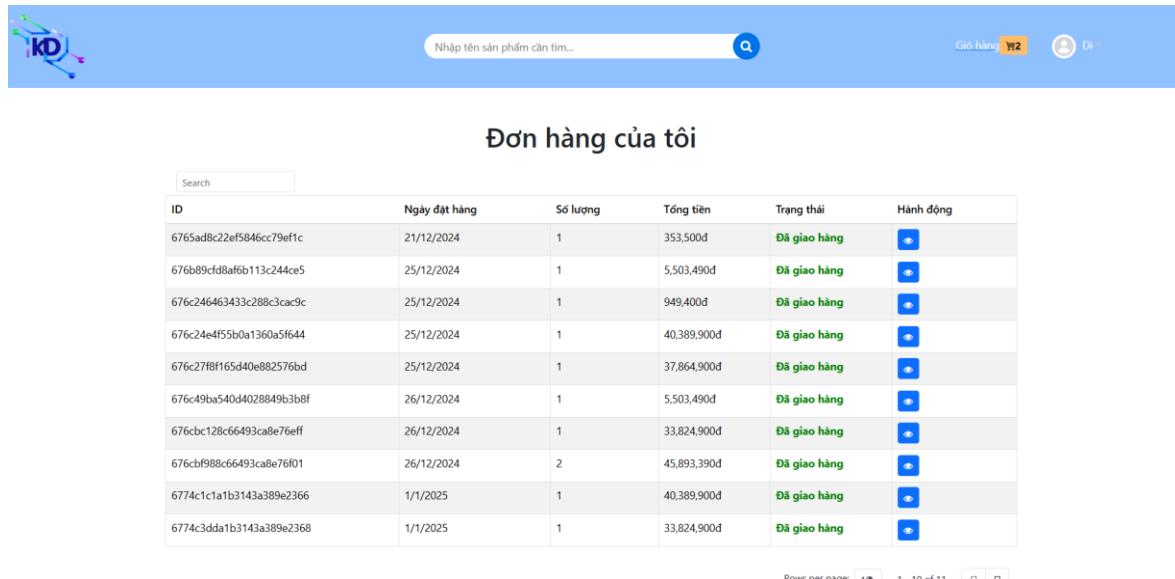


The screenshot displays a web application's payment page. At the top, there is a blue header with a logo on the left, a search bar in the center, and a shopping cart icon on the right. Below the header, a navigation bar shows three steps: 'Vận chuyển', 'Xác nhận hóa đơn', and 'Thanh toán', with 'Thanh toán' being the active step. The main content area features a white card titled 'Thông tin thẻ' (Card Information). This card contains three input fields: 'Số thẻ' (Card Number) with the value '1234 1234 1234 1234', 'Hạn thẻ' (Card Expiry) with the value 'MM / YY', and 'Số CVC' (CVC Number) with the value 'CVC'. At the bottom of the card, a blue button displays the total amount: 'Thanh toán - 48,871,880đ'.

Hình 19. Giao diện thanh toán

Giao diện thanh toán là bước cuối cùng trong quy trình đặt hàng, được thiết kế với mục tiêu đơn giản và bảo mật, đảm bảo khách hàng có thể hoàn tất giao dịch một cách nhanh chóng và an toàn. Người dùng cần cung cấp các thông tin cần thiết như số thẻ, hạn thẻ và số CVC. Bên dưới là nút “Thanh toán” được thiết kế nổi bật, hiển thị tổng số tiền cần thanh toán để khách hàng dễ dàng nhận biết. Hệ thống cũng đảm bảo các trường nhập liệu có tính năng kiểm tra để đảm bảo tính chính xác và tránh sai sót trong quá trình giao dịch.

4.1.9 Giao diện đơn hàng



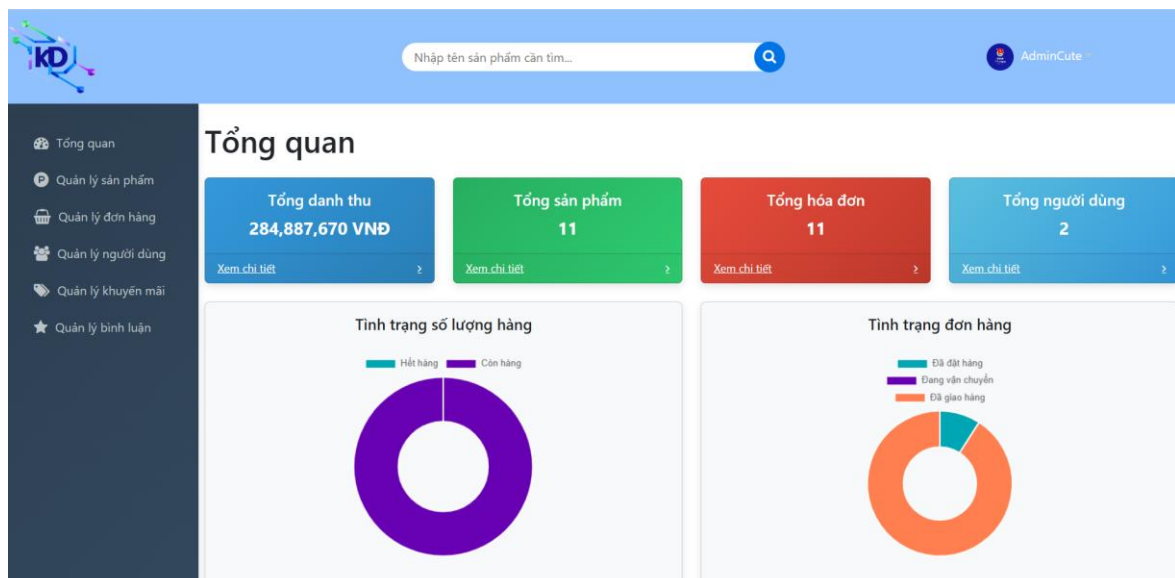
ID	Ngày đặt hàng	Số lượng	Tổng tiền	Trạng thái	Hành động
6765ad8c22ef5846cc79ef1c	21/12/2024	1	353,500đ	Đã giao hàng	+
676b89cfd8af6b113c244ce5	25/12/2024	1	5,503,490đ	Đã giao hàng	+
676c24646343c288c3cac9c	25/12/2024	1	949,400đ	Đã giao hàng	+
676c24e4f55b0a1360a5f644	25/12/2024	1	40,389,900đ	Đã giao hàng	+
676c27f8f165d40e882576bd	25/12/2024	1	37,864,900đ	Đã giao hàng	+
676c49ba540d4028849b3b8f	26/12/2024	1	5,503,490đ	Đã giao hàng	+
676cbc128c66493ca8e76eff	26/12/2024	1	33,824,900đ	Đã giao hàng	+
676cbf988c66493ca8e76f01	26/12/2024	2	45,893,390đ	Đã giao hàng	+
6774c1c1a1b3143a389e2366	1/1/2025	1	40,389,900đ	Đã giao hàng	+
6774c3dda1b3143a389e2368	1/1/2025	1	33,824,900đ	Đã giao hàng	+

Hình 20. Giao diện đơn hàng

Giao diện danh sách đơn hàng được thiết kế nhằm cung cấp cho người dùng cái nhìn tổng quan và chi tiết về tất cả các đơn hàng đã đặt. Bảng dữ liệu được phân trang với tính năng tùy chọn số hàng trên mỗi trang, giúp hiển thị mượt mà ngay cả khi có nhiều đơn hàng. Thanh tìm kiếm tích hợp cho phép người dùng dễ dàng tìm kiếm đơn hàng theo các tiêu chí cụ thể như ID hoặc thông tin khác.

4.2 Giao diện và chức năng của quản trị viên

4.2.1 Giao diện quản lý chung

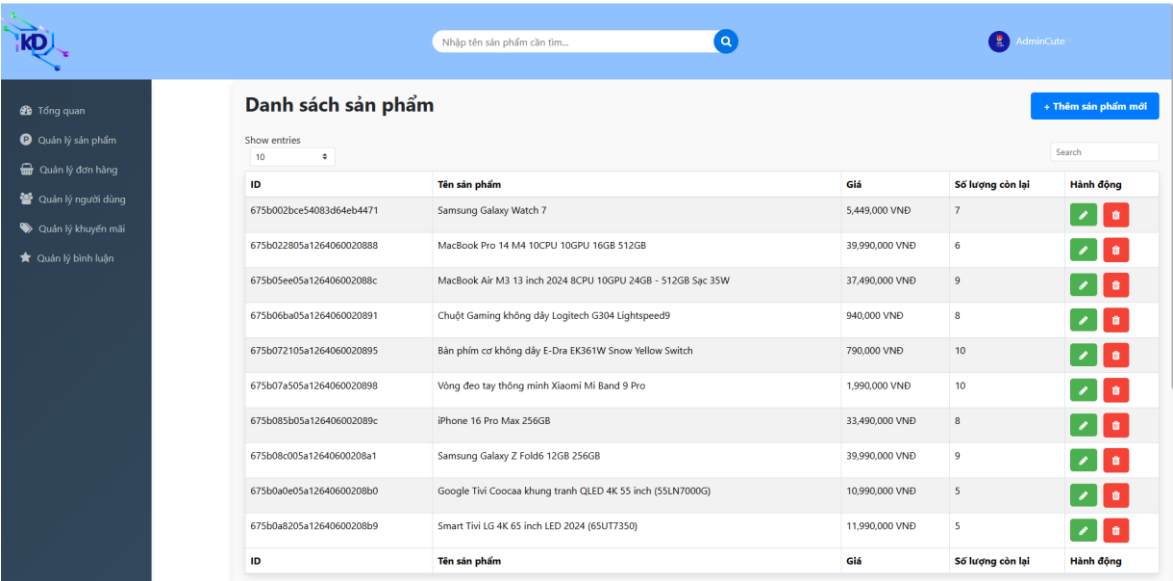


Hình 21. Quản lý tổng quan

Trang tổng quan dành cho quản trị viên được thiết kế hiện đại, trực quan, nhằm cung cấp cái nhìn tổng thể về tình hình hoạt động của hệ thống. Trang tổng quan

không chỉ cung cấp thông tin tổng hợp mà còn giúp quản trị viên dễ dàng theo dõi và quản lý hệ thống một cách hiệu quả và chính xác.

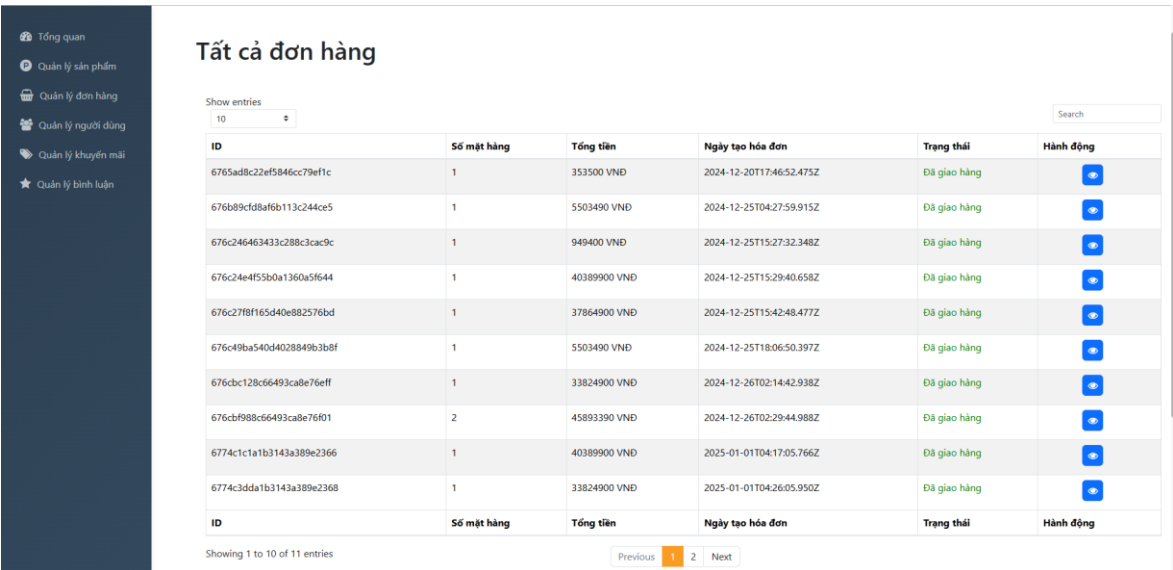
4.1.2 Giao diện quản lý sản phẩm



Hình 22. Giao diện quản lý sản phẩm

Giao diện quản lý sản phẩm giúp quản trị viên dễ dàng theo dõi và cập nhật thông tin sản phẩm. Các chức năng chính bao gồm hiển thị danh sách sản phẩm, tìm kiếm, chỉnh sửa, xóa và thêm sản phẩm mới. Giao diện được thiết kế trực quan, hỗ trợ quản lý nhanh chóng và hiệu quả.

4.1.3 Giao diện quản lý đơn hàng

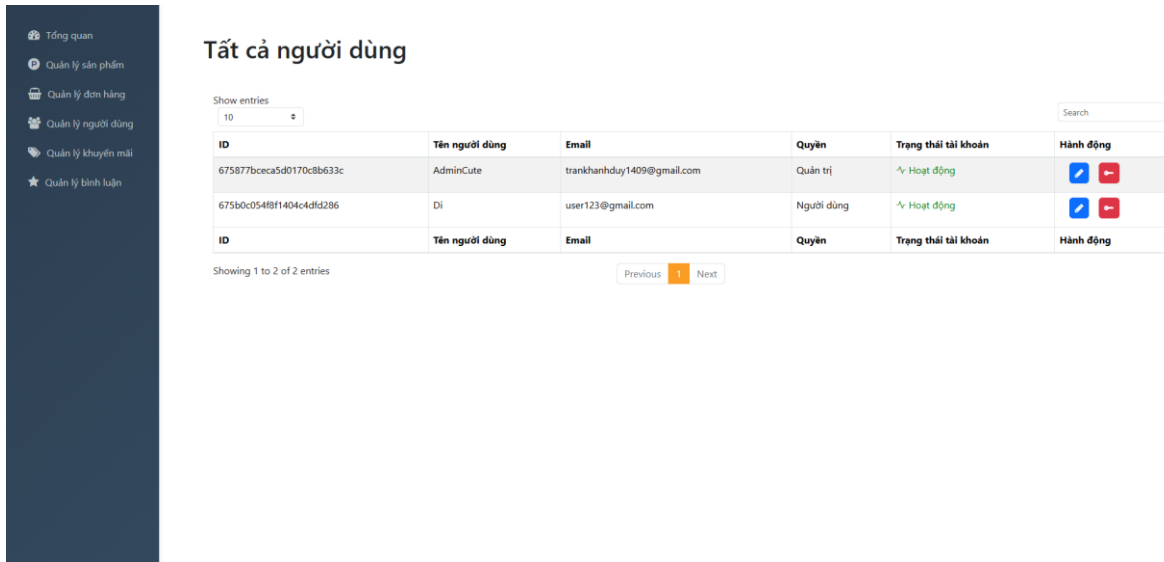


Hình 23. Quản lý đơn hàng





Giao diện quản lý đơn hàng cung cấp các chức năng theo dõi và xử lý đơn hàng hiệu quả. Danh sách các đơn hàng được hiển thị với thông tin chi tiết như số mặt hàng,

tổng tiền, ngày tạo hóa đơn và trạng thái. Người quản trị có thể xem chi tiết từng đơn hàng để xử lý hoặc kiểm tra thông tin giao dịch. Giao diện đơn giản, dễ sử dụng, giúp tối ưu hóa việc quản lý và theo dõi đơn hàng.

4.1.4 Giao diện quản lý người dùng



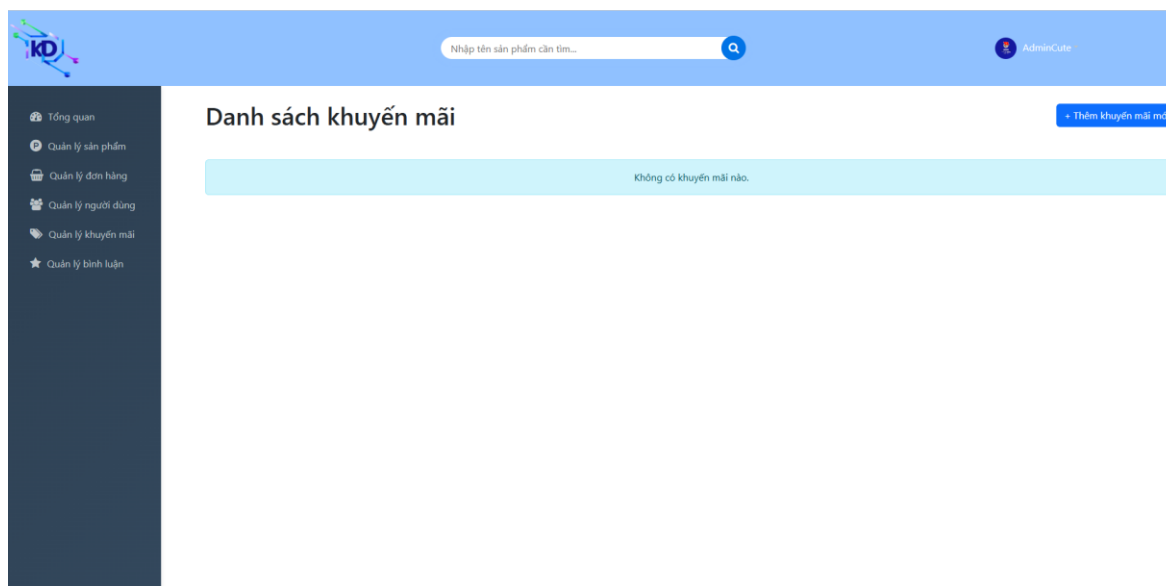
The screenshot shows a web application interface for managing users. On the left is a dark sidebar with navigation links: Tổng quan, Quản lý sản phẩm, Quản lý đơn hàng, Quản lý người dùng (active), Quản lý khuyến mãi, and Quản lý bình luận. The main content area is titled 'Tất cả người dùng'. It features a 'Show entries' dropdown set to 10 and a search bar. Below is a table with 6 columns: ID, Tên người dùng, Email, Quyền, Trạng thái tài khoản, and Hành động. Two users are listed: AdminCute (role: Quản trị, status: Hoạt động) and Di (role: Người dùng, status: Hoạt động). Each user has two action buttons: a blue plus icon and a red minus icon. At the bottom, it says 'Showing 1 to 2 of 2 entries' with 'Previous', '1', and 'Next' navigation buttons.

ID	Tên người dùng	Email	Quyền	Trạng thái tài khoản	Hành động
675877bceca5d0170c8b633c	AdminCute	trankhanhduy1409@gmail.com	Quản trị	✓ Hoạt động	 
675b0c0548f1404c4dfd286	Di	user123@gmail.com	Người dùng	✓ Hoạt động	 

Hình 24. Giao diện quản lý người dùng

Giao diện quản lý người dùng hiển thị danh sách các tài khoản người dùng, bao gồm thông tin như ID, tên người dùng, email, quyền (quản trị hoặc người dùng thông thường), và trạng thái tài khoản (hoạt động hoặc bị khóa). Người quản trị có thể chỉnh sửa thông tin hoặc thay đổi quyền truy cập của người dùng thông qua các nút hành động tương ứng.

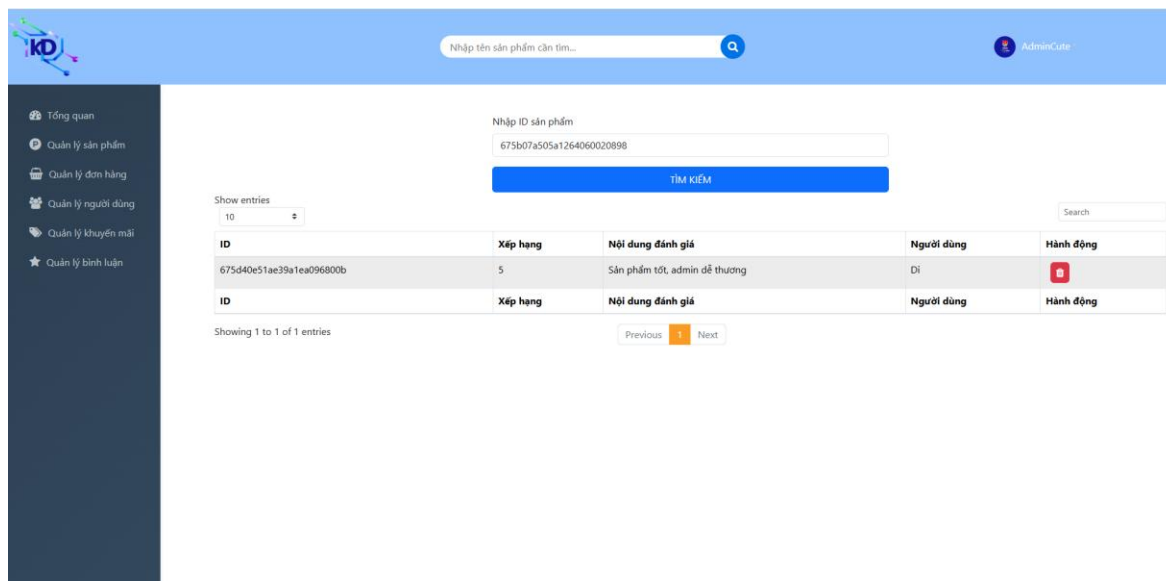
4.1.5 Giao diện quản lý khuyến mãi



Hình 25. Giao diện quản lý khuyến mãi

Giao diện quản lý khuyến mãi cung cấp tính năng giúp quản trị viên dễ dàng thêm, chỉnh sửa hoặc xóa các chương trình khuyến mãi. Trong giao diện, nếu chưa có chương trình khuyến mãi nào, một thông báo hiển thị để quản trị viên nhận biết và có thể nhấp vào nút “Thêm khuyến mãi mới” để bắt đầu tạo chương trình khuyến mãi. Giao diện được thiết kế đơn giản và trực quan, giúp quản lý các chiến dịch khuyến mãi một cách nhanh chóng và hiệu quả.

4.1.6 Giao diện quản lý bình luận



Hình 26. Giao diện quản lý bình luận

Giao diện quản lý bình luận cung cấp công cụ để quản trị viên dễ dàng xem, tìm kiếm và quản lý các đánh giá từ người dùng. Quản trị viên có thể nhập ID sản phẩm để lọc các đánh giá liên quan, kiểm tra thông tin xếp hạng và nội dung đánh giá chi tiết. Ngoài ra, giao diện cung cấp chức năng xóa bình luận không phù hợp thông qua nút “Xóa”. Thiết kế đơn giản, hỗ trợ hiệu quả trong việc duy trì chất lượng nội dung phản hồi của khách hàng.

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết quả đạt được

Qua quá trình thực hiện, đề tài đã đạt được những kết quả nổi bật như sau:

Nắm vững khái niệm, nguyên tắc, và cơ chế hoạt động của Restful API, cùng với công nghệ ReactJS và NodeJS.

Ứng dụng thành công các nguyên tắc và công nghệ trên để xây dựng website bán thiết bị di động với các chức năng tìm kiếm, quản lý đơn hàng, tài khoản, chương trình khuyến mãi, và đánh giá sản phẩm.

Triển khai website thành công trên môi trường mạng Internet, đảm bảo hoạt động ổn định và đáp ứng các yêu cầu kỹ thuật đã đề ra.

Về kiến thức và kỹ năng, đề tài giúp nâng cao năng lực lập trình front-end với ReactJS, lập trình back-end với NodeJS, quản lý dự án từ nghiên cứu đến triển khai, cũng như kỹ năng giải quyết vấn đề thực tế.

Website được xây dựng cung cấp nền tảng bán hàng trực tuyến hiệu quả, thân thiện với người dùng, đồng thời hỗ trợ quản lý linh hoạt các hoạt động như đơn hàng, tài khoản, và khuyến mãi.

5.2 Hướng phát triển

Nâng cấp thêm các tính năng hỗ trợ khách hàng như chatbot tự động, theo dõi đơn hàng theo thời gian thực.

Tích hợp thêm công nghệ trí tuệ nhân tạo (AI) để gợi ý sản phẩm phù hợp với từng khách hàng.

Tối ưu hóa hiệu năng và bảo mật cho hệ thống nhằm nâng cao trải nghiệm người dùng.

Mở rộng khả năng kết nối với các nền tảng thương mại điện tử khác.

DANH MỤC TÀI LIỆU THAM KHẢO

Sách/ Tài liệu

- [1] Đoàn Phước Miên, Phạm Thị Trúc Mai, Thiết kế và lập trình Web, Trường Đại học Trà Vinh, 2014.
- [2] H. T. T. Vi, Tài liệu Giảng dạy môn Cơ sở dữ liệu, Trường Đại học Trà Vinh, 2313.
- [3] Phạm Minh Dương, Tài liệu Giảng dạy môn Phân tích và thiết kế hệ thống thông tin, Trường Đại học Trà Vinh, 2014.

Website

- [4] Hướng nội, “ReactJS là gì? Thư viện Javascript tối ưu cho giao diện Web,” 200Lab, 12 9 2024. [Online]. Available: <https://200lab.io/blog/reactjs-la-gi>. [Accessed 2024 12 6].
- [5] Hoang Trong Hieu, “ReactJS - Ưu điểm và nhược điểm,” 2017. [Online]. Available: <https://viblo.asia/p/reactjs-uu-diem-va-nhuoc-diem-V3m5WzexlO7>. [Accessed 6 12 2024].
- [6] Software Engineer Training, “ReactJS là gì? Những điều có thể bạn chưa biết về ReactJS,” viblo, 25 6 2021. [Online]. Available: <https://viblo.asia/p/reactjs-la-gi-nhung-dieu-co-the-ban-chua-biet-ve-reactjs-eW65G4RRKDO>. [Accessed 6 12 2024].
- [7] HỒNG NGUYỄN, “Reactjs Là Gì? 6 Đặc Trưng Của Reactjs Hiện Nay,” jobsgo, 01 10 2024. [Online]. Available: <https://jobsgo.vn/blog/reactjs-la-gi/>. [Accessed 6 12 2024].
- [8] Trang Vũ, “ReactJS là gì? Tất tần tật những điều căn bản về ReactJS,” stringee, 27 2 2024. [Online]. Available: <https://stringee.com/vi/blog/post/reactJS-la-gi>. [Accessed 6 12 2024].
- [9] Tuong Uyen, “NodeJS là gì: Tổng quan kiến thức NodeJS và Top 5 NodeJS framework,” ITviec Blog, 29 9 2024. [Online]. Available: <https://itviec.com/blog/nodejs-la-gi/>. [Accessed 6 12 2024].
- [10] Tinh To, “15+ Tài liệu học NodeJS “chất” dành cho Developer,” ITviec Blog, 29 9 2024. [Online]. Available: <https://itviec.com/blog/hoc-nodejs/>. [Accessed 6 12 2024].
- [11] Cao Lê Viết Tiến, “NodeJS là gì? Tổng quan kiến thức về Node.JS,” VietNix, 13 09 2024. [Online]. Available: <https://vietnix.vn/nodejs-la-gi/>. [Accessed 6 12 2024].
- [12] TopDev, “Expressjs là gì? Framework mạnh mẽ Nodejs Express,” TopDev, [Online]. Available: <https://topdev.vn/blog/express-js-la-gi/>. [Accessed 6 12 2024].
- [13] Admin, “ExpressJS là gì? Hướng dẫn học ExpressJS chi tiết từ A-Z,” CodeGym, 8 1 2024. [Online]. Available: <https://codegym.vn/blog/expressjs-la-gi-huong-dan-cach-hoc-expressjs-chi-tiet/>. [Accessed 6 12 2024].
- [14] TopDev, “RESTful API là gì? Cách thiết kế RESTful API,” TopDev, [Online]. Available: <https://topdev.vn/blog/restful-api-la-gi/>. [Accessed 6 12 2024].
- [15] Lê Đức Mạnh, “RESTful API là gì ?,” ViBlo, 9 5 2020. [Online]. Available: <https://viblo.asia/p/restful-api-la-gi-1Je5EDJ4lnL>. [Accessed 6 12 2024].
- [16] Software Engineer Training, “MONGODB LÀ GÌ? CÁC TÍNH NĂNG NỔI BẬT CỦA MONGODB,” Viblo, 16 5 2023. [Online]. Available: <https://viblo.asia/p/mongodb-la-gi-cac-tinh-nang-noi-bat-cua-mongodb-PAoJex2N4lj>.

[Accessed 6 12 2024].

PHỤ LỤC