TRƯỜNG ĐẠI HỌC TRÀ VINH

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

ISO 9001:2015

TÔ VĂN TỚI

PHÁT TRIỂN WEBSITE BÁN THỜI TRANG TÍCH HỢP AI THỬ ĐỒ ẢO

KHÓA LUẬN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

VĨNH LONG, NĂM 2025

TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

PHÁT TRIỂN WEBSITE BÁN THỜI TRANG TÍCH HỢP AI THỬ ĐỒ ẢO

KHÓA LUẬN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

Sinh viên thực hiện: **TÔ VĂN TỚI**

Mã số sinh viên: **110121252**

Lớp: **DA21TTA**

GVHD: **PHẠM MINH ĐƯƠNG**

VĨNH LONG, NĂM 2025

LỜI CAM ĐOAN

Tôi xin cam đoan rằng đề tài khóa luận tốt nghiệp với tên "Phát triển website bán thời trang tích hợp AI thử đồ ảo" là công trình nghiên cứu do chính tôi thực hiện.

Các số liệu, kết quả, hình ảnh, bảng biểu được trình bày trong khóa luận là hoàn toàn trung thực, được thu thập từ những nguồn đáng tin cậy và đã được trích dẫn rõ ràng theo quy định. Tôi khẳng định rằng khóa luận này không sao chép từ bất kỳ công trình nghiên cứu nào khác.

Nếu có bất kỳ sự gian lận hay vi phạm nào liên quan đến nội dung cam đoan trên, tôi xin hoàn toàn chịu trách nhiệm trước pháp luật cũng như quy định của nhà trường.

Xin trân trọng cảm ơn!

Vĩnh long, ngày….tháng….năm 2025

Người cam đoan

Tô Văn Tới

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành và sâu sắc đến Thạc sĩ Phạm Minh Đương là giảng viên hướng dẫn, người đã luôn tận tình chỉ bảo, định hướng và hỗ trợ em trong suốt quá trình thực hiện đề tài "Phát triển website bán thời trang tích hợp AI thử đồ ảo". Nhờ sự hướng dẫn tận tâm của thầy, em đã có thể vượt qua những khó khăn, hoàn thiện khóa luận với tinh thần cao và trách nhiệm.

Em cũng xin trân trọng cảm ơn quý thầy cô trong Khoa Công nghệ Thông tin Trường Kỹ thuật và Công nghệ Trường Đại học Trà Vinh, những người đã truyền đạt cho em không chỉ kiến thức chuyên môn quý báu mà còn là những kinh nghiệm thực tiễn và thái độ làm việc chuyên nghiệp trong suốt 4 năm học tập tại trường. Chính sự dìu dắt và tận tâm của quý thầy cô là nền tảng vững chắc giúp em hình thành tư duy, kỹ năng và động lực để phát triển bản thân trong hành trình học tập cũng như sự nghiệp tương lai.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn đồng hành, động viên và tiếp thêm sức mạnh cho em vượt qua những thử thách trong suốt quá trình học tập và thực hiện khóa luận tốt nghiệp này.

Em xin kính chúc quý thầy cô luôn mạnh khỏe, công tác tốt và ngày càng thành công hơn trong sự nghiệp.

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
MỤC LỤC..... iii
KÍ HIỆU : CÁC CỤM TỪ VIẾT TẮT xii
DANH MỤC BẢNG BIỂU xiii
DANH MỤC HÌNH ẢNH xiv
CHƯƠNG 1: ĐẶT VẤN ĐỀ 2
1.1. Lý do chọn đề tài.....2
1.2. Mục tiêu nghiên cứu 2
1.3. Phạm vi và đối tượng nghiên cứu 3
1.4. Phương pháp thực hiện 3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 5
2.1. Các mô hình AI thử đồ ảo.....5
2.2. .NET Core, Entity Framework Core và MediatR 6
2.2.1. .NET Core 6
2.2.2. Entity Framework Core.....7
2.2.3. MediatR.....8
2.2.4. Kết hợp .Net Core, Entity Framework Core và MediatR 9
2.2.5. Lợi ích khi kết hợp .Net Core, Entity Framework Core và MediatR 10
2.3. RESTful API 10
2.3.1. Khái niệm RESTful.....10
2.3.2. Phương thức HTTP chính 11
2.3.3. Tài nguyên, URI và định dạng dữ liệu.....11

2.3.4. Mã trạng thái HTTP 11
2.3.5. Ưu điểm và nhược điểm của RESTful API 12
2.4. Kiến Trúc Clean Architecture 12
2.4.1. Mục Tiêu 13
2.4.2. Các thành phần chính trong Clean Architecture 14
2.4.3. Nguyên tắc Dependency Rule (quy tắc phụ thuộc) 15
2.4.4. Ưu và nhược điểm của Clean Architecture 15
2.4.5. Áp dụng Clean Architecture trong .NET Core 16
2.4.6. Tóm tắt quy trình làm việc trong Clean Architecture 16
2.5. MimeKit 17
2.5.1. Khái niệm 17
2.5.2. Chức năng của Mimekit 17
2.6. ChatBot trí tuệ nhân tạo 18
2.6.1. Giới thiệu 18
2.6.2. Nguyên tắc hoạt động 18
2.6.3. Ứng dụng Chatbot trong thương mại điện tử thời trang 19
2.6.4. Lợi ích của chatbot AI 19
2.7. Ngrok 20
2.7.1. Giới thiệu Ngrok 20
2.7.2. Mục đích sử dụng 20
2.7.3. Nguyên lý hoạt động 21
2.7.4. Ưu điểm và Nhược điểm của Ngrok 21
2.8. React 22
2.8.1. Khái niệm 22
2.8.2. Kiến trúc Component 22
2.8.3. Virtual DOM 22

2.8.4. JSX (JavaScript XML) 23
2.8.5. State và Props (Trạng thái và Thuộc tính) 23
2.8.6. React Hooks 23
2.8.7. Lifecycle Methods (Các phương thức vòng đời) 24
2.8.8. Routing trong React 24
2.8.9. State Management (Quản lý trạng thái) 24
2.8.10. Ưu và nhược điểm của React 24
2.9. SQL Server 25
2.9.1. Khái niệm 25
2.9.2. Cấu trúc 26
2.10. EmailJS 26
2.10.1. Khái niệm EmailJS 26
2.10.2. Ứng dụng của EmailJS 27
2.10.3. Ưu điểm và nhược điểm EmailJS 27
2.11. Swagger và Postman 28
2.12. Các công trình nghiên cứu liên quan 29
CHƯƠNG 3: HIỆN THỰC HOÁ NGHIÊN CỨU 31
3.1. Mô tả hệ thống 31
3.2. Các chức năng của hệ thống 31
3.3. Thiết kế dữ liệu 35
3.3.1. Mô hình dữ liệu mức quan niệm 35
3.4. Mô tả thực thể 36
3.5. Thiết kế xử lý 44
3.5.1. Biểu đồ Use Case tác nhân vãng lai 44
3.5.2. Biểu đồ Use Case tác nhân khách hàng 45
3.5.3. Biểu đồ Use Case tác nhân quản trị 46

3.6. Kiến trúc hệ thống 46
3.6.1. Mô hình hoạt động hệ thống 48
3.6.2. Mô hình hoạt động của ChatbotAI 49
3.6.3. Mô hình hoạt động tính năng thử đồ ảo 49
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU 51
4.1. Giao diện khách hàng 51
4.1.1. Giao diện trang chủ 51
4.1.2. Giao diện đăng ký, đăng nhập 52
4.1.3. Giao diện sản phẩm 53
4.1.4. Giao diện chi tiết sản phẩm 53
4.1.5. Giao diện thử đồ ảo 54
4.1.6. Giao diện giỏ hàng 55
4.1.7. Giao diện thanh toán đơn hàng 56
4.1.8. Giao diện kiểm tra đơn hàng 57
4.1.9. Giao diện chat với trợ lí ảo AI 57
4.2. Giao diện quản trị 58
4.2.1. Giao diện thống kê 58
4.2.2. Giao diện quản lý đơn hàng 59
4.2.3. Giao diện quản lý danh mục 60
4.2.4. Giao diện quản lý sản phẩm 60
4.2.5. Giao diện quản lý tồn kho 61
4.2.6. Giao diện quản lý Blogs 62
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 63
5.1. Kết quả đạt được 63
5.2. Kết quả chưa đạt được 63
5.3. Hướng phát triển 64

DANH MỤC TÀI LIỆU THAM KHẢO..65

UBND TỈNH VĨNH LONG
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP

(Của giảng viên hướng dẫn)

Họ và tên sinh viên: Tô Văn Tới MSSV: 110121252
Ngành: Công nghệ Thông tin Khóa: 2021
Tên đề tài: Phát triển website bán thời trang tích hợp AI thử đồ ảo
Họ và tên Giáo viên hướng dẫn: Phạm Minh Đương
Chức danh: Giảng viên Học vị: Thạc sĩ

NHẬN XÉT

1. Nội dung đề tài:
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
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:

.....
.....
.....
.....

……., ngày tháng năm 20…

Giảng viên hướng dẫn

(Ký & ghi rõ họ tên)

UBND TỈNH VĨNH LONG
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

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

.....
.....
.....

3. Ứng dụng thực tế:

.....
.....
.....
.....
.....

.....

.....

II. CÁC VẤN ĐỀ CẦN LÀM RÕ

(Các câu hỏi của giáo viên phản biện)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

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)

KÍ HIỆU : CÁC CỤM TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
AI	Trí tuệ nhân tạo (Artificial Intelligence)
VTO	Thử đồ ảo (Virtual Try-On)
API	Giao diện lập trình ứng dụng (Application Programming Interface)
CRUD	Tạo, đọc, cập nhật, xoá (Create, Read, Update, Delete)
IDE	Môi trường phát triển tích hợp (Integrated Development Environment)
NLP	Xử lý ngôn ngữ tự nhiên (Natural Language Processing)

DANH MỤC BẢNG BIỂU

Bảng 3.1 Bảng Categories....36
Bảng 3.2 Bảng Products....36
Bảng 3.3 Bảng Customer....37
Bảng 3.4 Bảng CustomerAddresses....38
Bảng 3.5 Bảng Coupon....39
Bảng 3.6 Bảng Order....39
Bảng 3.7 Bảng OrderItem....40
Bảng 3.8 Bảng ProductCategory....41
Bảng 3.9 Bảng Blogs....41
Bảng 3.10 Bảng ProductImage....42
Bảng 3.11 Bảng ProductReview....42
Bảng 3.12 Bảng ChatHistory....43
Bảng 3.13 Bảng PaymentTransaction....43
Bảng 3.14 Bảng GoogleAccounts....44

DANH MỤC HÌNH ẢNH

Hình 2.1 Nguyên lý hoạt động mô hình thử đồ ảo.....5
Hình 2.2 .Net Core 7
Hình 2.3 Nguyên lý hoạt động MediatR.....9
Hình 2.4 Nguyên lý hoạt động của RESTful API.....11
Hình 2.5 Kiến trúc Clean Architecture 13
Hình 2.6 Chatbot AI trí tuệ nhân tạo.....19
Hình 2.7 Nguyên lý hoạt động của Ngrok 21
Hình 2.8 Cấu trúc của SQL Server 26
Hình 2.9 Nguyên lý hoạt động EmailJS.....27
Hình 3.1 Mô hình dữ liệu mức quan niệm.....35
Hình 3.2 Biểu đồ Use Case tác nhân vãng lai.....45
Hình 3.3 Biểu đồ Use Case tác nhân khách hàng 45
Hình 3.4 Biểu đồ Use Case tác nhân quản trị 46
Hình 3.5 Kiến trúc hệ thống.....47
Hình 3.6 Mô hình hoạt động hệ thống 48
Hình 3.7 Mô hình hoạt động chatbot AI 49
Hình 3.8 Mô hình hoạt động tính năng thử đồ ảo.....50
Hình 4.1 Giao diện trang chủ.....51
Hình 4.2 Giao diện đăng ký tài khoản 52
Hình 4.3 Giao diện đăng nhập tài khoản.....52
Hình 4.4 Giao diện sản phẩm.....53
Hình 4.5 Giao diện chi tiết sản phẩm.....54
Hình 4.6 Giao diện thử đồ ảo.....55
Hình 4.7 Giao diện giỏ hàng.....56
Hình 4.8 Giao diện thanh toán 56

Hình 4.9 Giao diện theo dõi đơn hàng 57
Hình 4.10 Giao diện Chat với trợ lý ảo AI 58
Hình 4.11 Giao diện thống kê 59
Hình 4.12 Giao diện quản lý đơn hàng 59
Hình 4.13 Giao diện quản lý danh mục 60
Hình 4.14 Giao diện quản lý sản phẩm 61
Hình 4.15 Giao diện quản lý tồn kho 61
Hình 4.16 Giao diện quản lý blog 62

CHƯƠNG 1: ĐẶT VẤN ĐỀ

1.1. Lý do chọn đề tài

Sự phát triển vượt bậc của công nghệ thông tin trong kỷ nguyên số đã và đang làm thay đổi mạnh mẽ cách con người mua sắm, tiêu dùng và tương tác với các sản phẩm. Trong đó, thương mại điện tử ngành thời trang nổi lên như một lĩnh vực có tốc độ tăng trưởng nhanh và tiềm năng phát triển lớn. Tuy nhiên, một trong những rào cản khiến người tiêu dùng còn e ngại khi mua quần áo trực tuyến chính là không thể thử trực tiếp, dẫn đến khó khăn trong việc đánh giá sự phù hợp của trang phục với vóc dáng của người dùng.

Nhằm giải quyết bài toán thực tiễn này, công nghệ trí tuệ nhân tạo (AI) ngày càng được ứng dụng rộng rãi, đặc biệt trong lĩnh vực xử lý ảnh và thị giác máy tính. Mô hình AI thử đồ ảo như TryOnGAN, cho phép mô phỏng hình ảnh người mặc quần áo ảo dựa trên ảnh thật của người dùng, từ đó mang lại trải nghiệm chân thực và cá nhân hóa hơn khi mua sắm trực tuyến. Bên cạnh đó, chatbot AI sử dụng công nghệ xử lý ngôn ngữ tự nhiên giúp website có thể tương tác với người dùng như một nhân viên tư vấn thời trang chuyên nghiệp.

Chính vì vậy, việc thực hiện đề tài phát triển website bán thời trang tích hợp AI thử đồ ảo mang tính cấp thiết và thực tiễn cao, vừa giải quyết nhu cầu của người tiêu dùng hiện đại, vừa thể hiện khả năng ứng dụng tổng hợp các công nghệ tiên tiến vào hệ thống phần mềm hoàn chỉnh.

1.2. Mục tiêu nghiên cứu

Đề tài hướng đến việc phát triển một nền tảng thương mại điện tử trong lĩnh vực thời trang có tích hợp các công nghệ hiện đại nhằm nâng cao trải nghiệm người dùng. Các mục tiêu cụ thể bao gồm:

Xây dựng website thương mại điện tử với giao diện hiện đại, thân thiện người dùng. Tích hợp tính năng thử đồ ảo bằng công nghệ AI, cho phép người dùng tải ảnh chân dung hoặc toàn thân để thử các sản phẩm thời trang trên chính hình ảnh của mình. Phát triển hoặc tích hợp chatbot AI có khả năng tương tác tự nhiên, hỗ trợ người dùng trong việc lựa chọn sản phẩm phù hợp, giải đáp thắc mắc và hỗ trợ sau bán hàng.

Hệ thống có khả năng quản lý sản phẩm, người dùng, đơn hàng, thanh toán một cách hiệu quả và bảo mật. Đảm bảo khả năng mở rộng hệ thống trong tương lai, có thể tích hợp thêm các mô-đun phân tích hành vi khách hàng, đề xuất sản phẩm cá nhân hóa dựa trên AI.

1.3. Phạm vi và đối tượng nghiên cứu

Phạm vi thực hiện của đề tài được xác định rõ ràng nhằm đảm bảo tính khả thi trong thời gian thực hiện:

Về chức năng: Xây dựng các chức năng chính như đăng ký/đăng nhập người dùng, quản lý danh mục sản phẩm, giỏ hàng, thanh toán, tải ảnh thử đồ ảo, và chatbot tư vấn.

Về công nghệ: Sử dụng ReactJS cho frontend, ASP.NET Core cho backend và kết hợp Python cho các xử lý liên quan đến mô hình AI.

Về AI: Nghiên cứu, tích hợp mô hình thử đồ ảo dựa trên kiến trúc TryOnGAN và chatbot AI Gemini sử dụng xử lý ngôn ngữ tự nhiên.

Đối tượng nghiên cứu bao gồm:

Người tiêu dùng có nhu cầu mua sắm thời trang trực tuyến, đặc biệt là giới trẻ và người thường xuyên sử dụng thiết bị di động.

Các công nghệ và framework liên quan đến xây dựng website, xử lý ảnh bằng AI, và chatbot AI.

1.4. Phương pháp thực hiện

Đề tài sử dụng các phương pháp nghiên cứu chủ yếu sau đây:

Phương pháp nghiên cứu lý thuyết: Thu thập, nghiên cứu tài liệu từ sách, báo cáo khoa học, các bài viết, tutorial chính thức về ReactJS, ASP.NET Core, các mô hình AI thử đồ TryOnGAN và chatbot Gemini.

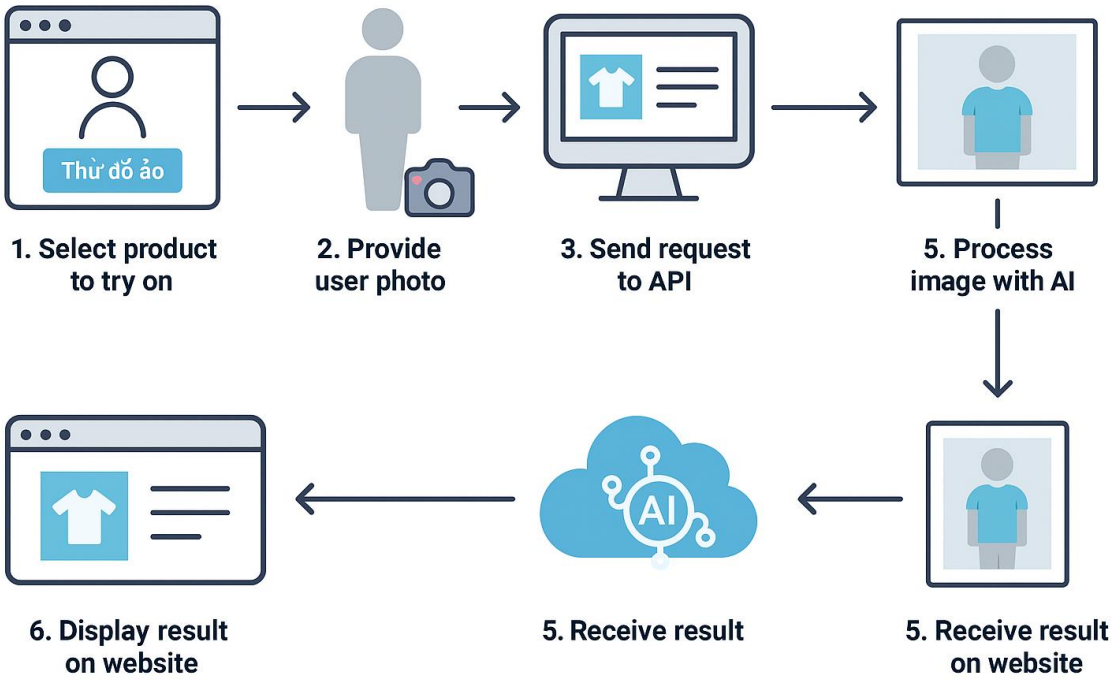
Phương pháp khảo sát thực tế: Tiến hành khảo sát người dùng mục tiêu về nhu cầu và thói quen mua sắm thời trang trực tuyến, từ đó xác định các tính năng quan trọng cần ưu tiên phát triển.

Phương pháp thực nghiệm: Thiết kế hệ thống, xây dựng website, tích hợp mô hình AI thử đồ và chatbot vào hệ thống, kiểm thử và đánh giá kết quả trên người dùng thật.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Các mô hình AI thử đồ ảo

Công nghệ thử đồ ảo (Virtual Try-On) là một lĩnh vực đang phát triển mạnh mẽ trong ứng dụng AI, đặc biệt là trong xử lý ảnh và thị giác máy tính, cho phép ghép quần áo kỹ thuật số lên cơ thể người trong ảnh [1].



Hình 2.1 Nguyên lý hoạt động mô hình thử đồ ảo

VITON (2017)

Mô hình đầu tiên trong VTO sử dụng phương pháp chia ảnh (segmentation) và cấu trúc U-Net.

Quá trình gồm hai bước: tạo ảnh thô với quần áo ghép lên người rồi dùng mạng điều chỉnh (refinement network) làm rõ chi tiết, mô phỏng quần áo biến dạng tự nhiên và giữ nguyên hoa văn.

VITON-HD (2021)

Nâng cấp lên định dạng ảnh độ phân giải cao (1024×768). Sử dụng ALIgnment-Aware Segment (ALIAS) normalization để xử lý khu vực lệch so với vùng mặc quần áo, giữ chi tiết sắc nét và tránh lỗi misalignment.

C-VTON (2022)

Cải tiến hơn bằng cách đưa thông tin ngữ cảnh (context-aware) vào cả quá trình định vị và tạo ảnh.

Sử dụng chức năng ghép hình hình học (geometric matching) kết hợp mạng tạo ảnh để xử lý tư thế phức tạp và phần bị che phủ đồng thời vẫn giữ độ chân thực cao.

IDM-VTON (2024 – diffusion-based)

Chuyển sang dùng mô hình diffusion (lan truyền) theo kiểu hai luồng (two-stream conditional diffusion).

Một luồng đảm nhiệm semantic-level (qua visual encoder và cross-attention), luồng còn lại tập trung vào low-level texture (qua self-attention).

Cho hình ảnh kết quả rất thực, phù hợp nhiều kiểu trang phục và thân hình người.

MC-VTON (2025 – diffusion transformer)

Mô hình khuếch đại mới nhất sử dụng diffusion transformer (DiT).

Loại bỏ nhu cầu về pose estimation, human parsing; chỉ cần ảnh người được mask và ảnh quần áo.

Hiệu quả hơn với ít bước inference (chỉ 8 bước) và giữ chi tiết cao cùng cấu trúc mạng.

Các mô hình này thường yêu cầu ảnh người đầu vào, ảnh quần áo (không có nền) và các thông tin bổ sung như pose, human parsing. Việc lựa chọn mô hình phù hợp tùy thuộc vào độ phức tạp và yêu cầu hiệu năng của hệ thống.

2.2. .NET Core, Entity Framework Core và MediatR

2.2.1. .NET Core

.NET Core là một nền tảng phát triển ứng dụng đa nền tảng và mã nguồn mở, được phát triển bởi Microsoft. Nó hỗ trợ tốt cho việc xây dựng các ứng dụng web, dịch vụ API, ứng dụng console và các dịch vụ microservices.

Hình 2.2 .Net Core

Một số tính năng nổi bật của .NET Core:

Đa nền tảng (Cross-platform): .NET Core có thể chạy trên Windows, Linux, và macOS, điều này giúp phát triển các ứng dụng có tính khả chuyển cao.

Hiệu suất cao (High Performance): .NET Core được tối ưu để có hiệu suất cao, đặc biệt là với các ứng dụng web và dịch vụ RESTful. Nó cung cấp các công cụ như Kestrel, một web server nhẹ và nhanh, giúp tăng tốc độ xử lý và tiết kiệm tài nguyên.

Hỗ trợ cho các kiến trúc hiện đại: .NET Core tích hợp tốt với Clean Architecture, Microservices, và Cloud-native development, giúp các nhà phát triển dễ dàng xây dựng các hệ thống phức tạp.

Dependency Injection (DI): .NET Core cung cấp tích hợp Dependency Injection sẵn có, giúp quản lý các phụ thuộc một cách dễ dàng và nâng cao tính mở rộng cũng như khả năng kiểm thử.

.NET Core đã phát triển thành .NET 6/7 (từ năm 2021 trở đi), trở thành một nền tảng hợp nhất, giúp các nhà phát triển dễ dàng lựa chọn và sử dụng cho tất cả các loại ứng dụng: từ desktop, web, cloud, cho đến IoT và AI [2], [3].

2.2.2. Entity Framework Core

Entity Framework Core (EF Core) là một ORM (Object-Relational Mapper) hiện đại và nhẹ, giúp các nhà phát triển làm việc với cơ sở dữ liệu theo hướng đối tượng thay

vì viết các câu truy vấn SQL thủ công. EF Core làm việc với nhiều hệ quản trị cơ sở dữ liệu khác nhau như SQL Server, SQLite, MySQL và PostgreSQL [4].

Một số tính năng chính của Entity Framework Core:

Làm việc với dữ liệu qua các đối tượng: EF Core cho phép ánh xạ các bảng trong cơ sở dữ liệu thành các lớp trong code, từ đó giúp các nhà phát triển thao tác với dữ liệu một cách trực quan và đơn giản thông qua các đối tượng.

Hỗ trợ Migrations: Migration giúp quản lý các thay đổi trong schema cơ sở dữ liệu một cách dễ dàng, giảm thiểu lỗi khi cập nhật cấu trúc của cơ sở dữ liệu. Có thể thực hiện các lệnh như Add-Migration, Update-Database để áp dụng các thay đổi này.

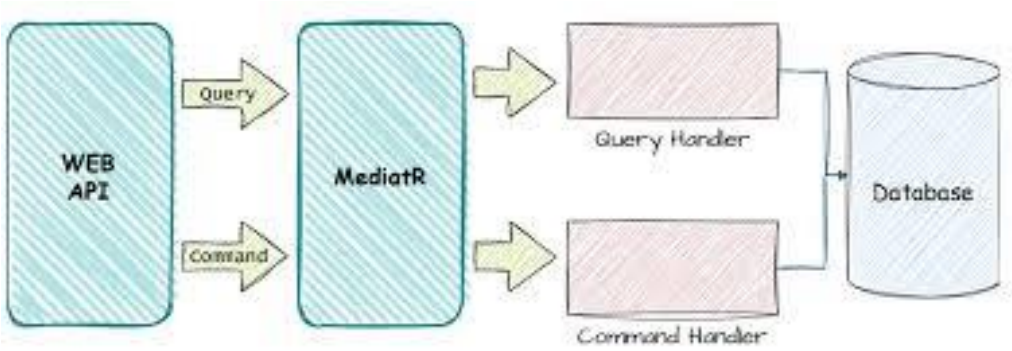
Query Linq mạnh mẽ: EF Core cung cấp khả năng thực hiện các truy vấn sử dụng **LINQ** (Language Integrated Query), giúp viết các câu truy vấn dễ đọc và dễ hiểu hơn so với câu truy vấn SQL truyền thống. LINQ cho phép sử dụng cú pháp mạnh mẽ, linh hoạt và được kiểm tra lỗi biên dịch.

Hỗ trợ Lazy Loading, Eager Loading và Explicit Loading: EF Core cung cấp nhiều cách để tải dữ liệu, bao gồm Lazy Loading (tải dữ liệu khi được yêu cầu), Eager Loading (tải dữ liệu ngay lập tức cùng với thực thể chính), và Explicit Loading (tải dữ liệu khi được yêu cầu một cách thủ công).

EF Core giúp tăng tính năng bảo trì, làm cho mã nguồn dễ đọc hơn bằng cách giảm sự phụ thuộc vào các câu truy vấn SQL tĩnh, đồng thời cũng giúp tránh lỗi bảo mật SQL Injection khi các câu truy vấn được viết một cách động và an toàn.

2.2.3. MediatR

MediatR là một thư viện giúp thực hiện CQRS (Command Query Responsibility Segregation) và giảm thiểu sự phụ thuộc cứng giữa các lớp trong hệ thống bằng cách sử dụng Mediator Pattern. Trong Clean Architecture, MediatR được sử dụng để tách biệt rõ ràng giữa các hành động và truy vấn, từ đó giúp hệ thống dễ dàng mở rộng và bảo trì [5].



Hình 2.3 Nguyên lý hoạt động MediatR

Một số đặc điểm chính của MediatR:

CQRS Pattern: CQRS là một mẫu kiến trúc tách biệt giữa Command (các lệnh để thay đổi dữ liệu) và Query (các truy vấn để lấy dữ liệu). MediatR giúp triển khai CQRS một cách dễ dàng bằng cách chia các lớp xử lý nghiệp vụ thành các lớp CommandHandler và QueryHandler. Điều này giúp hệ thống có tính linh hoạt cao hơn, đặc biệt khi mở rộng các yêu cầu nghiệp vụ.

Giảm sự phụ thuộc giữa các lớp: MediatR cho phép các lớp có thể giao tiếp với nhau thông qua Mediator mà không cần phải tham chiếu trực tiếp đến nhau. Các thành phần chỉ cần đăng ký hoặc lắng nghe các lệnh (command) hoặc truy vấn (query) mà chúng quan tâm. Điều này giúp hệ thống ít bị thay đổi hơn khi cần bổ sung hoặc điều chỉnh các chức năng.

Pipeline Behaviors: MediatR cũng hỗ trợ khái niệm Pipeline Behaviors, giúp chèn các logic xử lý trước hoặc sau khi thực hiện các Handler. Ví dụ, có thể thêm các chức năng như logging, validation, hoặc authorization cho các lệnh và truy vấn mà không cần thay đổi code xử lý chính.

2.2.4. Kết hợp .Net Core, Entity Framework Core và MediatR

Khi phát triển một ứng dụng theo kiến trúc Clean Architecture, .NET Core, EF Core, và MediatR thường được sử dụng kết hợp với nhau để tận dụng tối đa các ưu điểm của từng công cụ.

Sử dụng Dependency Injection (DI)

.NET Core tích hợp DI để tiêm các phụ thuộc một cách dễ dàng. DbContext của EF Core có thể được thêm vào các Repository, còn MediatR có thể được sử dụng để gọi các lệnh và truy vấn.

Triển khai CQRS với MediatR

Tạo các lớp Command để tạo, cập nhật, hoặc xóa tài nguyên.

Tạo các lớp Query để lấy thông tin tài nguyên.

Sử dụng Handler để xử lý logic cho từng Command và Query. Handler sử dụng Repository để thao tác với cơ sở dữ liệu thông qua EF Core.

Một API CreateCustomer sẽ gửi một yêu cầu đến Application Layer để tạo khách hàng mới thông qua CreateCustomerCommand. MediatR sẽ tìm Handler tương ứng để thực thi lệnh này, sử dụng EF Core để lưu khách hàng vào cơ sở dữ liệu.

Khi một yêu cầu GetCustomerById được gửi tới, một GetCustomerByIdQuery được khởi tạo và gửi tới MediatR để tìm QueryHandler và thực hiện việc lấy dữ liệu.

2.2.5. Lợi ích khi kết hợp .Net Core, Entity Framework Core và MediatR

Tính mở rộng cao: Việc sử dụng MediatR giúp tách biệt rõ ràng các lớp xử lý nghiệp vụ, giúp hệ thống dễ dàng mở rộng khi các yêu cầu thay đổi.

Dễ bảo trì: .NET Core kết hợp với Clean Architecture và MediatR giúp tổ chức mã nguồn rõ ràng và dễ bảo trì hơn, khi mỗi thành phần đều có trách nhiệm cụ thể và không gây ảnh hưởng đến các thành phần khác.

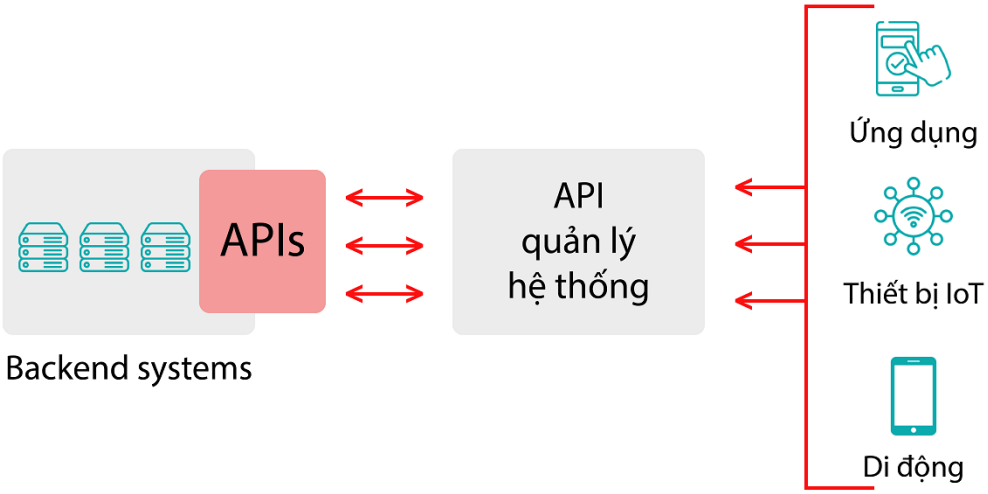
Tránh sự phụ thuộc cứng: Sử dụng MediatR giảm bớt sự phụ thuộc giữa các lớp trong hệ thống. Các lớp chỉ cần giao tiếp thông qua các abstraction mà không cần biết chi tiết về các lớp khác.

Tích hợp ORM mạnh mẽ: EF Core giúp giảm thiểu sự phức tạp trong thao tác với cơ sở dữ liệu, giúp nhà phát triển tập trung vào logic nghiệp vụ thay vì xử lý các câu truy vấn SQL phức tạp.

2.3. RESTful API

2.3.1. Khái niệm RESTful

RESTful API là một phương thức giao tiếp giữa các hệ thống phần mềm, tuân theo nguyên tắc của kiến trúc REST (Representational State Transfer). API này cho phép truy cập và thao tác với dữ liệu thông qua các phương thức HTTP tiêu chuẩn như GET, POST, PUT, PATCH và DELETE. Nhờ tính đơn giản, dễ tích hợp và hiệu quả, RESTful API được sử dụng rộng rãi trong các ứng dụng web và di động hiện nay [6].



Hình 2.4 Nguyên lý hoạt động của RESTful API

2.3.2. Phương thức HTTP chính

Các phương thức HTTP trong RESTful API có vai trò cụ thể như sau:

GET: Lấy dữ liệu từ máy chủ, không làm thay đổi trạng thái tài nguyên.

POST: Tạo mới một tài nguyên.

PUT: Cập nhật toàn bộ một tài nguyên hiện có.

PATCH: Cập nhật một phần thông tin của tài nguyên.

DELETE: Xóa tài nguyên khỏi hệ thống.

2.3.3. Tài nguyên, URI và định dạng dữ liệu

Trong kiến trúc REST, mỗi tài nguyên (resource) là một thực thể dữ liệu và được định danh duy nhất qua URI (Uniform Resource Identifier). Ví dụ, GET /users/123 sẽ truy vấn thông tin người dùng có ID là 123. Quy ước đặt tên tài nguyên thường ở dạng số nhiều (ví dụ: /products, /orders), đồng thời tránh sử dụng động từ trong đường dẫn mà thay vào đó sử dụng đúng phương thức HTTP tương ứng.

RESTful API thường sử dụng định dạng dữ liệu JSON nhờ tính gọn nhẹ, dễ đọc và dễ tích hợp với JavaScript. Ngoài JSON, một số hệ thống cũng hỗ trợ XML, tuy nhiên ít phổ biến hơn.

2.3.4. Mã trạng thái HTTP

Mỗi phản hồi từ API nên đi kèm mã trạng thái HTTP để mô tả kết quả xử lý:

200 OK: Thành công.

201 Created: Tạo mới thành công.

400 Bad Request: Lỗi từ phía người dùng (ví dụ sai tham số).

401 Unauthorized: Không có quyền truy cập.

404 Not Found: Không tìm thấy tài nguyên.

500 Internal Server Error: Lỗi hệ thống phía máy chủ.

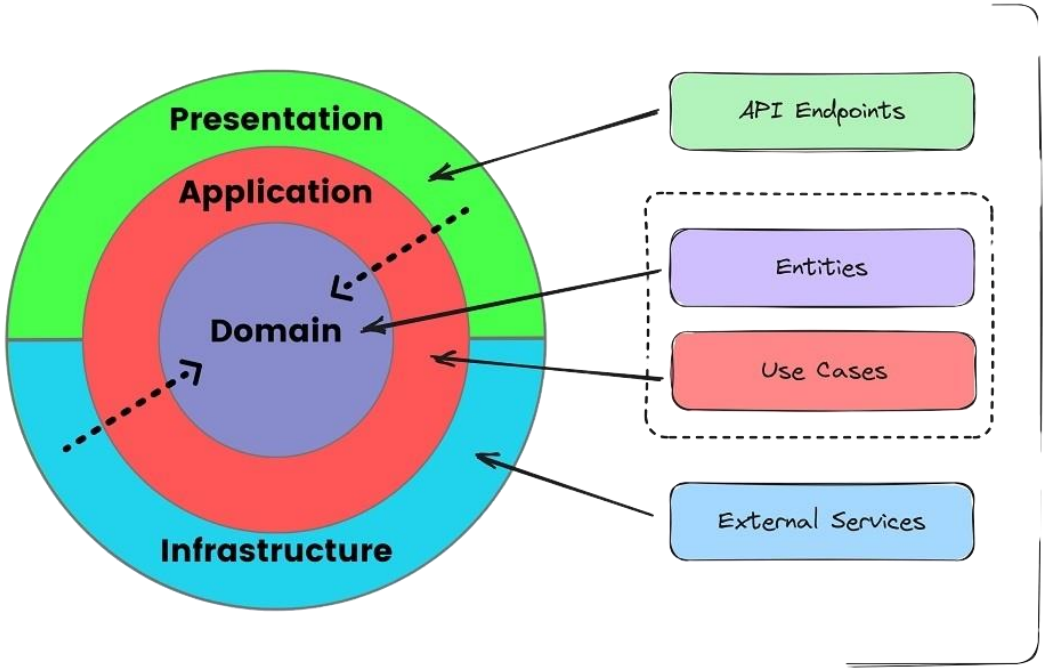
2.3.5. Ưu điểm và nhược điểm của RESTful API

RESTful API có nhiều ưu điểm nổi bật, trở thành lựa chọn phổ biến trong phát triển hệ thống. Đơn giản và dễ sử dụng nhờ giao tiếp thông qua HTTP với các phương thức chuẩn như GET, POST, PUT, DELETE giúp lập trình viên dễ học và dễ triển khai. Thiết kế không trạng thái (stateless) cũng là một điểm mạnh, giúp API dễ mở rộng và phục vụ nhiều client cùng lúc mà không cần lưu trạng thái phiên làm việc. Ngoài ra, RESTful API có tính tương thích cao, có thể hoạt động tốt trên nhiều nền tảng và ngôn ngữ lập trình khác nhau. Khi kết hợp với bộ nhớ đệm (cache), API giúp giảm tải cho server và cải thiện hiệu suất tổng thể. Nhờ cấu trúc rõ ràng và chuẩn hóa, RESTful API rất dễ tích hợp vào các hệ thống hiện có như ứng dụng web, mobile hoặc dịch vụ đám mây. Tuy nhiên không phù hợp cho các ứng dụng thời gian thực, vì hoạt động theo mô hình yêu cầu phản hồi, không hỗ trợ kết nối liên tục như WebSocket, khiến việc cập nhật dữ liệu tức thời trở nên khó khăn. Ngoài ra, REST kém hiệu quả khi xử lý khối lượng dữ liệu lớn hoặc có cấu trúc phức tạp, mỗi yêu cầu đều phải mang đầy đủ thông tin, dễ dẫn đến tình trạng quá tải mạng. Một hạn chế khác là REST không có chuẩn ràng buộc chặt chẽ về mặt thiết kế, dẫn đến nguy cơ thiếu thống nhất giữa các hệ thống nếu không có quy ước cụ thể, gây khó khăn cho việc bảo trì và mở rộng lâu dài [7].

2.4. Kiến Trúc Clean Architecture

Clean Architecture do Robert C. Martin đề xuất, phân chia ứng dụng thành các tầng với trách nhiệm rõ ràng, bao gồm Presentation, Application, Domain, và Infrastructure. Mô hình này giúp giảm thiểu sự phụ thuộc lẫn nhau giữa các lớp, tạo điều kiện bảo trì và mở rộng.

Kiến trúc Clean Architecture giúp đảm bảo hệ thống có tính linh hoạt, dễ dàng thay thế các thành phần mà không ảnh hưởng toàn hệ thống, đáp ứng yêu cầu mở rộng và tích hợp trong tương lai [8].



Hình 2.5 Kiến trúc Clean Architecture

2.4.1. Mục Tiêu

Clean Architecture hướng tới việc giải quyết các vấn đề như tính khó bảo trì, sự phụ thuộc lẫn nhau giữa các thành phần và sự phức tạp trong việc mở rộng hệ thống. Một số mục tiêu chính bao gồm:

Độc lập với Framework: Các framework chỉ được sử dụng như công cụ hỗ trợ. Chúng không chi phối kiến trúc chính của ứng dụng, đảm bảo rằng việc thay thế framework không ảnh hưởng lớn đến hệ thống.

Dễ dàng kiểm thử: Hệ thống được chia thành nhiều lớp với trách nhiệm cụ thể, giúp đơn giản hóa quá trình kiểm thử, bao gồm kiểm thử đơn vị (unit test).

Tách biệt giao diện người dùng (UI): Việc thay đổi UI có thể thực hiện mà không cần thay đổi logic nghiệp vụ, giúp tăng khả năng mở rộng cho nhiều nền tảng khác nhau.

Độc lập với cơ sở dữ liệu: Cơ sở dữ liệu chỉ là một trong những thành phần lưu trữ, và có thể thay đổi dễ dàng mà không làm ảnh hưởng đến các logic của ứng dụng.

Độc lập với logic nghiệp vụ: Mọi nghiệp vụ cốt lõi của ứng dụng đều độc lập với các yếu tố hạ tầng như cơ sở dữ liệu, API hoặc thư viện.

2.4.2. Các thành phần chính trong Clean Architecture

Clean Architecture được chia thành nhiều lớp khác nhau. Các lớp này có thứ tự phụ thuộc từ ngoài vào trong, và mỗi lớp chỉ có thể phụ thuộc vào các lớp nằm bên trong nó. Các thành phần chính bao gồm:

Entities (Domain Layer - Lớp Miền)

Entities là các đối tượng nghiệp vụ cốt lõi của ứng dụng. Chúng biểu diễn các quy tắc nghiệp vụ và các thuộc tính liên quan đến ứng dụng. Lớp này có thể chứa các định nghĩa lớp, các giá trị bất biến và các quy tắc xử lý liên quan.

Entities không phụ thuộc vào bất kỳ lớp nào khác. Chúng là nền tảng của toàn bộ ứng dụng và thường được kiểm thử độc lập.

Use Cases (Application Layer - Lớp Ứng Dụng)

Lớp Use Cases (hoặc Interactors) chứa logic nghiệp vụ ứng dụng. Nó đại diện cho các trường hợp sử dụng của hệ thống (những điều người dùng có thể làm với hệ thống).

Các Use Case định nghĩa các hành động mà hệ thống có thể thực hiện và chúng thường sử dụng các Entities để hoàn thành nhiệm vụ. Lớp này không biết gì về giao diện người dùng hoặc cơ sở dữ liệu. Điều này giúp logic nghiệp vụ không bị phụ thuộc vào các yếu tố khác của hệ thống.

Interface Adapters (Presentation Layer - Lớp Giao Diện)

Presentation Layer hoặc Interface Adapters chịu trách nhiệm chuyển đổi dữ liệu giữa các lớp khác nhau. Điều này có thể là chuyển đổi từ các Entities thành dữ liệu mà lớp giao diện người dùng cần hoặc ngược lại.

Lớp này chứa các thành phần như Controllers, Presenters, Views hoặc bất kỳ phần nào liên quan đến giao tiếp giữa người dùng và hệ thống. Interface Adapters giúp các Use Case có thể giao tiếp với hệ thống bên ngoài mà không cần phải biết cách hệ thống đó hoạt động.

Infrastructure Layer (Lớp Hạ Tầng)

Infrastructure Layer chịu trách nhiệm cung cấp các dịch vụ cần thiết cho ứng dụng, chẳng hạn như cơ sở dữ liệu, các dịch vụ lưu trữ, hoặc các thư viện, framework.

Lớp này chứa các thành phần liên quan đến lưu trữ dữ liệu, triển khai framework, tích hợp với các dịch vụ bên thứ ba. Có thể sử dụng Entity Framework, MongoDB, hoặc bất kỳ công cụ nào khác để thao tác với cơ sở dữ liệu.

Các lớp bên trên không phụ thuộc trực tiếp vào hạ tầng, mà phụ thuộc vào các interface, do đó Infrastructure có thể dễ dàng được thay thế.

2.4.3. Nguyên tắc Dependency Rule (quy tắc phụ thuộc)

Một trong những nguyên tắc quan trọng của Clean Architecture là Dependency Rule. Theo quy tắc này, tất cả các phụ thuộc phải hướng từ ngoài vào trong. Điều này có nghĩa là các lớp bên ngoài (như giao diện người dùng hoặc cơ sở dữ liệu) không được phép ảnh hưởng trực tiếp đến các lớp bên trong (như Domain hoặc Use Cases). Các lớp bên trong không biết gì về các lớp bên ngoài. Chúng chỉ có thể phụ thuộc vào các abstraction (interface hoặc contract), giúp giảm thiểu sự phụ thuộc cứng và tăng tính linh hoạt khi thay đổi hệ thống.

2.4.4. Ưu và nhược điểm của Clean Architecture

Kiến trúc này giúp hệ thống dễ bảo trì và mở rộng nhờ việc phân tách rõ ràng các lớp theo chức năng, cho phép thay đổi từng phần mà không ảnh hưởng đến toàn bộ hệ thống. Một điểm mạnh khác là tính độc lập với framework và hạ tầng tức là có thể linh hoạt thay đổi công nghệ như framework, cơ sở dữ liệu hay giao diện người dùng mà không cần chỉnh sửa logic nghiệp vụ cốt lõi. Clean Architecture còn hỗ trợ khả năng kiểm thử cao, vì logic nghiệp vụ được tách biệt hoàn toàn khỏi các phần phụ thuộc như UI hoặc database, giúp việc viết và thực thi các bài kiểm thử trở nên dễ dàng và hiệu quả hơn.

Kiến trúc này có thể làm hệ thống trở nên phức tạp hơn, không phù hợp với những dự án nhỏ hoặc có quy mô trung bình. Bên cạnh đó, Clean Architecture thường yêu cầu tạo ra nhiều lớp và abstraction, điều này dễ dẫn đến việc mã nguồn trở nên rườm rà và khó quản lý nếu không có kinh nghiệm trong việc tổ chức và duy trì cấu trúc hệ thống một cách hợp lý [2].

2.4.5. Áp dụng Clean Architecture trong .NET Core

.NET Core là nền tảng lý tưởng để áp dụng Clean Architecture do tính linh hoạt và khả năng hỗ trợ đa nền tảng. Các bước áp dụng kiến trúc Clean Architecture với .NET Core bao gồm:

Tạo các Dự Án Con (Subprojects)

Domain Project: Chứa các lớp entities và các interface của repository (repository interfaces).

Application Project: Chứa các lớp xử lý logic nghiệp vụ và các Use Case.

Infrastructure Project: Chứa các triển khai chi tiết cho repository, dịch vụ lưu trữ, và các thành phần phụ thuộc.

Web API Project: Chứa các controller để xử lý các yêu cầu HTTP. Đây là nơi API giao tiếp với người dùng.

Sử Dụng Dependency Injection (DI)

Sử dụng DI để tiêm các phụ thuộc vào các lớp, giúp tăng tính mở rộng và giảm thiểu sự phụ thuộc cứng. DI là công cụ chính để áp dụng nguyên tắc Dependency Rule trong .NET Core.

Sử Dụng Các Interface Cho Các Thành Phần Hạ Tầng

Tạo các interface cho việc lưu trữ dữ liệu, dịch vụ bên thứ ba và các lớp infrastructure sẽ triển khai các interface này. Các lớp nghiệp vụ chỉ tương tác với interface, giúp dễ dàng thay thế các thành phần này khi kiểm thử.

Tích Hợp MediatR

Sử dụng MediatR để áp dụng CQRS và xử lý các lệnh và truy vấn. MediatR giúp giảm bớt sự phụ thuộc giữa các lớp nghiệp vụ và tăng tính tách biệt giữa các xử lý nghiệp vụ và tầng giao tiếp.

2.4.6. Tóm tắt quy trình làm việc trong Clean Architecture

UI gửi yêu cầu đến lớp Application, nơi chứa các Use Cases.

Application Layer sử dụng Domain Layer (các Entities và logic nghiệp vụ) để thực hiện yêu cầu.

Infrastructure Layer cung cấp các chi tiết kỹ thuật như lưu trữ dữ liệu hoặc gọi API bên ngoài.

Dependency Injection đảm bảo rằng tất cả các lớp chỉ phụ thuộc vào abstraction.

2.5. MimeKit

2.5.1. Khái niệm

MimeKit là một API .NET mã nguồn mở có thể được sử dụng để tạo, phân tích cú pháp và xử lý các tin nhắn MIME (Phần mở rộng thư Internet đa năng). Nó có một bộ tính năng toàn diện để xử lý định dạng email, tệp đính kèm, mã hóa và chữ ký. API này rất giàu tính năng và cung cấp khả năng tương thích với các tiêu chuẩn emial hiện đại, khiến nó trở thành lựa chọn tốt nhất cho các nhà phát triển đang tìm cách làm việc với nội dung email trong các ứng dụng .NET của họ. MimeKit hỗ trợ S/MIME, PGP và DKIM, cho phép liên lạc email an toàn và thường được sử dụng cùng với MailKit để cung cấp giải pháp hoàn chỉnh cho việc xử lý email [10].

2.5.2. Chức năng của Mimekit

Tạo và phân tích cú pháp email MIME

MimeKit giúp bạn dễ dàng tạo email với các phần tử như tiêu đề (headers), người gửi (from), người nhận (to, cc, bcc), và các phần thân email (body) hỗ trợ nhiều loại nội dung khác nhau như text và HTML.

Phân tích cú pháp email (parse) để đọc và xử lý email đã nhận từ các định dạng MIME.

Hỗ trợ tệp đính kèm

MimeKit cho phép bạn gửi email với các tệp đính kèm dưới nhiều định dạng khác nhau (hình ảnh, tài liệu). Hỗ trợ thêm các tệp đính kèm vào email dễ dàng, thậm chí với các mã hóa đặc biệt.

Mã hóa và giải mã (Encoding/Decoding)

MimeKit hỗ trợ các phương pháp mã hóa email như Base64 và Quoted-Printable, giúp gửi các nội dung đặc biệt hoặc các ký tự không thể hiển thị đúng trong email một cách an toàn.

Tích hợp với SmtpClient và Pop3Client

MimeKit có thể được sử dụng kết hợp với thư viện MailKit (cũng của tác giả MimeKit) để gửi email qua SMTP, hoặc nhận email qua POP3/IMAP, giúp bạn xây dựng một hệ thống email hoàn chỉnh.

Quản lý định dạng và các mã hóa của văn bản

Với MimeKit, bạn có thể dễ dàng quản lý việc chuyển đổi và hiển thị các định dạng văn bản khác nhau như HTML, PlainText, và các kiểu MIME khác.

Hỗ trợ nhiều ngôn ngữ

MimeKit hỗ trợ nhiều ngôn ngữ, bao gồm khả năng xử lý các tiêu đề email với mã hóa đặc biệt cho các ký tự không phải ASCII (như UTF-8).

2.6. ChatBot trí tuệ nhân tạo

2.6.1. Giới thiệu

Chatbot trí tuệ nhân tạo là một ứng dụng phần mềm trí tuệ nhân tạo được thiết kế để thực hiện các cuộc trò chuyện tự động với con người thông qua các nền tảng trò chuyện. Cụ thể, Chatbot có khả năng được lập trình để tự động phản hồi các câu hỏi, giải quyết vấn đề, thực hiện các tác vụ đơn giản, cung cấp tư vấn về sản phẩm/dịch vụ, hoặc cung cấp thông tin liên quan đến lĩnh vực hoặc nhà cung cấp cụ thể. Hoạt động dựa trên các thuậttoán học máy và xử lý ngôn ngữ tự nhiên, Chatbot trí tuệ nhân tạo có khả năng hiểu và đáp ứng các yêu cầu của người dùng.

2.6.2. Nguyên tắc hoạt động

Thu thập thông tin: Chatbot trí tuệ nhân tạo thực hiện quá trình thu thập thông tin từ nhiều nguồn khác nhau, bao gồm cơ sở dữ liệu, hệ thống quản lý quan hệ khách hàng, tài liệu văn bản, và các cuộc trò chuyện trước đó. Mục tiêu là đảm bảo rằng chatbot cung cấp câu trả lời chính xác và đầy đủ cho người dùng.

Xử lý ngôn ngữ tự nhiên: Chatbot trí tuệ nhân tạo sử dụng các thuật toán xử lý ngôn ngữ tự nhiên để hiểu ngôn ngữ của người dùng và phản hồi tự động. Bằng cách này, chatbot có khả năng nhận biết từ khóa và ý nghĩa của câu hỏi và yêu cầu từ phía người dùng.

Tích hợp với hệ thống khác: Chatbot trí tuệ nhân tạo có khả năng kết nối với nhiều hệ thống khác nhau, chẳng hạn như hệ thống quản lý quan hệ khách hàng, quản

lý dữ liệu, hệ thống thanh toán trực tuyến. Điều này giúp chatbot đáp ứng hiệu quả các nhu cầu của người dùng và tương tác linh hoạt với các hệ thống có sẵn.

Hình 2.6 Chatbot AI trí tuệ nhân tạo

2.6.3. Ứng dụng Chatbot trong thương mại điện tử thời trang

Tư vấn sản phẩm gợi ý quần áo phù hợp theo nhu cầu, phong cách, kích thước người dùng. Hỗ trợ mua hàng giải đáp câu hỏi về thanh toán, chính sách đổi trả, hướng dẫn đặt hàng. Tự động hóa dịch vụ khách hàng giảm tải cho nhân viên hỗ trợ, hoạt động 24/7.

2.6.4. Lợi ích của chatbot AI

Tăng trải nghiệm khách hàng: Chatbot trí tuệ nhân tạo giúp khách hàng tiếp cận thông tin và dịch vụ của doanh nghiệp một cách nhanh chóng và thuận tiện hơn. Trợ lý ảo hoạt động liên tục có thể giải đáp câu hỏi, cung cấp lời khuyên và hỗ trợ khách hàng trong quá trình mua sắm hoặc sử dụng sản phẩm/dịch vụ.

Tiết kiệm chi phí: Chatbot trí tuệ nhân tạo giúp doanh nghiệp giảm chi phí nhân sự và đào tạo. Thay vì cần thuê nhân viên để giải quyết câu hỏi của khách hàng, doanh nghiệp có thể sử dụng chatbot trí tuệ nhân tạo để xử lý những vấn đề đó.

Tăng hiệu quả làm việc: Chatbot trí tuệ nhân tạo có thể tự động hóa quy trình như

chăm sóc khách hàng, xử lý đơn hàng, đặt lịch hẹn, từ đó tăng hiệu quả làm việc của doanh nghiệp.

Thu thập dữ liệu và phân tích: Chatbot trí tuệ nhân tạo thu thập dữ liệu về hành vi khách hàng, giúp doanh nghiệp hiểu rõ hơn về nhu cầu và mong muốn của họ. Thông tin này có thể được sử dụng để cải thiện sản phẩm, dịch vụ, và nâng cao trải nghiệm khách hàng.

Tăng tính tương tác: Chatbot trí tuệ nhân tạo có thể được lập trình để tương tác thú vị và hấp dẫn, tăng cường tính tương tác giữa khách hàng và doanh nghiệp.

2.7. Ngrok

2.7.1. Giới thiệu Ngrok

Ngrok là một công cụ hỗ trợ lập trình viên chia sẻ ứng dụng web đang chạy trên máy tính cá nhân (localhost) ra môi trường Internet một cách nhanh chóng và an toàn. Thông qua việc tạo một "đường hầm" (tunnel), Ngrok cho phép người khác truy cập vào server nội bộ mà không cần triển khai lên hosting hay máy chủ thật [1].

2.7.2. Mục đích sử dụng

Ngrok được sử dụng trong nhiều tình huống thực tế của quá trình phát triển phần mềm, cụ thể như:

Chia sẻ ứng dụng đang phát triển: Cho phép gửi link truy cập đến ứng dụng chạy trên máy local cho khách hàng, người quản lý hoặc giảng viên mà không cần triển khai lên server.

Kiểm thử webhook từ các dịch vụ bên ngoài: Dùng để nhận phản hồi từ các hệ thống như:

Cổng thanh toán (MoMo, VNPAY, PayOS, Stripe…).

Hệ thống xác thực người dùng (OAuth, xác minh email/số điện thoại…).

API gửi thông báo từ các nền tảng khác.

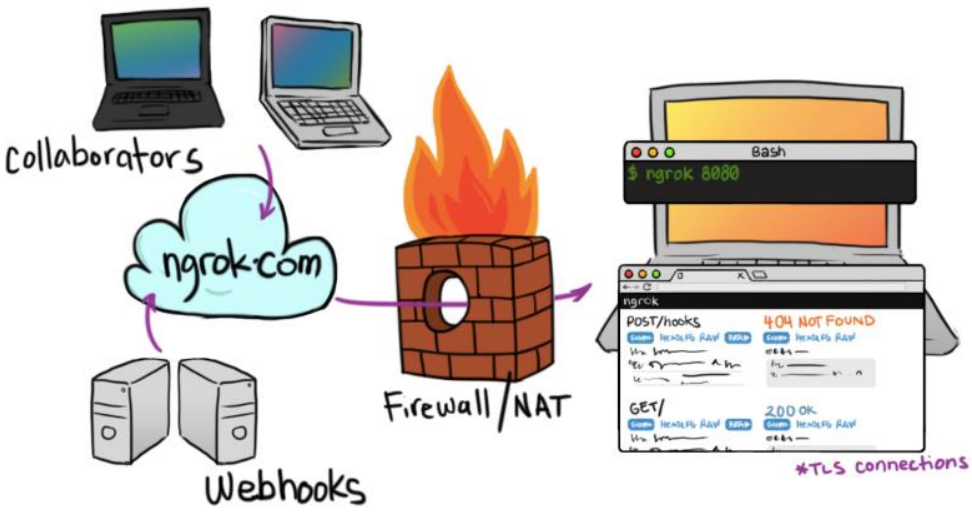
Tích hợp tính năng chatbot và phản hồi thời gian thực: Cho phép các ứng dụng AI hoặc hệ thống phản hồi truy cập trực tiếp vào server local thông qua một địa chỉ công khai.

Hỗ trợ demo và thuyết trình sản phẩm: Tiện lợi khi cần trình diễn sản phẩm online với đội nhóm hoặc nhà đầu tư.

Truy cập máy chủ nội bộ từ xa: Có thể dùng để điều khiển hoặc kiểm tra ứng dụng trong mạng nội bộ mà không cần cấu hình router hay port forwarding.

2.7.3. Nguyên lý hoạt động

Khởi tạo một kết nối bảo mật từ máy tính của lập trình viên đến máy chủ của Ngrok. Cung cấp một đường dẫn công khai (thường có dạng https://abc.ngrok-free.app) để người dùng Internet có thể truy cập. Khi có người truy cập vào đường dẫn đó, Ngrok sẽ tự động chuyển tiếp (forward) yêu cầu đến cổng đang mở trên máy cục bộ (ví dụ: localhost:5000).



Hình 2.7 Nguyên lý hoạt động của Ngrok

2.7.4. Ưu điểm và Nhược điểm của Ngrok

Ngrok mang lại nhiều ưu điểm đáng kể khi sử dụng trong quá trình phát triển và thử nghiệm ứng dụng. Nó hỗ trợ giao thức HTTPS, giúp tạo đường dẫn an toàn và dễ dàng tích hợp với các API yêu cầu bảo mật. Người dùng có thể theo dõi các request một cách trực quan thông qua bảng điều khiển tại địa chỉ http://127.0.0.1:4040, cho phép xem chi tiết các yêu cầu gửi đến máy local. Ngoài ra, Ngrok hỗ trợ nhiều giao thức như HTTP, TCP, WebSocket và TLS, đồng thời cho phép tùy chỉnh cấu hình nâng cao như chọn subdomain, bảo mật bằng token và tạo nhiều tunnel cùng lúc ở phiên bản trả phí. Tuy vậy, công cụ này cũng có một số nhược điểm, đặc biệt trong phiên bản miễn phí

các đường dẫn được tạo ra chỉ mang tính tạm thời và sẽ thay đổi sau mỗi lần chạy số lượng tunnel và tốc độ kết nối cũng bị giới hạn. Để sử dụng tên miền cố định hoặc có nhiều tính năng nâng cao hơn, người dùng cần nâng cấp lên phiên bản trả phí.

2.8. React

2.8.1. Khái niệm

React, là một thư viện JavaScript mã nguồn mở do Facebook phát triển và duy trì. Được ra mắt lần đầu vào năm 2013, React nhanh chóng trở thành một trong những công nghệ hàng đầu trong lĩnh vực phát triển giao diện người dùng (UI). React tập trung chủ yếu vào việc xây dựng giao diện ứng dụng một cách hiệu quả và dễ bảo trì, chủ yếu thông qua cách tiếp cận dựa trên các component những khối xây dựng cơ bản của giao diện người dùng [12].

2.8.2. Kiến trúc Component

Giao diện trong React được chia nhỏ thành các component (thành phần), có thể là một nút bấm đơn giản hoặc một trang hoàn chỉnh. Các component có thể tái sử dụng và được viết bằng JavaScript kết hợp với JSX, cho phép viết HTML trong mã JavaScript[1].

2.8.3. Virtual DOM

Virtual DOM là một trong những công nghệ quan trọng mà React sử dụng để cải thiện hiệu suất của ứng dụng. DOM (Document Object Model) là một cây cấu trúc của các thành phần trong trang web. Khi DOM thay đổi, trình duyệt cần phải cập nhật lại giao diện, và điều này thường tốn nhiều thời gian và gây giảm hiệu suất khi thao tác trên các trang phức tạp.

React sử dụng Virtual DOM - một bản sao của DOM thật - để giảm thiểu các thao tác với DOM. Khi trạng thái của ứng dụng thay đổi, React sẽ thực hiện cập nhật trên Virtual DOM trước, sau đó tính toán sự khác biệt (diffing) giữa DOM thật và Virtual DOM, và cuối cùng chỉ cập nhật những phần cần thiết của DOM thật, giúp cải thiện đáng kể hiệu suất.

2.8.4. JSX (JavaScript XML)

JSX là một cú pháp mở rộng cho JavaScript, cho phép viết mã trông giống như HTML ngay trong JavaScript. JSX không phải là một yêu cầu bắt buộc để sử dụng React nhưng nó giúp viết các thành phần trực quan và dễ hiểu hơn.

JSX giúp các nhà phát triển dễ dàng định nghĩa và tổ chức giao diện, vì nó kết hợp sức mạnh của JavaScript và tính dễ hiểu của HTML. Khi được biên dịch, JSX sẽ trở thành các lệnh gọi hàm React.createElement(), tạo ra cấu trúc Virtual DOM.

2.8.5. State và Props (Trạng thái và Thuộc tính)

State và props là hai khái niệm quan trọng trong React để quản lý dữ liệu bên trong và bên ngoài của component.

Props: Props (viết tắt của properties) là những thuộc tính mà một component nhận từ component cha. Props không thể thay đổi, có tính chất giống như dữ liệu "read-only". Chúng được sử dụng để truyền dữ liệu từ component cha xuống component con.

State: State là dữ liệu bên trong của một component và có thể thay đổi. State giúp component phản hồi với các tương tác của người dùng (chẳng hạn như nhấp chuột, nhập dữ liệu) bằng cách cập nhật lại giao diện khi có thay đổi trong state. State có thể được quản lý thông qua hàm useState() (trong Functional Components) hoặc thông qua this.state (trong Class Components).

2.8.6. React Hooks

React Hooks là tính năng mới được giới thiệu từ phiên bản 16.8, giúp các Functional Components có thể sử dụng state và các tính năng khác của Class Components mà không cần phải chuyển sang class.

Một số hooks phổ biến:

useState(): Dùng để quản lý state trong component.

useEffect(): Dùng để thay thế cho các lifecycle methods như componentDidMount, componentDidUpdate, và componentWillUnmount. useEffect() cho phép thực hiện các tác vụ phụ như gọi API, thiết lập bộ đếm thời gian...

useContext(): Dùng để chia sẻ state giữa các component mà không cần phải truyền props xuống qua từng cấp.

2.8.7. Lifecycle Methods (Các phương thức vòng đời)

React cung cấp các phương thức như componentDidMount, componentDidUpdate, componentWillUnmount để quản lý vòng đời component, nhưng hiện nay đa số dùng hook useEffect() thay thế trong function component.

2.8.8. Routing trong React

React Router là thư viện phổ biến được sử dụng để quản lý điều hướng trong ứng dụng React. Với React Router, bạn có thể định nghĩa các "route" khác nhau trong ứng dụng của mình và điều hướng giữa các trang một cách dễ dàng mà không cần tải lại trang.

2.8.9. State Management (Quản lý trạng thái)

Khi ứng dụng trở nên phức tạp hơn, việc quản lý state trở nên quan trọng và khó khăn hơn. Các công cụ quản lý trạng thái như Redux hay Context API của React được sử dụng để quản lý và chia sẻ state trên toàn bộ ứng dụng[1].

Redux: Là thư viện quản lý state mạnh mẽ với mô hình một "store" duy nhất cho toàn bộ ứng dụng, sử dụng các khái niệm như actions, reducers, và store.

Context API: Được sử dụng để quản lý và chia sẻ state giữa nhiều component mà không cần phải truyền props.

2.8.10. Ưu và nhược điểm của React

React cho phép xây dựng giao diện từ các component độc lập, giúp dễ dàng tái sử dụng, mở rộng và bảo trì. Với sự hỗ trợ của hooks, việc quản lý trạng thái và xử lý logic trong function component trở nên linh hoạt hơn mà không cần dùng đến class. Bên cạnh đó, React sở hữu cộng đồng phát triển lớn và tài liệu phong phú, kèm theo nhiều thư viện và công cụ hỗ trợ, giúp tăng tốc quá trình phát triển. React cũng tương thích tốt với mô hình ứng dụng một trang (SPA), rất phù hợp với các ứng dụng web hiện đại. Tuy nhiên, React cũng có một số nhược điểm như việc học ban đầu có thể gây khó khăn cho người mới do cần làm quen với JSX, Virtual DOM và khái niệm hooks. Ngoài ra, do React chỉ tập trung vào phần view, lập trình viên thường phải tích hợp thêm các thư viện phụ trợ như Redux hoặc React Router để xây dựng một ứng dụng hoàn chỉnh. Việc thiếu

một hướng dẫn chính thức toàn diện và có quá nhiều cách tiếp cận khác nhau cũng có thể khiến người mới dễ bị rối [13].

2.9. SQL Server

2.9.1. Khái niệm

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System – RDBMS) được phát triển bởi tập đoàn công nghệ hàng đầu thế giới – Microsoft. Được giới thiệu lần đầu tiên vào năm 1989, SQL Server đã nhanh chóng trở thành một trong những nền tảng lưu trữ và quản lý dữ liệu phổ biến nhất, đặc biệt trong các hệ thống doanh nghiệp, và vẫn duy trì được vị thế đáng kể cho đến ngày nay.

Trái với sự hiểu nhầm phổ biến rằng "server" luôn đồng nghĩa với thiết bị phần cứng, SQL Server về bản chất là một phần mềm. Mặc dù nó thường được cài đặt và vận hành trên các máy chủ vật lý (server), chức năng cốt lõi của SQL Server là phần mềm hệ quản trị dữ liệu. Chính vì vậy, khái niệm "server" trong tên gọi của nó phản ánh vai trò như một máy chủ dữ liệu – nơi tiếp nhận, xử lý và cung cấp dữ liệu cho các ứng dụng client – chứ không chỉ đơn thuần đề cập đến phần cứng.

Về mặt chức năng, SQL Server đảm nhiệm vai trò lưu trữ, truy vấn và quản lý dữ liệu. Các ứng dụng phần mềm, thông qua cơ chế giao tiếp client-server, gửi yêu cầu truy xuất dữ liệu đến SQL Server. Khi đó, SQL Server thực hiện các phép truy vấn, xử lý dữ liệu theo logic đã định và trả về kết quả phù hợp. Điều này cho phép các hệ thống phần mềm như ứng dụng quản lý, website thương mại điện tử, hoặc hệ thống kế toán vận hành hiệu quả trên một nguồn dữ liệu tập trung.

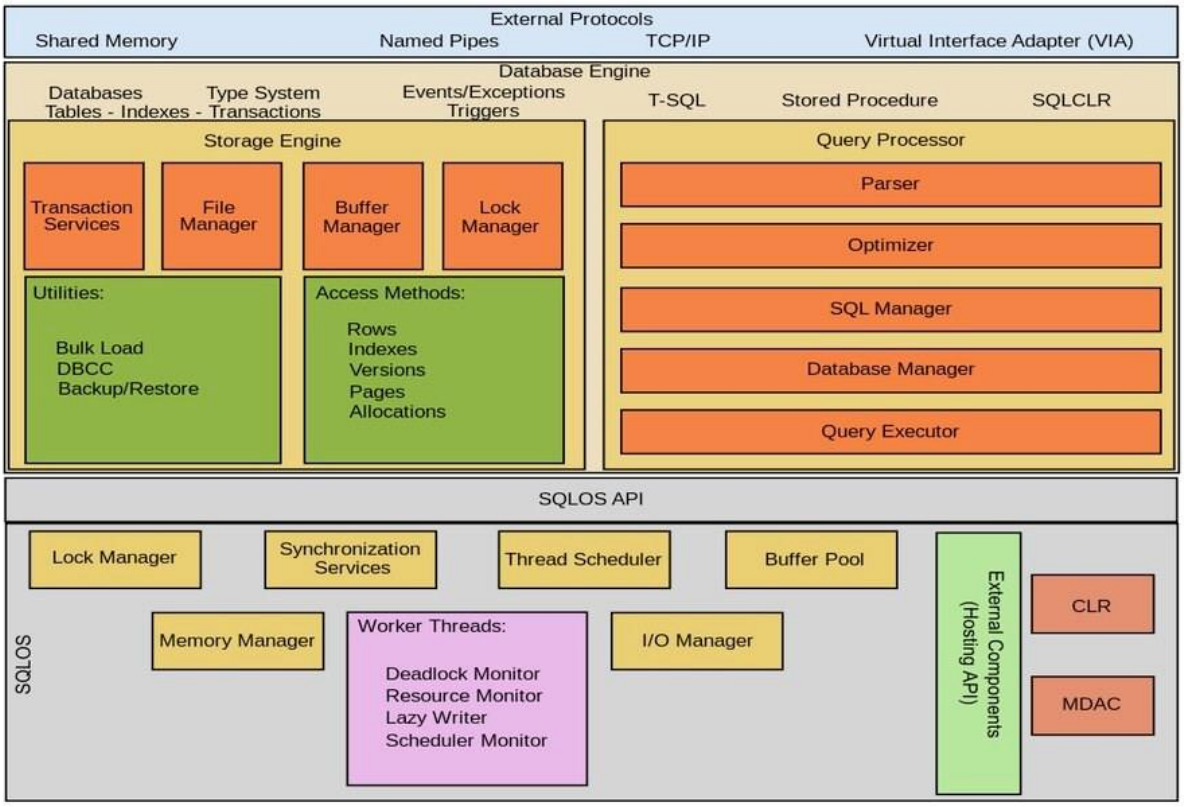
Một đặc điểm kỹ thuật quan trọng của SQL Server là nó sử dụng ngôn ngữ SQL (Structured Query Language) để thao tác với dữ liệu. Cụ thể hơn, Microsoft đã mở rộng SQL chuẩn thành một ngôn ngữ riêng có tên là Transact-SQL (T-SQL), bổ sung thêm các chức năng lập trình như biến, vòng lặp, điều kiện rẽ nhánh và xử lý lỗi. T-SQL chính là phương tiện giao tiếp giữa client và server trong hệ thống sử dụng SQL Server, cho phép các lập trình viên thực hiện từ các truy vấn đơn giản đến các thao tác phức tạp trên cơ sở dữ liệu.

2.9.2. Cấu trúc

SQLOS: là viết tắt của hệ điều hành SQL server. Đây là tầng cuối cùng trong kiến trúc tổng thể của SQL server. Tại đây sẽ chịu trách nhiệm xử lý các nhiệm vụ như quản lý bộ nhớ, lên lịch nhiệm vụ, khoá dữ liệu nhằm tránh các xung đột ngoài ý muốn có thể xảy ra mỗi khi thực hiện các thao tác cập nhật.

Database engine: đây là một công cụ có chức năng quản lý việc lưu trữ, xử lý và bảo mật dữ liệu. Trong đây sẽ bao gồm rất nhiều các công cụ khác nhau như một công cụ lưu trữ quản lý các tệp, bảng, trang, chỉ mục, bộ đệm dữ liệu và giao dịch cơ sở dữ liệu.

External protocol: đây là các giao thức được sử dụng để giao tiếp với Database engine. Nó bao gồm TCP/IP hay VIA (Virtual Interface Adapter),.... [14].



Hình 2.8 Cấu trúc của SQL Server

2.10. EmailJS

2.10.1. Khái niệm EmailJS

EmailJS là một dịch vụ cho phép gửi email trực tiếp từ phía trình duyệt (client-side) mà không cần xây dựng backend riêng. Công cụ này đặc biệt hữu ích trong các

ứng dụng web như form liên hệ, gửi phản hồi hoặc xác nhận đơn hàng, nơi việc gửi email cần diễn ra nhanh chóng và đơn giản.

Hình 2.9 Nguyên lý hoạt động EmailJS

EmailJS hoạt động dựa trên việc kết nối với các dịch vụ email như Gmail, Outlook hoặc Mailgun thông qua một tài khoản trung gian. Người dùng chỉ cần cài đặt thư viện JavaScript của EmailJS, sau đó cấu hình mẫu email, ID dịch vụ và ID mẫu. Khi người dùng điền form và nhấn gửi, thư viện sẽ gửi yêu cầu tới máy chủ của EmailJS để gửi email dựa trên dữ liệu đã định trước [15].

2.10.2. Ứng dụng của EmailJS

EmailJS thường được sử dụng trong các chức năng sau:

Form liên hệ (Contact Form): Khi người dùng điền thông tin liên hệ (họ tên, email, nội dung), hệ thống sử dụng EmailJS để gửi email trực tiếp đến địa chỉ quản trị viên. Điều này giúp phản hồi khách hàng nhanh chóng mà không cần backend xử lý riêng.

Gửi thông báo đơn hàng: Trong các website thương mại điện tử, EmailJS có thể dùng để gửi thông báo xác nhận đơn hàng hoặc phản hồi đặt hàng cho khách hàng ngay sau khi họ hoàn tất mua sắm.

Gửi phản hồi hoặc góp ý: Tính năng gửi góp ý từ người dùng có thể kết hợp với EmailJS để chuyển nội dung về email quản trị mà không cần xây dựng API hoặc hệ thống xử lý phức tạp.

2.10.3. Ưu điểm và nhược điểm EmailJS

EmailJS dễ tích hợp, không cần máy chủ riêng, hỗ trợ nhiều nền tảng (HTML, React, Vue,...). Tuy nhiên, dịch vụ này có giới hạn gửi email miễn phí mỗi tháng và phụ thuộc vào bảo mật API Key, nên cần cấu hình cẩn thận để tránh bị lạm dụng.

Phiên bản miễn phí giới hạn số lượng email mỗi tháng. Dữ liệu gửi đi vẫn có thể bị lộ nếu API Key không được bảo vệ cẩn thận. Phụ thuộc vào dịch vụ bên thứ ba, không phù hợp với hệ thống yêu cầu bảo mật cao hoặc gửi số lượng lớn.

2.11. Swagger và Postman

Trong quá trình phát triển các hệ thống phần mềm hiện đại, đặc biệt là các ứng dụng có kiến trúc API-first như RESTful Web API, việc tài liệu hóa và kiểm thử các giao tiếp giữa các thành phần trong hệ thống trở nên cần thiết. Hai công cụ phổ biến được sử dụng nhằm hỗ trợ các hoạt động này chính là Swagger và Postman. Mỗi công cụ mang một vai trò khác nhau trong quy trình phát triển và kiểm thử phần mềm, đồng thời giúp chuẩn hóa và tự động hóa nhiều thao tác vốn phức tạp và dễ sai sót khi thực hiện thủ công.

Swagger, hiện nay thuộc hệ sinh thái OpenAPI Specification (OAS), là một bộ công cụ mã nguồn mở cho phép mô tả, tiêu chuẩn hóa và trực quan hóa các API. Khi xây dựng một Web API, việc định nghĩa rõ các endpoint, phương thức HTTP (GET, POST, PUT, DELETE...), kiểu dữ liệu vào - ra, mã trạng thái trả về và các mô tả liên quan là rất quan trọng để đảm bảo các bên liên quan (developers, testers, frontend, đối tác tích hợp...) có thể hiểu và sử dụng đúng API. Swagger giúp tự động hóa quá trình này bằng cách cho phép khai báo cấu trúc API ngay trong mã nguồn (thông qua các annotation hoặc YAML/JSON), từ đó sinh ra tài liệu trực quan và có thể tương tác trực tiếp qua giao diện người dùng. Thêm vào đó, Swagger còn hỗ trợ tạo mã client từ định nghĩa API để tích hợp dễ dàng hơn vào ứng dụng thực tế, đồng thời đóng vai trò như một công cụ mô phỏng API (mocking) trong giai đoạn phát triển frontend.

Bên cạnh đó, Postman là một công cụ mạnh mẽ dùng để gửi các yêu cầu HTTP đến API và quan sát phản hồi nhằm phục vụ việc kiểm thử và gỡ lỗi. Với giao diện trực quan và thân thiện, Postman cho phép người dùng mô phỏng đầy đủ các yêu cầu đến API, bao gồm cài đặt headers, body, tham số truy vấn (query params), authentication (API key, OAuth, JWT,...) và các chuỗi kịch bản kiểm thử tự động thông qua tính năng viết test scripts bằng JavaScript. Đặc biệt, Postman hỗ trợ lưu lại các collection (tập hợp các API), phục vụ việc tổ chức kiểm thử có hệ thống và chia sẻ với các thành viên khác trong nhóm. Trong các môi trường CI/CD, Postman còn được tích hợp với Newman công cụ dòng lệnh cho phép chạy các bộ kiểm thử Postman trong pipeline tự động, từ

đó đảm bảo chất lượng dịch vụ liên tục và nhất quán trong suốt vòng đời phát triển phần mềm.

Có thể thấy rằng, Swagger và Postman đóng vai trò bổ trợ lẫn nhau. Trong khi Swagger đóng vai trò như bản thiết kế chính thức và là cầu nối giữa đội phát triển và tài liệu kỹ thuật, thì Postman là công cụ tương tác thực tế với API, giúp kiểm thử chức năng, đánh giá tính đúng đắn, và phát hiện sớm các lỗi logic hoặc dữ liệu. Việc sử dụng đồng thời cả hai công cụ mang lại sự nhất quán trong quy trình tài liệu hóa kiểm thử tích hợp, từ đó nâng cao độ tin cậy và khả năng bảo trì của hệ thống phần mềm.

Tóm lại, Swagger và Postman là hai thành phần không thể thiếu trong phát triển API hiện đại. Bằng cách cung cấp tài liệu rõ ràng, khả năng tương tác và kiểm thử linh hoạt, hai công cụ này góp phần đáng kể vào việc tăng cường hiệu suất làm việc nhóm, giảm thiểu lỗi tích hợp và nâng cao chất lượng tổng thể của sản phẩm phần mềm [16], [17].

2.12. Các công trình nghiên cứu liên quan

Troelsen và Japikse (2017), trong cuốn *Pro C# 7: With .NET and .NET Core (8th ed.)*, Apress, đã trình bày chi tiết về C# và .NET Core. Công trình cung cấp nền tảng ngôn ngữ và công nghệ để triển khai backend cho hệ thống website [3].

Nguyễn Văn Thắng (2020), với bài viết *Triển khai trong .NET Core với MediatR CQRS Pattern* đăng trên Viblo, đã trình bày giải pháp áp dụng mẫu thiết kế CQRS cùng MediatR trong phát triển ứng dụng. Đây là một hướng tiếp cận quan trọng để nâng cao khả năng mở rộng và bảo trì hệ thống [5].

Trần Văn (2022), trong bài viết *Clean Architecture là gì – Ưu nhược và cách dùng hợp lý*, 200Lab Blog, đã phân tích ưu và nhược điểm của Clean Architecture. Nghiên cứu này giúp tác giả khóa luận lựa chọn và triển khai kiến trúc phù hợp với yêu cầu thực tế [9].

Trần Toàn Phát (2022), trong luận văn *Phát triển hệ thống quảng cáo dựa trên web sử dụng ReactJS và NodeJS* – Đại học Cần Thơ, đã trình bày cách kết hợp ReactJS với NodeJS/ExpressJS và MongoDB để xây dựng website quảng cáo trực tuyến. Công trình là nền tảng tham khảo hữu ích trong phần frontend của khóa luận [19].

Cao Tiến Quang & Trần Phước Thành (2020), trong báo cáo *Tìm hiểu Spring cho backend, ReactJS cho frontend và xây dựng website bán giày minh họa* – Đại học Giao thông Vận tải TP. HCM, đã nghiên cứu mô hình xây dựng website thương mại điện tử kết hợp Spring và ReactJS. Nghiên cứu này giúp cung cấp ý tưởng về cấu trúc frontend-backend cho hệ thống khóa luận [18].

CHƯƠNG 3: HIỆN THỰC HOÁ NGHIÊN CỨU

3.1. Mô tả hệ thống

Công ty chuyên kinh doanh thời trang hiện đang hoạt động chủ yếu theo mô hình bán hàng truyền thống tại cửa hàng offline. Tuy nhiên, trước sự thay đổi nhanh chóng trong thói quen mua sắm của người tiêu dùng, đặc biệt là xu hướng mua sắm trực tuyến ngày càng phổ biến, công ty quyết định mở rộng kinh doanh bằng cách phát triển một website chuyên cung cấp các sản phẩm thời trang đa dạng và hiện đại.

Mục tiêu chính của công ty là xây dựng một nền tảng thương mại điện tử chuyên nghiệp, giúp khách hàng có thể dễ dàng tiếp cận, lựa chọn và mua sắm các sản phẩm thời trang một cách thuận tiện và nhanh chóng.

Một trong những điểm khác biệt và cũng là trọng tâm công nghệ của hệ thống là tính năng thử đồ ảo ứng dụng trí tuệ nhân tạo. Đây là giải pháp công nghệ hiện đại được tích hợp trực tiếp vào nền tảng thương mại điện tử, nhằm nâng cao trải nghiệm mua sắm trực tuyến cho khách hàng trong lĩnh vực thời trang.

Tính năng thử đồ ảo được phát triển để giải quyết một trong những hạn chế lớn nhất của việc mua sắm online đó là không thể hình dung chính xác sản phẩm mặc lên người sẽ như thế nào. Khác với trải nghiệm mua hàng truyền thống tại cửa hàng vật lý, việc mua sắm trực tuyến thường khiến khách hàng phân vân vì thiếu yếu tố trải nghiệm thực tế.

Ngoài ra, khi khách hàng gặp các vấn đề hoặc có yêu cầu hỗ trợ, bộ phận quản trị cần đảm bảo tiếp nhận và xử lý thông tin kịp thời, nhằm duy trì sự hài lòng và tạo dựng niềm tin lâu dài với khách hàng.

3.2. Các chức năng của hệ thống

Người dùng là khách hàng truy cập website để tìm kiếm, lựa chọn và mua sắm các sản phẩm thời trang. Người dùng có thể đăng ký tài khoản, đăng nhập, quản lý thông tin cá nhân và địa chỉ giao hàng. Họ có thể xem thông tin sản phẩm chi tiết, sử dụng tính năng thử đồ ảo dựa trên công nghệ AI để có trải nghiệm chọn lựa sản phẩm phù hợp hơn. Ngoài ra, người dùng có thể đặt hàng, sử dụng mã giảm giá khi thanh toán và theo dõi trạng thái đơn hàng của mình trên hệ thống.

Bộ phận quản trị có quyền cao nhất trong hệ thống, chịu trách nhiệm quản lý toàn bộ website. Bộ phận này đảm nhiệm việc thêm mới, cập nhật và xóa bỏ thông tin sản phẩm như tên, mô tả, giá bán, giá khuyến mãi, hình ảnh, số lượng tồn kho và các thuộc tính liên quan để đảm bảo thông tin trên website luôn chính xác và cập nhật. Khi có sản phẩm mới hoặc chương trình ưu đãi, bộ phận quản trị sẽ cập nhật kịp thời để khách hàng dễ dàng tiếp cận.

Bên cạnh đó, quản trị viên có nhiệm vụ duyệt đơn hàng của khách hàng, đồng thời xử lý việc in ấn đơn hàng để hỗ trợ công tác vận chuyển diễn ra nhanh chóng và hiệu quả. Khi khách hàng có các yêu cầu hỗ trợ hoặc khiếu nại, bộ phận quản trị cũng phải tiếp nhận và xử lý kịp thời nhằm đảm bảo sự hài lòng và tin tưởng của khách hàng đối với hệ thống.

Quy trình làm việc:

Khách hàng truy cập vào trang chủ của website bằng cách nhập trực tiếp URL hoặc thông qua các công cụ tìm kiếm trên Internet.

Khách hàng có thể duyệt sản phẩm theo các danh mục được phân loại sẵn hoặc sử dụng chức năng tìm kiếm để nhanh chóng tìm thấy danh sách sản phẩm phù hợp với nhu cầu. Khi tìm thấy sản phẩm quan tâm, khách hàng nhấp vào để xem chi tiết bao gồm mô tả sản phẩm, hình ảnh chính và các hình ảnh liên quan, cùng với giá bán của sản phẩm.

Tại trang chi tiết sản phẩm, khách hàng có thể sử dụng tính năng thử đồ ảo bằng công nghệ AI. Khách hàng tải lên ảnh của mình để xem sản phẩm được mặc trực tiếp trên hình ảnh, từ đó dễ dàng hình dung sự phù hợp về kích thước, kiểu dáng và phong cách trước khi quyết định mua hàng.

Nếu cần hỗ trợ hoặc tư vấn thêm, khách hàng có thể chọn trò chuyện với Chatbot AI tư vấn bán hàng tự động hoặc liên hệ trực tiếp với nhân viên tư vấn qua hệ thống chat tích hợp trên website.

Sau khi quyết định mua, khách hàng chọn số lượng sản phẩm mong muốn và thêm vào giỏ hàng của mình.

Tiếp theo, khách hàng chuyển sang trang thanh toán, tại đây sẽ điền đầy đủ thông tin giao hàng và lựa chọn phương thức thanh toán phù hợp. Khách hàng có thể xem lại

toàn bộ thông tin đơn hàng, bao gồm giá cả, số lượng sản phẩm và các chi tiết liên quan trước khi xác nhận đặt hàng. Sau đó, khách hàng tiến hành thanh toán theo phương thức đã chọn.

Khi đơn hàng được đặt thành công, hệ thống sẽ hiển thị đơn hàng này trong trang quản trị dành cho người quản lý.

Người quản trị sẽ kiểm tra, xét duyệt thông tin đơn hàng, đảm bảo tính hợp lệ và chính xác trước khi tiến hành các bước tiếp theo.

Sau khi duyệt đơn hàng, người quản trị sẽ thực hiện in ấn đơn hàng để phục vụ công tác vận chuyển và giao nhận hàng hóa được nhanh chóng, chính xác.

Yêu cầu chức năng:

Trang chủ: thể hiện tổng quan về toàn bộ nội dung của website, bao gồm thông tin giới thiệu về công ty, danh mục sản phẩm, các sản phẩm mới nhất, sản phẩm nổi bật và các mặt hàng đang được giảm giá. Đây là nơi thu hút khách hàng ngay từ lần đầu truy cập và giúp họ dễ dàng tiếp cận các sản phẩm và thông tin quan trọng.

Thông tin sản phẩm: cung cấp đầy đủ các thông tin quan trọng như giá gốc, giá khuyến mãi, thông số kỹ thuật, hình ảnh chính và hình ảnh chi tiết của sản phẩm. Ngoài ra, khách hàng còn được cung cấp thông tin về quy trình vận chuyển và chính sách trả hàng. Tại đây, khách hàng có thể để lại đánh giá về sản phẩm đã mua.

Điểm đặc biệt của hệ thống là tích hợp tính năng thử đồ ảo bằng công nghệ AI, cho phép khách hàng sử dụng ảnh cá nhân để trải nghiệm thử sản phẩm trực tuyến. Tính năng này giúp khách hàng hình dung rõ hơn về sự phù hợp của sản phẩm về kích thước, kiểu dáng và phong cách trước khi quyết định mua hàng.

Giỏ hàng: Giúp khách hàng quản lý các sản phẩm đã chọn, số lượng và có thể thực hiện mua nhiều sản phẩm cùng lúc một cách thuận tiện và dễ dàng.

Thông tin đơn hàng: Cho phép khách hàng quản lý và xác nhận lại các thông tin sản phẩm, số lượng, địa chỉ giao hàng và các chi tiết liên quan trước khi hoàn tất đơn hàng. Đồng thời, khách hàng cũng có thể theo dõi quá trình xử lý đơn hàng và nhận phản hồi từ người quản trị.

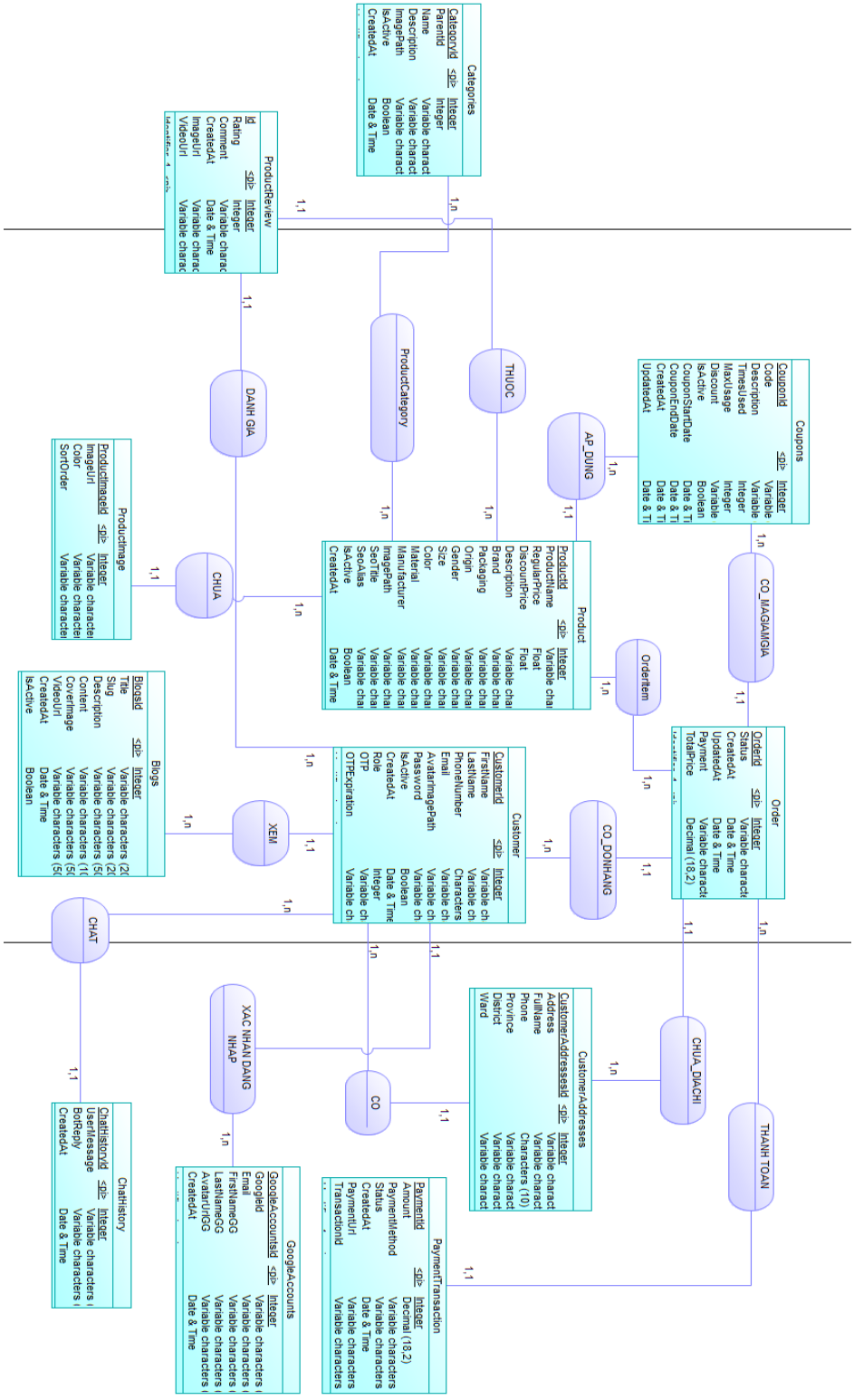
Tìm kiếm: Cung cấp chức năng tìm kiếm sản phẩm theo các tiêu chí như tên sản phẩm, danh mục nhằm giúp khách hàng nhanh chóng tìm được mặt hàng mong muốn.

Chat: Hệ thống tích hợp hai hình thức hỗ trợ khách hàng gồm Chatbot AI tự động thông minh giúp trả lời các câu hỏi phổ biến và tư vấn cơ bản để giải đáp các thắc mắc cụ thể về sản phẩm, cửa hàng, giá cả hay thời gian làm việc.

Trang quản trị: Giúp người quản trị thực hiện các công việc quản lý sản phẩm, thông tin khách hàng, đơn hàng một cách nhanh chóng và hiệu quả. Người quản trị có quản lý số lượng tồn kho, duyệt đơn hàng và gửi phản hồi về cho khách hàng. Bên cạnh đó, hệ thống còn cung cấp các báo cáo tổng quan như tổng số đơn hàng, tổng số khách hàng, doanh thu giúp người quản trị nắm bắt tình hình kinh doanh toàn diện.

3.3. Thiết kế dữ liệu

3.3.1. Mô hình dữ liệu mức quan niệm



Hình 3.1 Mô hình dữ liệu mức quan niệm

3.4. Mô tả thực thể

Cơ sở dữ liệu được thiết kế như sau:

Bảng **Categories:**

Bảng 3.1 Bảng Categories

Cột	Kiểu Dữ Liệu	Mô Tả
CategoryId	int	Mã danh mục (khóa chính)
ParentId	int	Mã danh mục cha
Name	nvarchar	Tên danh mục
Description	nvarchar	Mô tả danh mục
ImagePath	nvarchar	Đường dẫn ảnh
IsActive	bit	Trạng thái hoạt động
CreatedAt	datetime	Ngày tạo danh mục

Bảng **Products:**

Bảng 3.2 Bảng Products

Cột	Kiểu Dữ Liệu	Mô Tả
ProductId	int	Mã sản phẩm, khóa chính.
ProductName	nvarchar	Tên sản phẩm
RegularPrice	float	Giá niêm yết của sản phẩm
DiscountPrice	float	Giá khuyến mãi của sản phẩm
Description	nvarchar	Mô tả chi tiết về sản phẩm
Brand	nvarchar	Thương hiệu sản phẩm

Packaging	nvarchar	Loại bao bì hoặc cách đóng gói sản phẩm
Origin	nvarchar	Nơi sản xuất sản phẩm
Gender	nvarchar	Giới tính
Size	varchar	Kích cỡ sản phẩm (nếu áp dụng)
Color	nvarchar	Màu sắc của sản phẩm
Material	nvarchar	Chất liệu sản phẩm
Manufacturer	nvarchar	Nhà sản xuất sản phẩm
ImagePath	nvarchar	Đường dẫn đến hình ảnh sản phẩm
SeoTitle	nvarchar	Tiêu đề SEO của sản phẩm
SeoAlias	nvarchar	Tên gọi ngắn gọn của sản phẩm
IsActive	bit	Trạng thái hoạt động của sản phẩm
CreatedAt	datetime	Thời gian tạo sản phẩm

Bảng **Customer:**

Bảng 3.3 Bảng Customer

Cột	**Kiểu Dữ Liệu**	**Mô Tả**
CustomerId	int	Mã khách hàng (khóa chính)
FirstName	nvarchar	Tên khách hàng
LastName	nvarchar	Họ khách hàng
PhoneNumber	varchar	Số điện thoại của khách hàng
Email	nvarchar	Email của khách hàng (unique)

AvatarImagePath	nvarchar	Đường dẫn đến ảnh đại diện của khách hàng
Password	nvarchar	Mật khẩu của khách hàng
CreatedAt	datetime	Thời gian tạo tài khoản
Role	int	Vai trò của khách hàng (0: User, 1: Admin)
OTP	varchar	Mã OTP cho xác thực (nếu có)
OTPExpiration	datetime	Thời gian hết hạn của OTP

Bảng **CustomerAddresses:**

Bảng 3.4 Bảng CustomerAddresses

Cột	**Kiểu Dữ Liệu**	**Mô Tả**
CustomerAddressesId	int	Mã địa chỉ của khách hàng (khóa chính)
CustomerId	int	Mã khách hàng (khóa ngoại liên kết với bảng Customers)
Address	nvarchar	Địa chỉ giao hàng của khách hàng
FullName	nvarchar	Tên đầy đủ của người nhận
Phone	varchar	Số điện thoại của người nhận
Province	nvarchar	Tỉnh hoặc thành phố
District	nvarchar	Quận hoặc huyện

Bảng **Coupon:**

Bảng 3.5 Bảng Coupon

Cột	Kiểu Dữ Liệu	Mô Tả
CouponId	int	Mã giảm giá (khóa chính)
Code	nvarchar	Mã code giảm giá
Description	nvarchar	Mô tả về mã giảm giá
TimesUsed	int	Số lần đã sử dụng mã giảm giá
MaxUsage	int	Số lần sử dụng tối đa của mã giảm giá
Discount	nvarchar	Mức giảm giá (ví dụ: "10%", "100,000 VND")
IsActive	bit	Trạng thái mã giảm giá
CouponStartDate	datetime	Ngày bắt đầu hiệu lực của mã giảm giá
CouponEndDate	datetime	Ngày hết hạn của mã giảm giá
CreatedAt	datetime	Thời gian tạo mã giảm giá
UpdatedAt	datetime	Thời gian cập nhật mã giảm giá

Bảng **Order:**

Bảng 3.6 Bảng Order

Cột	Kiểu Dữ Liệu	Mô Tả
OrderId	int	Mã đơn hàng (khóa chính)
CustomerId	int	Mã khách hàng (khóa ngoại liên kết với bảng Customers)

Status	int	Trạng thái đơn hàng
CreatedAt	datetime	Thời gian tạo đơn hàng
UpdatedAt	datetime	Thời gian cập nhật đơn hàng
CustomerAddressId	int	Mã địa chỉ giao hàng (khóa ngoại liên kết với bảng CustomerAddresses)
CouponId	int	Mã giảm giá (khóa ngoại liên kết với bảng Coupons)
Payment	nvarchar	Phương thức thanh toán (e.g., tiền mặt, thẻ tín dụng, ví điện tử)
TotalPrice	decimal	Tổng giá trị đơn hàng

Bảng **OrderItem:**

Bảng 3.7 Bảng OrderItem

Cột	**Kiểu Dữ Liệu**	**Mô Tả**
OrderItemId	int	Mã chi tiết đơn hàng (khóa chính)
OrderId	int	Mã đơn hàng (khóa ngoại liên kết với bảng Order)
ProductId	int	Mã sản phẩm (khóa ngoại liên kết với bảng Products)
Quantity	int	Số lượng sản phẩm trong đơn hàng
TotalPrice	decimal	Tổng giá trị sản phẩm trong đơn hàng (Số lượng x Giá)

Bảng **ProductCategory:**

Bảng 3.8 Bảng ProductCategory

Cột	Kiểu Dữ Liệu	Mô Tả
ProductId	int	Mã sản phẩm (khóa ngoại liên kết với bảng Products)
CategoryId	int	Mã danh mục (khóa ngoại liên kết với bảng Categories)

Bảng **Blogs:**

Bảng 3.9 Bảng Blogs

Cột	Kiểu Dữ Liệu	Mô Tả
Id	int	Mã sản phẩm (khóa ngoại liên kết với bảng Products).
Title	nvarchar	Tiêu đề của bài viết blog.
Slug	nvarchar	Đường dẫn thân thiện với URL.
Description	nvarchar	Mô tả ngắn nội dung của bài viết, dùng để hiển thị phần tóm tắt.
Content	nvarchar	Nội dung chi tiết của bài viết, có thể chứa HTML hoặc markdown.
CoverImage	nvarchar	URL hoặc đường dẫn đến ảnh đại diện của bài viết.
VideoUrl	nvarchar	URL video (nếu bài viết có video minh họa, thường là YouTube, Vimeo).
CreatedAt	Datetime	Thời điểm bài viết được tạo.
IsActive	bit	Trạng thái hoạt động (1: hiển thị, 0: ẩn).

Bảng **ProductImage:**

Bảng 3.10 Bảng ProductImage

Cột	Kiểu Dữ Liệu	Mô Tả
ProductImageId	int	Khóa chính, định danh duy nhất cho mỗi ảnh phụ.
ProductId	int	Khóa ngoại tham chiếu đến sản phẩm.
ImageUrl	nvarchar	Đường dẫn ảnh phụ.
Color	nvarchar	Màu ảnh phụ sản phẩm.
SortOrder	int	Thứ tự hiển thị ảnh phụ.

Bảng **ProductReview:**

Bảng 3.11 Bảng ProductReview

Cột	Kiểu Dữ Liệu	Mô Tả
Id	int	Khóa chính, định danh duy nhất cho mỗi đánh giá.
ProductId	int	Khóa ngoại tham chiếu đến sản phẩm được đánh giá.
UserId	int	Khóa ngoại tham chiếu đến người dùng thực hiện đánh giá.
Rating	int	Số sao đánh giá (từ 1 đến 5).
Comment	nvarchar	Nội dung nhận xét của người dùng.
CreatedAt	Datetime	Ngày giờ đánh giá được tạo.
ImageUrl	nvarchar	URL hình ảnh minh họa (nếu người dùng tải lên).
VideoUrl	nvarchar	URL video minh họa (nếu người dùng tải lên).

Bảng **ChatHistory:**

Bảng 3.12 Bảng ChatHistory

Cột	Kiểu Dữ Liệu	Mô Tả
Id	int	Khóa chính, định danh duy nhất cho mỗi cuộc hội thoại.
UserMessage	int	Mã danh mục (khóa ngoại liên kết với bảng Categories)
BotReply	nvarchar	Phản hồi từ chatbot tương ứng với tin nhắn người dùng.
CreatedAt	datetime	Thời gian ghi nhận cuộc hội thoại.

Bảng **PaymentTransaction:**

Bảng 3.13 Bảng PaymentTransaction

Cột	Kiểu Dữ Liệu	Mô Tả
Id	int	Khóa chính, định danh duy nhất cho giao dịch.
OrderId	int	Khóa ngoại tham chiếu đến đơn hàng tương ứng.
Amount	decimal	Số tiền giao dịch.
PaymentMethod	nvarchar	Phương thức thanh toán
Status	nvarchar	Trạng thái giao dịch
CreatedAt	Datetime	Ngày giờ tạo giao dịch
PaymentUrl	nvarchar	URL để chuyển hướng đến cổng thanh toán

TransactionId	nvarchar	Mã giao dịch từ cổng thanh toán (mã xác nhận duy nhất từ bên thứ ba).

Bảng **GoogleAccounts:**

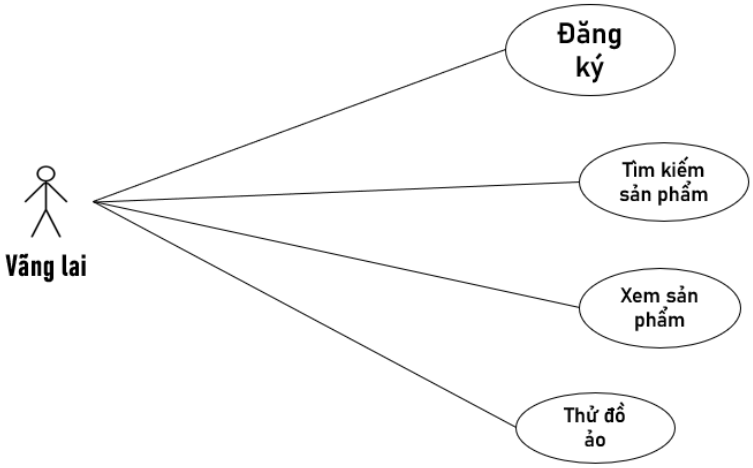
Bảng 3.14 Bảng GoogleAccounts

Cột	Kiểu Dữ Liệu	Mô Tả
Id	int	Khóa chính tự tăng, định danh nội bộ của người dùng.
GoogleId	nvarchar	ID định danh người dùng do Google cấp (sub trong token).
Email	nvarchar	Địa chỉ email của người dùng (lấy từ Google).
FirstName	nvarchar	Tên người dùng
LastName	nvarchar	Họ người dùng
AvatarUrl	nvarchar	URL ảnh đại diện Google
CreatedAt	Datetime	Thời điểm tài khoản được tạo lần đầu trong hệ thống.

3.5. Thiết kế xử lý

3.5.1. Biểu đồ Use Case tác nhân vãng lai

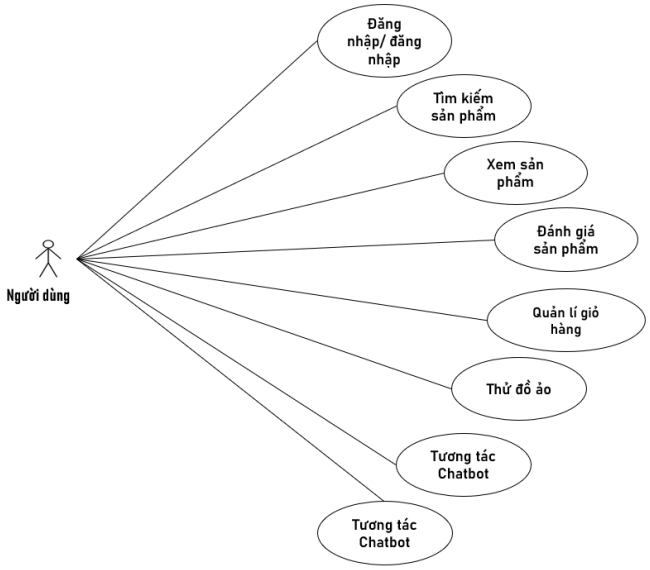
Người dùng vãng lai có thể đăng ký tài khoản mới để trở thành khách hàng chính thức, họ cũng có thể tìm kiếm sản phẩm, xem thông tin chi tiết về các sản phẩm có trên hệ thống mà không cần phải đăng nhập. Ngoài ra, vãng lai còn có thể sử dụng tính năng thử đồ ảo để trải nghiệm sản phẩm một cách trực quan hơn. Những chức năng này giúp tạo điều kiện cho người dùng tiếp cận, trải nghiệm và cân nhắc việc đăng ký tài khoản, đồng thời nâng cao trải nghiệm người dùng ngay từ lần truy cập đầu tiên.



Hình 3.2 Biểu đồ Use Case tác nhân vãng lai

3.5.2. Biểu đồ Use Case tác nhân khách hàng

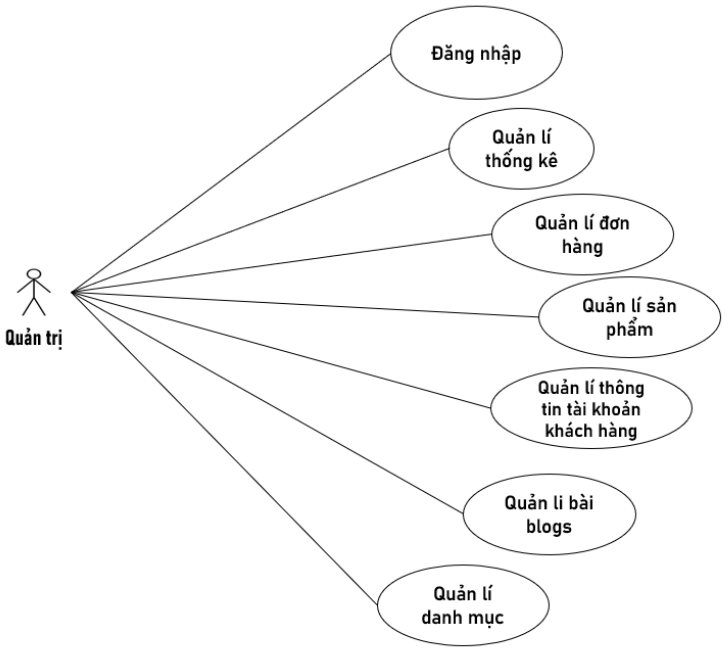
Khách hàng: Là trung tâm của mọi hoạt động mua sắm, khách hàng được trang bị đầy đủ các tính năng để khám phá và lựa chọn sản phẩm. Khách hàng có thể dễ dàng duyệt qua danh sách sản phẩm đa dạng, tìm hiểu chi tiết về từng sản phẩm và sử dụng công cụ thử đồ ảo, công cụ tìm kiếm để nhanh chóng tìm thấy sản phẩm mong muốn. Khi đã chọn được sản phẩm ưng ý, khách hàng có thể thêm vào giỏ hàng và tiến hành đặt hàng một cách thuận tiện. Để trải nghiệm các tính năng đặt hàng và đánh giá sản phẩm, khách hàng cần đăng ký tài khoản hoặc đăng nhập nếu đã có tài khoản. Sau khi mua và sử dụng sản phẩm, khách hàng có thể chia sẻ trải nghiệm của mình bằng cách để lại đánh giá, đóng góp vào việc xây dựng cộng đồng mua sắm đáng tin cậy.



Hình 3.3 Biểu đồ Use Case tác nhân khách hàng

3.5.3. Biểu đồ Use Case tác nhân quản trị

Quản trị: Đóng vai trò là người điều hành toàn bộ hoạt động của hệ thống, quản trị có toàn quyền kiểm soát thông tin sản phẩm. Họ có thể thêm sản phẩm mới, cập nhật thông tin chi tiết về sản phẩm như mô tả, giá cả và hình ảnh hoặc xóa sản phẩm không còn kinh doanh. Bên cạnh đó, quản trị còn có khả năng quản lý danh sách sản phẩm, đảm bảo tính chính xác và cập nhật của thông tin. Việc xem danh sách đơn hàng, cập nhật trạng thái đơn hàng và giải quyết các vấn đề liên quan đến đơn hàng cũng nằm trong phạm vi trách nhiệm của quản trị. Ngoài ra, quản trị còn có thể quản các bài blog bằng cách thêm, sửa, xóa và ẩn hiện thông tin bài blog. Để nắm bắt tình hình kinh doanh, quản trị có thể xem các báo cáo thống kê về doanh số, sản phẩm bán chạy, danh mục bán chạy, thông tin khách hàng và nhiều chỉ số quan trọng khác.

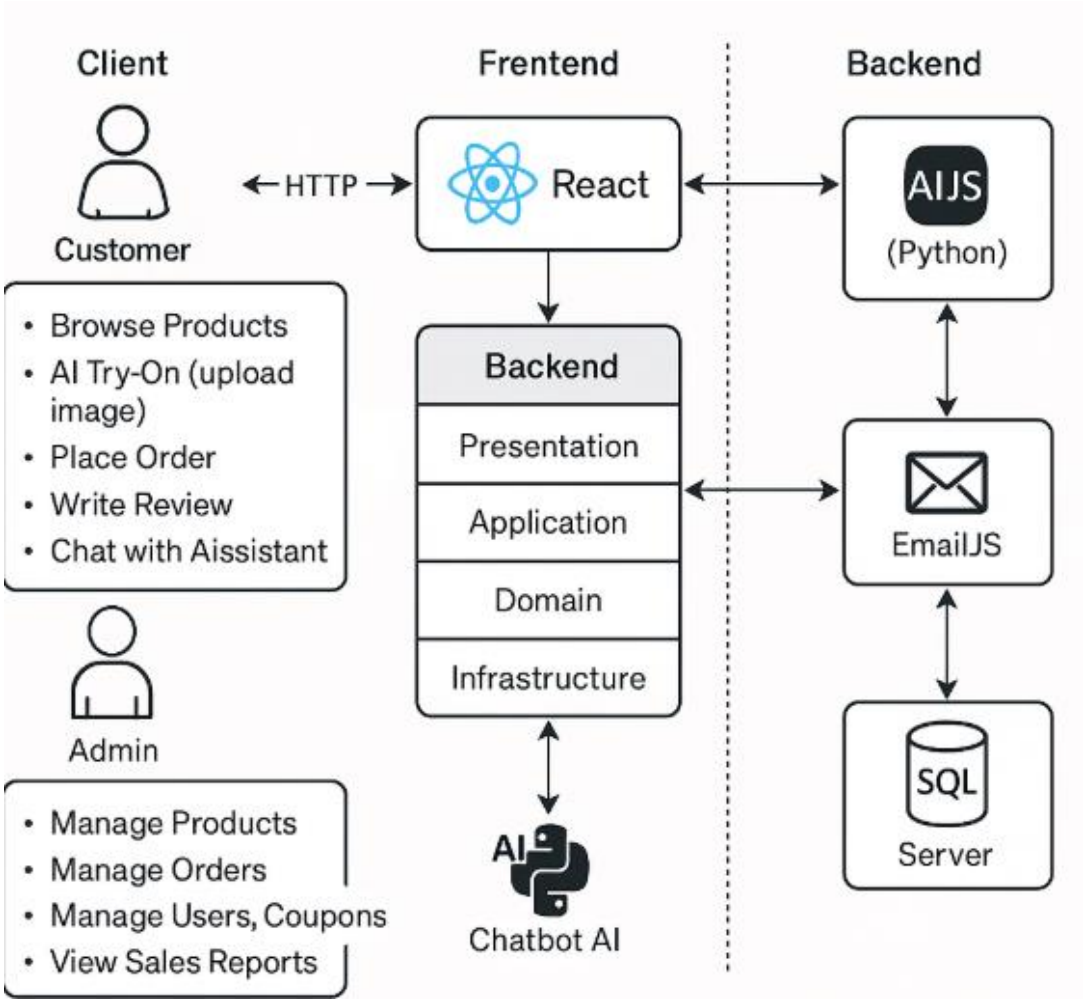


Hình 3.4 Biểu đồ Use Case tác nhân quản trị

3.6. Kiến trúc hệ thống

Hệ thống ứng dụng web được xây dựng theo kiến trúc ba lớp, sử dụng các công nghệ phổ biến và mạnh mẽ trong phát triển phần mềm hiện nay bao gồm: React JS cho giao diện người dùng (frontend), ASP.NET Core Web API cho xử lý logic nghiệp vụ (backend), và Microsoft SQL Server cho lưu trữ dữ liệu (database). Mô hình này giúp

tách biệt rõ ràng các tầng chức năng, tạo thuận lợi trong việc phát triển, mở rộng và bảo trì hệ thống.



Hình 3.5 Kiến trúc hệ thống

React JS là thư viện JavaScript được sử dụng để xây dựng giao diện người dùng hiện đại, thân thiện và phản hồi nhanh. Đây là nơi người dùng tương tác trực tiếp với hệ thống, thực hiện các hành động như đăng nhập, nhập dữ liệu, gửi biểu mẫu... Mọi dữ liệu và thao tác từ người dùng sẽ được gửi về backend thông qua các yêu cầu API (sử dụng HTTP qua thư viện fetch). React hoạt động hoàn toàn độc lập với backend, giúp tăng tính linh hoạt và dễ tái sử dụng.

ASP.NET Core Web API đóng vai trò là trung tâm xử lý logic của hệ thống, là cầu nối giữa giao diện người dùng và cơ sở dữ liệu. Các chức năng nghiệp vụ như xác thực người dùng, xử lý đơn hàng, truy vấn dữ liệu… đều được xử lý tại lớp này. Web API nhận yêu cầu từ frontend, xử lý, sau đó phản hồi dữ liệu ở định dạng JSON. Công nghệ ASP.NET Core hỗ trợ hiệu suất cao, khả năng mở rộng tốt và bảo mật mạnh mẽ.

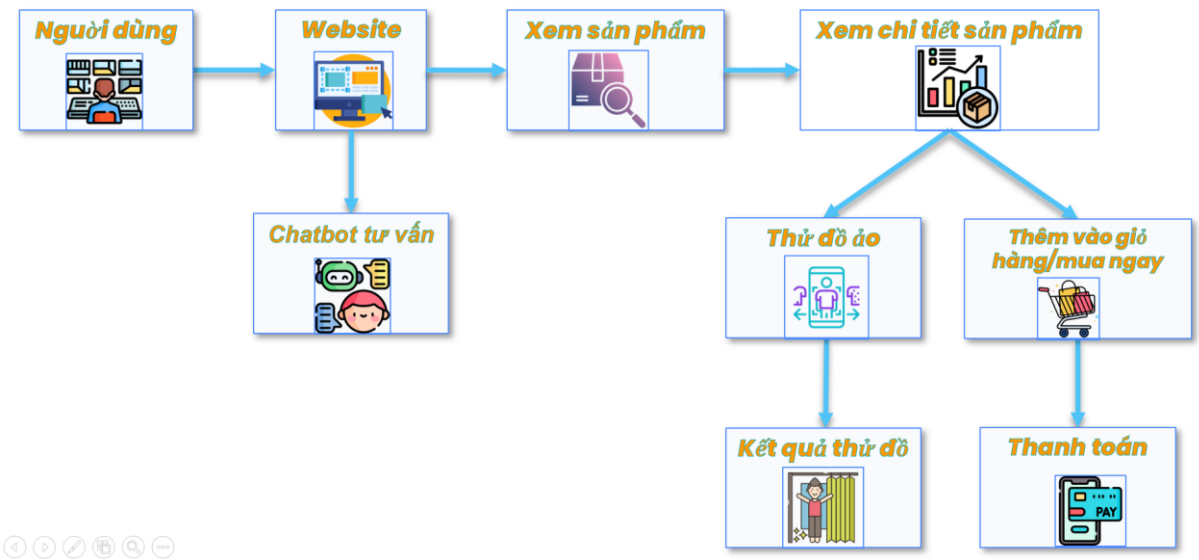
Microsoft SQL Server là hệ quản trị cơ sở dữ liệu quan hệ được sử dụng để lưu trữ toàn bộ dữ liệu của hệ thống như thông tin người dùng, sản phẩm, đơn hàng, lịch sử giao dịch. Lớp Web API sử dụng các kỹ thuật như Entity Framework Core để thao tác với dữ liệu tạo, đọc, cập nhật và xóa (CRUD). SQL Server đảm bảo tính toàn vẹn dữ liệu, hỗ trợ truy vấn tối ưu và khả năng mở rộng cho hệ thống lớn.

3.6.1. Mô hình hoạt động hệ thống

Khách hàng bắt đầu hành trình trải nghiệm bằng cách truy cập vào hệ thống website. Tại trang chủ, khách hàng có thể thực hiện nhiều hành động như tìm kiếm sản phẩm, xem thông tin khuyến mãi, đăng nhập hoặc tương tác với chatbot tư vấn.

Khách hàng có thể sử dụng chatbot tư vấn thông minh để nhận hỗ trợ, giải đáp thắc mắc về sản phẩm, kích thước, giá cả, chính sách đổi trả,... Chatbot hoạt động 24/7 và có thể kết nối đến cơ sở dữ liệu để đưa ra câu trả lời chính xác.

Khách hàng duyệt qua danh sách sản phẩm (theo danh mục, từ khóa tìm kiếm hoặc gợi ý). Mỗi sản phẩm có thể được nhấn vào để xem thông tin chi tiết hơn.



Hình 3.6 Mô hình hoạt động hệ thống

Giao diện chi tiết sản phẩm hiển thị đầy đủ thông tin: hình ảnh, mô tả, giá, đánh giá, màu sắc, kích thước và các lựa chọn. Từ đây, khách hàng có thể chọn một trong hai hướng (thử đồ ảo, thêm vào giỏ hàng hoặc mua ngay).

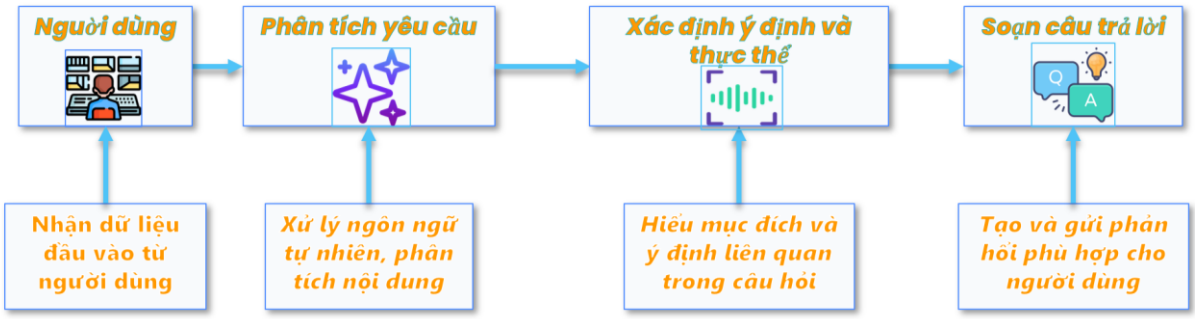
Khách hàng có thể trải nghiệm tính năng thử đồ ảo ngay tại trang chi tiết sản phẩm để kiểm tra sự phù hợp về kiểu dáng, màu sắc trên cơ thể. Sau khi thử đồ, hệ thống

sẽ trả về kết quả thử đồ ảo, cho phép khách hàng đưa ra quyết định phù hợp. Nếu khách hàng hài lòng, họ có thể thêm sản phẩm vào giỏ hàng hoặc chọn mua ngay.

Cuối cùng là thanh toán đây là bước cuối cùng trong luồng khách hàng khi đặt mua một sản phẩm tại website. Hệ thống sẽ hướng dẫn khách hàng nhập thông tin thanh toán, địa chỉ nhận hàng, mã giảm giá và lựa chọn phương thức thanh toán phù hợp.

3.6.2. Mô hình hoạt động của ChatbotAI

Người dùng nhập văn bản hoặc gửi câu hỏi vào giao diện chatbot. Đây là bước đầu tiên trong luồng hoạt động, nơi chatbot tiếp nhận các tín hiệu đầu vào (input) dưới dạng ngôn ngữ tự nhiên (tiếng Việt, tiếng Anh,...). Hệ thống sử dụng các kỹ thuật NLP để phân tích ngữ nghĩa câu hỏi. Mục tiêu là tách các thành phần quan trọng như chủ thể, hành động, đối tượng... và nhận diện rõ nhu cầu thực sự mà người dùng đang hướng đến.

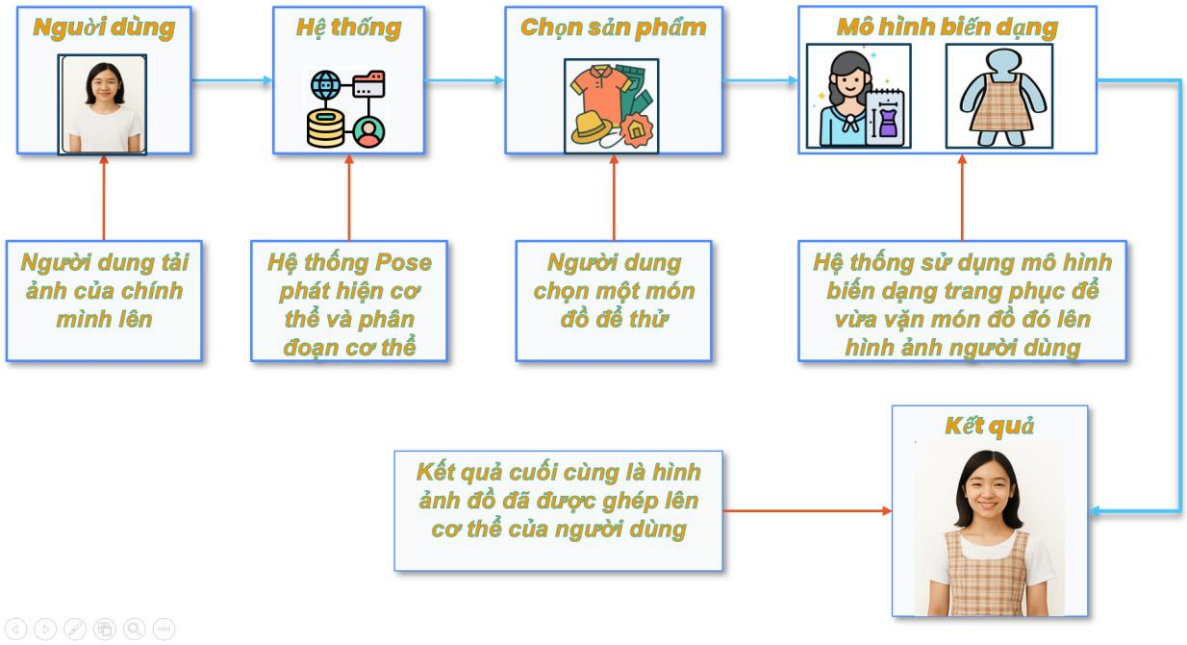


Hình 3.7 Mô hình hoạt động chatbot AI

Dựa trên kết quả phân tích, chatbot tiến hành xác định "ý định" (intent) của người dùng và các thực thể (entities) liên quan (sản phẩm, màu sắc, kích thước, danh mục...). Cuối cùng, chatbot tạo câu trả lời phù hợp với ngữ cảnh, có thể bao gồm, câu trả lời dạng văn bản tự nhiên, gợi ý sản phẩm có liên quan, đường dẫn tới trang chi tiết hoặc giỏ hàng.

3.6.3. Mô hình hoạt động tính năng thử đồ ảo

Khi truy cập vào trang chi tiết của một sản phẩm thời trang, người dùng có thể nhấn vào nút "Thử đồ ảo" để bắt đầu trải nghiệm tính năng. Đây là điểm kích hoạt luồng xử lý thử đồ thông qua AI. Sau khi chọn thử đồ, hệ thống yêu cầu người dùng cung cấp ảnh toàn thân. Website sẽ gửi một HTTP POST request đến endpoint của FitRoom API.



Hình 3.8 Mô hình hoạt động tính năng thử đồ ảo

Sau khi nhận được yêu cầu từ website, phía hệ thống của FitRoom sẽ thực hiện xử lý ảnh bằng các mô hình trí tuệ nhân tạo tiên tiến như MC-VTON. Hệ thống sẽ trang phục ảo lên ảnh cơ thể thật của người dùng, giữ lại tỷ lệ cơ thể, màu da, dáng người và tạo hình ảnh mặc thử sát thực tế.

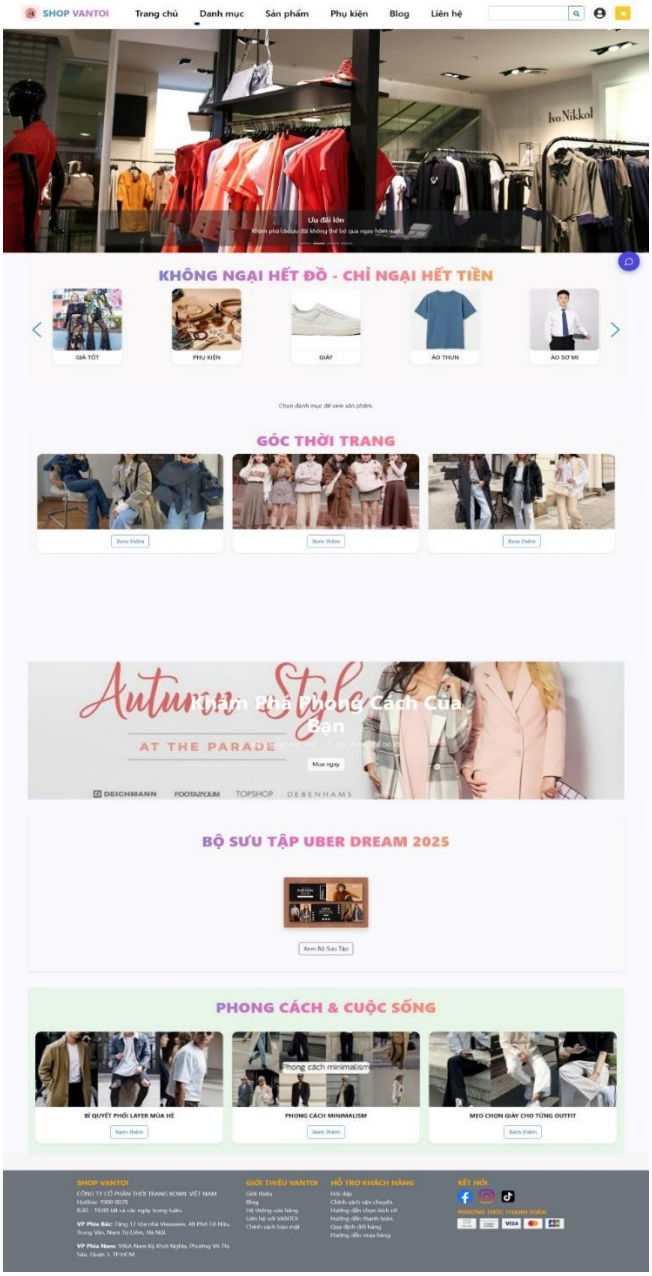
Sau khi xử lý xong, hệ thống FitRoom sẽ gửi lại kết quả cho website qua API response. Dữ liệu phản hồi có ảnh được mã hóa dưới dạng base64. Giao diện frontend tiếp nhận kết quả và hiển thị ảnh thử đồ cho người dùng ngay tại trang sản phẩm.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Giao diện khách hàng

4.1.1. Giao diện trang chủ

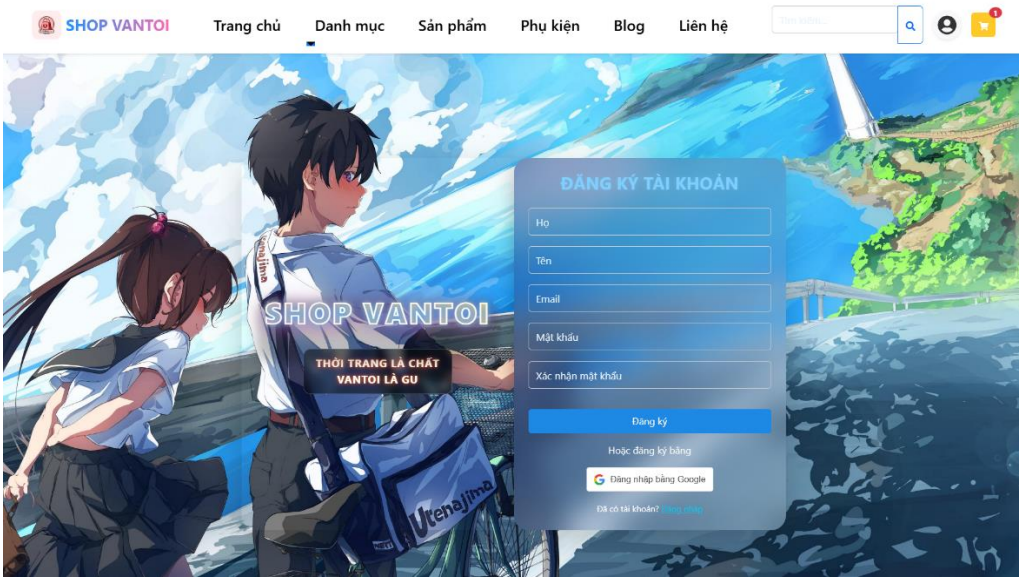
Giao diện trang chủ là trang đầu tiên khách hàng khi truy cập vào địa chỉ website nhìn thấy. Thiết kế của trang chủ được xây dựng với bố cục thông minh, tạo điều kiện thuận lợi để khách hàng dễ dàng theo dõi những sản phẩm mới, góc thời thời trang, phong cách thời trang và bộ sưu tập giúp khách hàng tìm kiếm được sản phẩm đúng với nhu cầu và gu thời trang của mình.



Hình 4.1 Giao diện trang chủ

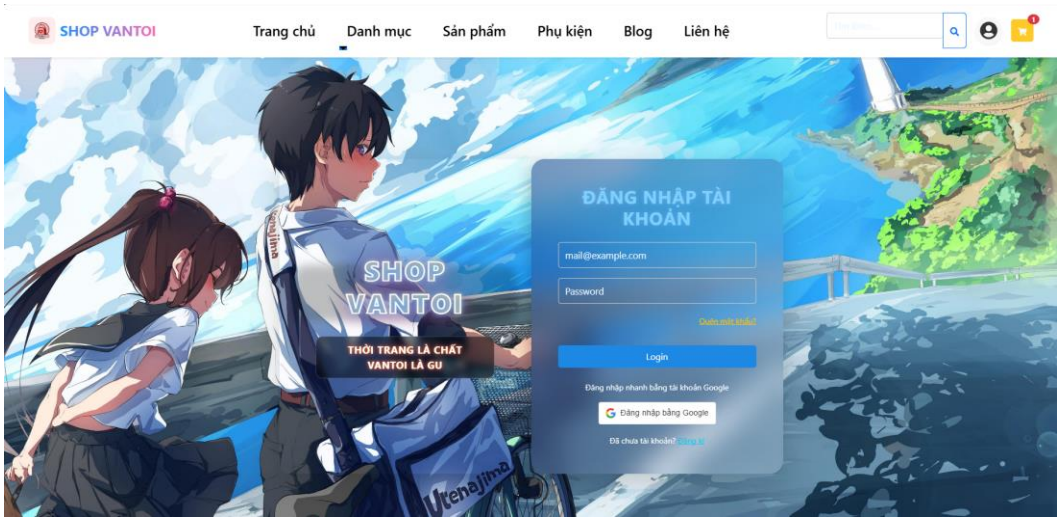
4.1.2. Giao diện đăng ký, đăng nhập

Giao diện đăng ký tài khoản thành viên giúp khách hàng có thể đăng ký tài khoản nhằm cung cấp và lưu trữ thông tin, hỗ trợ cho việc cung cấp thông tin nhận hàng với lần mua hàng đầu tiên sau đó lưu trữ và được sử dụng cho những lần mua hàng tiếp theo.



Hình 4.2 Giao diện đăng ký tài khoản

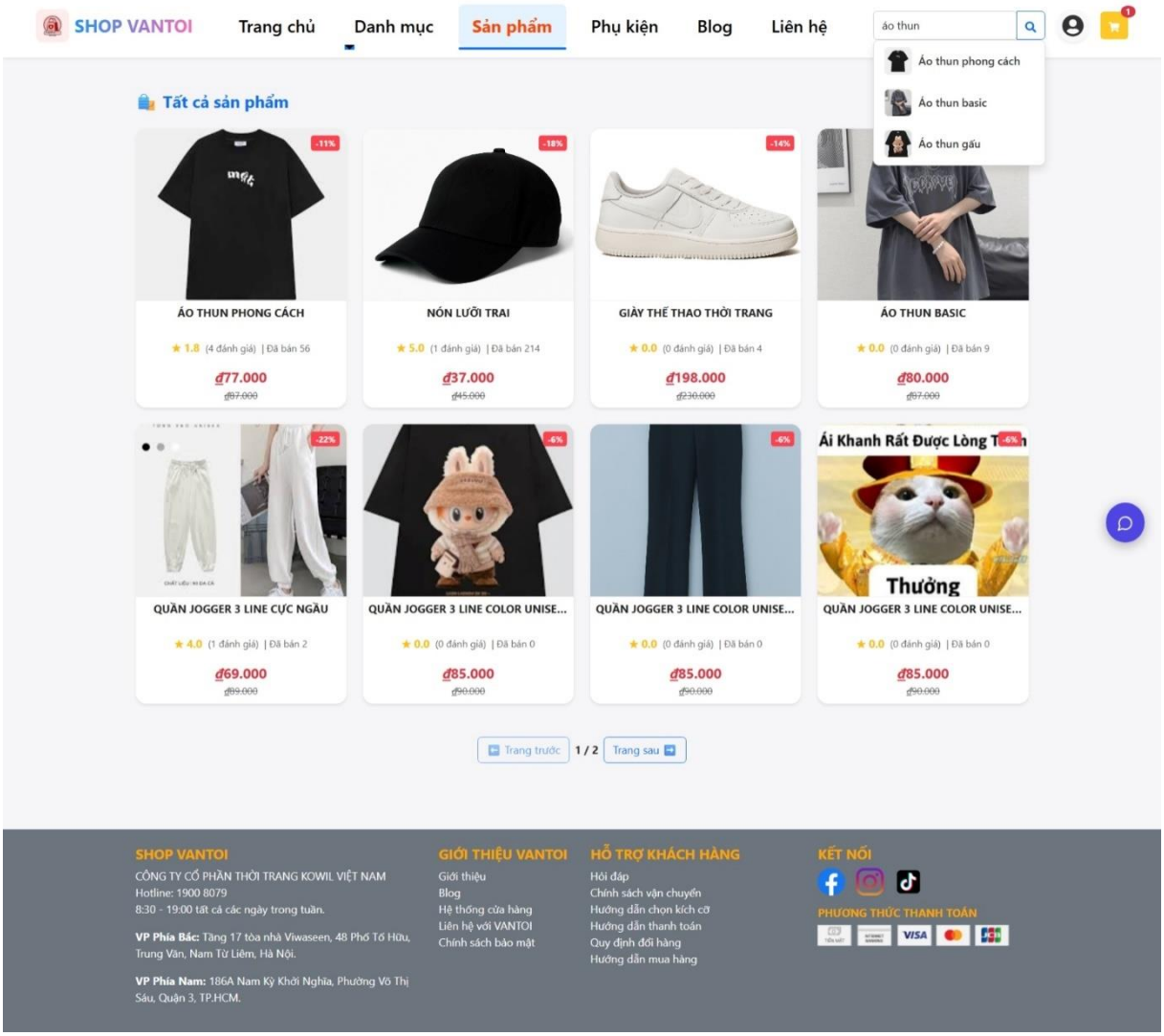
Khi khách hàng đã có tài khoản, giao diện đăng nhập giúp khách hàng đăng nhập để sử dụng những thông tin đã lưu trữ trên website. Tại đây khách hàng có thể đăng nhập với tài khoản đã đăng ký trước đó hoặc tùy chọn đăng nhập bằng tài khoản Google.



Hình 4.3 Giao diện đăng nhập tài khoản

4.1.3. Giao diện sản phẩm

Cung cấp cho khách hàng cái nhìn tổng quan về những sản phẩm trong cửa hàng. Tại đây khách hàng có thể dễ dàng tìm kiếm được các sản phẩm ưng ý với các mẫu thời trang mới nhất trong năm. Ngoài ra còn có thể biết được lượt đánh giá, số lượng bán và số sao của sản phẩm.

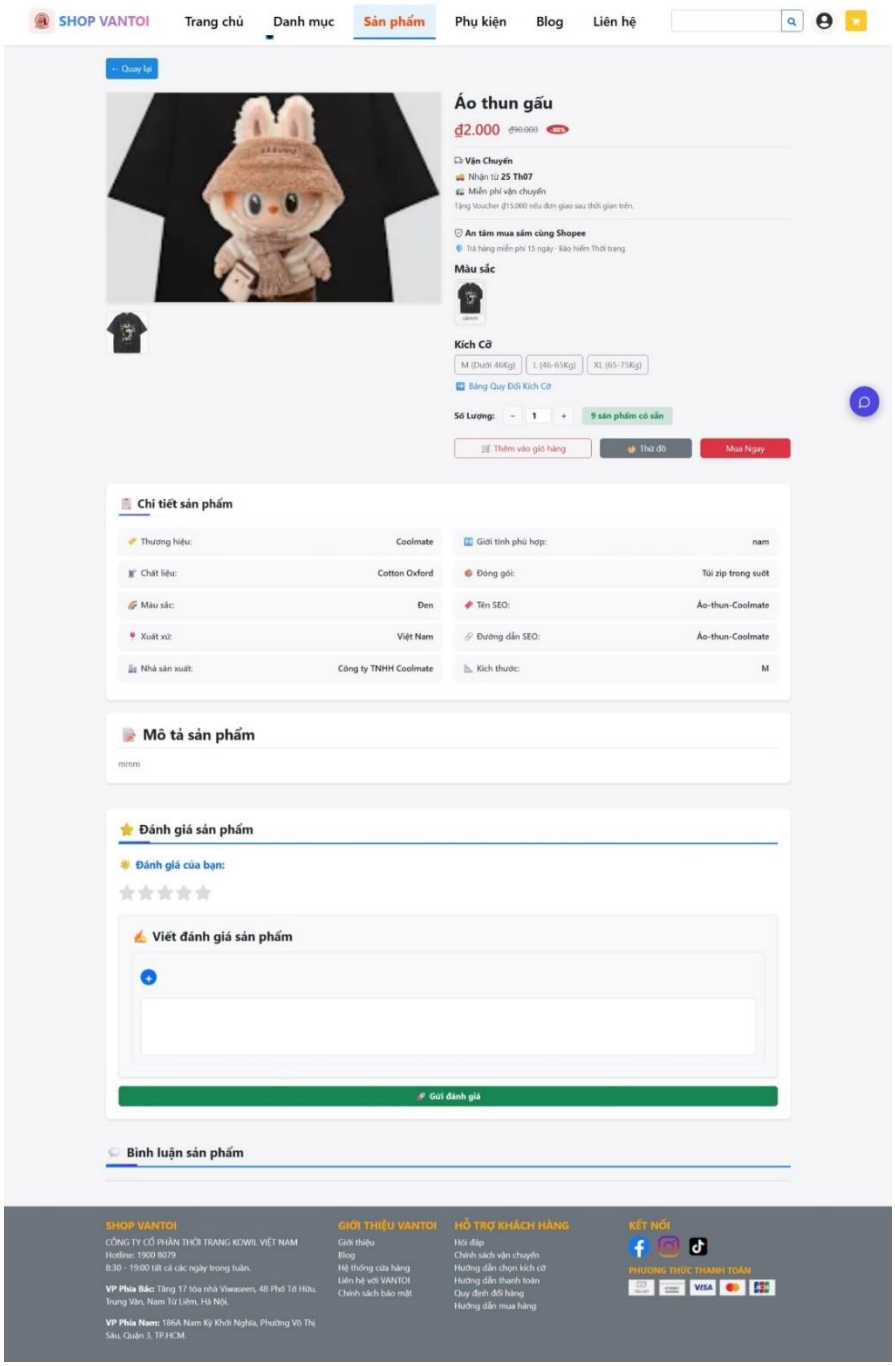


Hình 4.4 Giao diện sản phẩm

4.1.4. Giao diện chi tiết sản phẩm

Sau khi khách hàng chọn một sản phẩm bất kì, website sẽ chuyển hướng khách hàng đến giao diện chi tiết sản phẩm, tại đây khách hàng có thể xem được các hình ảnh chi tiết của sản phẩm. Bên cạnh đó khách hàng cũng có thể đọc được mô tả sản phẩm, xem nguồn gốc xuất xứ, bảng quy dổi kích cỡ. Hoặc có thể chọn đánh giá sản phẩm sau khi khách hàng đã mua hàng.

Giao diện chi tiết sản phẩm còn cung cấp cho khách hàng công cụ thử đồ ảo, khách hàng có thể tải ảnh của mình lên để ghép với ảnh sản phẩm để khách hàng có thể dễ dàng hình dung vóc dáng khi mặc.

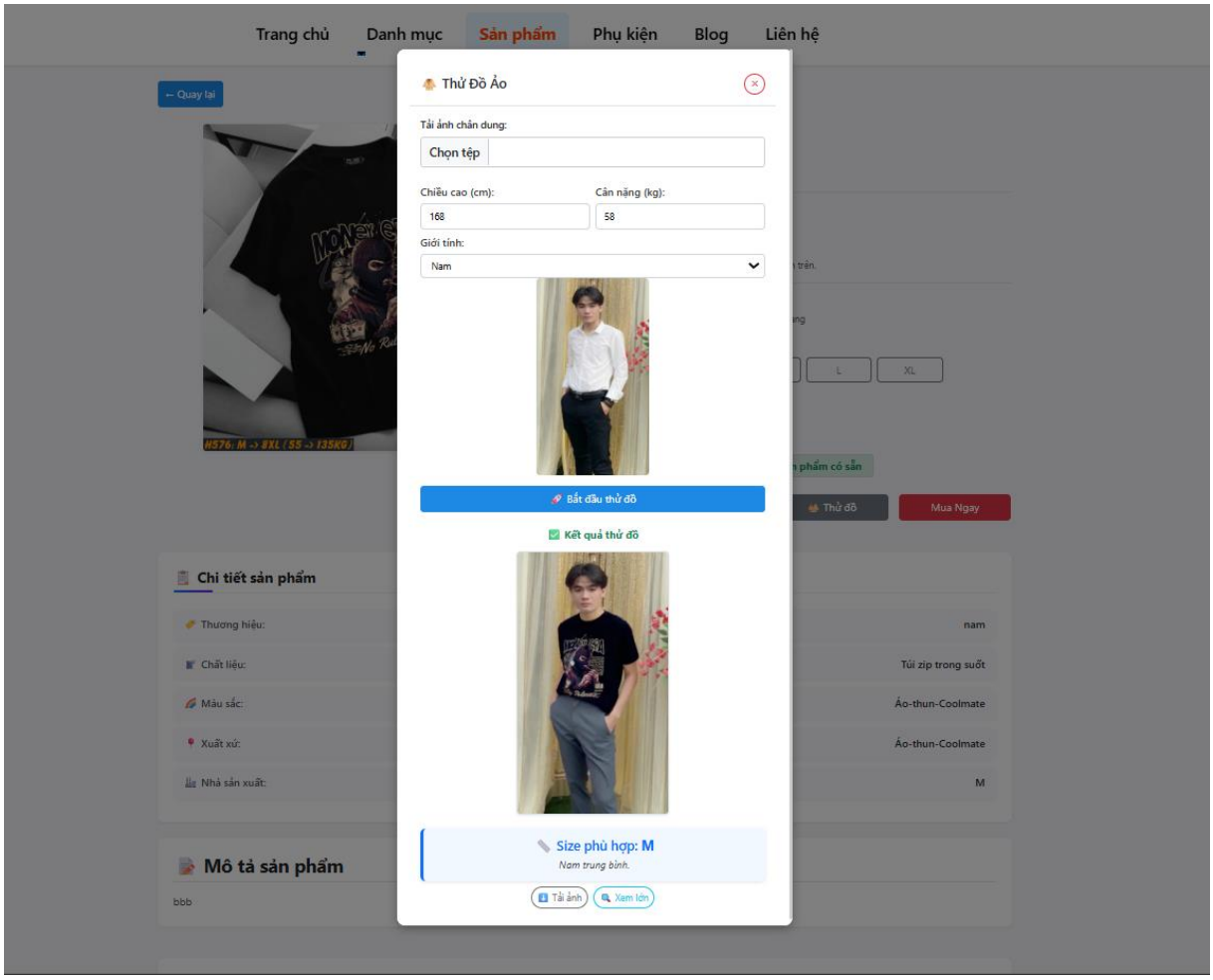


Hình 4.5 Giao diện chi tiết sản phẩm

4.1.5. Giao diện thử đồ ảo

Trang thử đồ áo cung cấp cho người dùng trải nghiệm thử áo trực tuyến một cách tiện lợi và trực quan. Người dùng có thể tải lên ảnh chân dung, nhập thông tin về chiều cao, cân nặng và chọn giới tính để hệ thống phân tích và hiển thị hình ảnh mặc thử áo

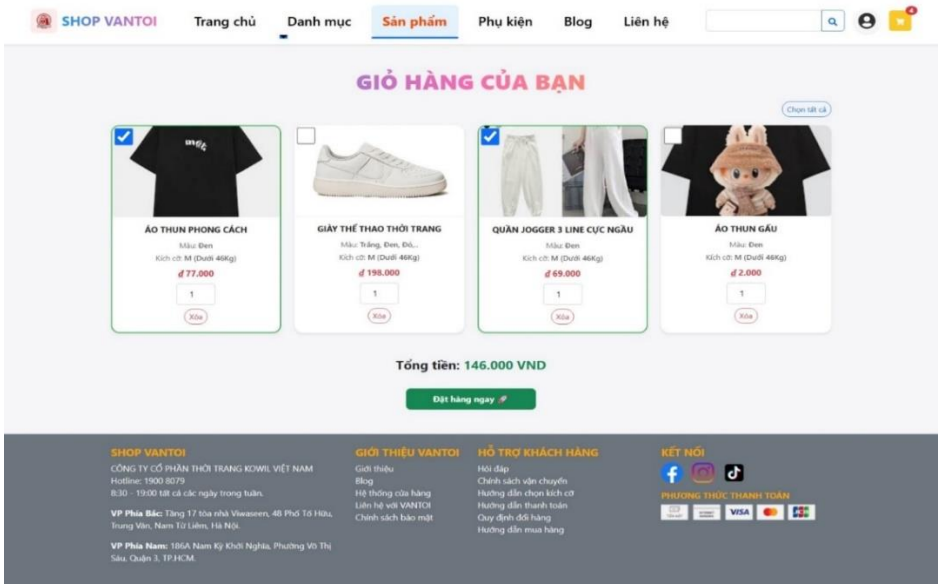
lên cơ thể mẫu. Sau khi thực hiện thao tác "Bắt đầu thử đồ", hệ thống sẽ trả về kết quả thử đồ kèm theo gợi ý size phù hợp dựa trên các thông số cá nhân đã nhập, giúp người dùng dễ dàng lựa chọn sản phẩm phù hợp với vóc dáng của mình. Chức năng này góp phần nâng cao trải nghiệm mua sắm, giảm rủi ro chọn sai kích cỡ khi đặt hàng online.



Hình 4.6 Giao diện thử đồ ảo

4.1.6. Giao diện giỏ hàng

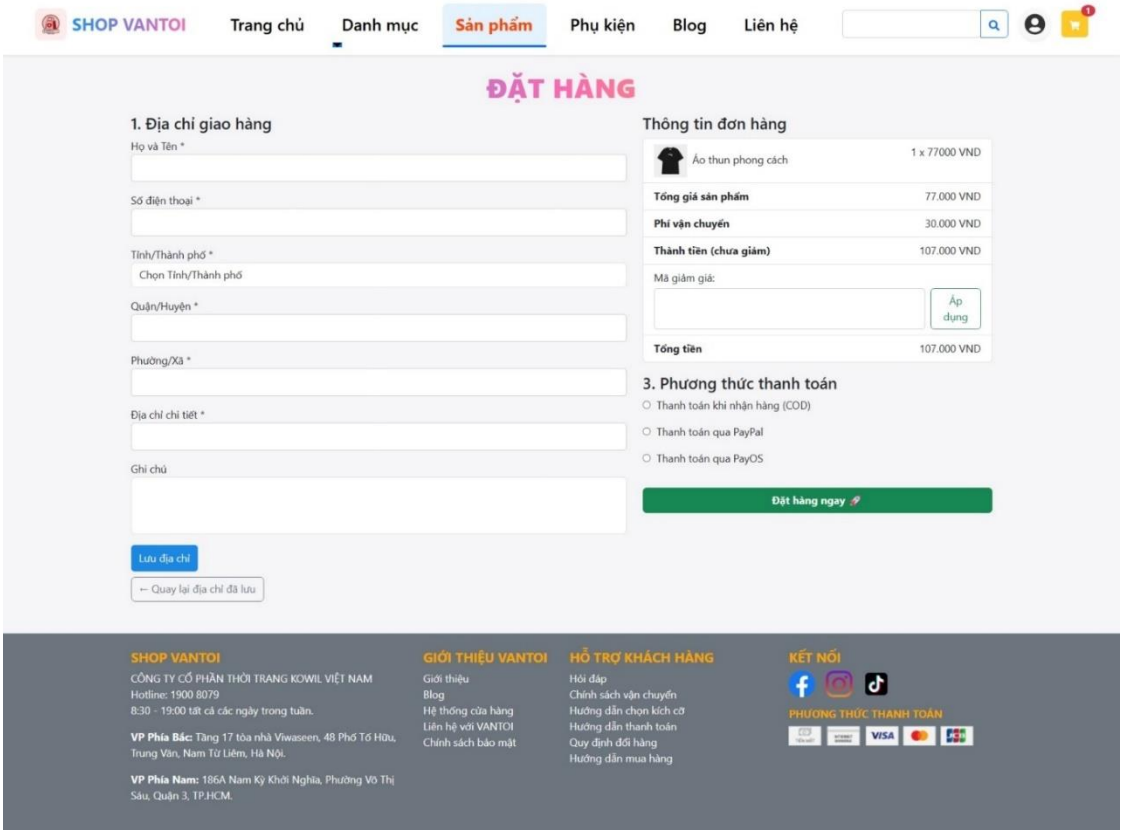
Giúp khách hàng khi đưa ra quyết định mua hàng có thể quản lý được số lượng và sản phẩm đã chọn. Giỏ hàng giúp hiển thị đầy đủ các thông tin về giá từng sản phẩm cũng như tổng số tiền phải trả cho khách hàng được biết.



Hình 4.7 Giao diện giỏ hàng

4.1.7. Giao diện thanh toán đơn hàng

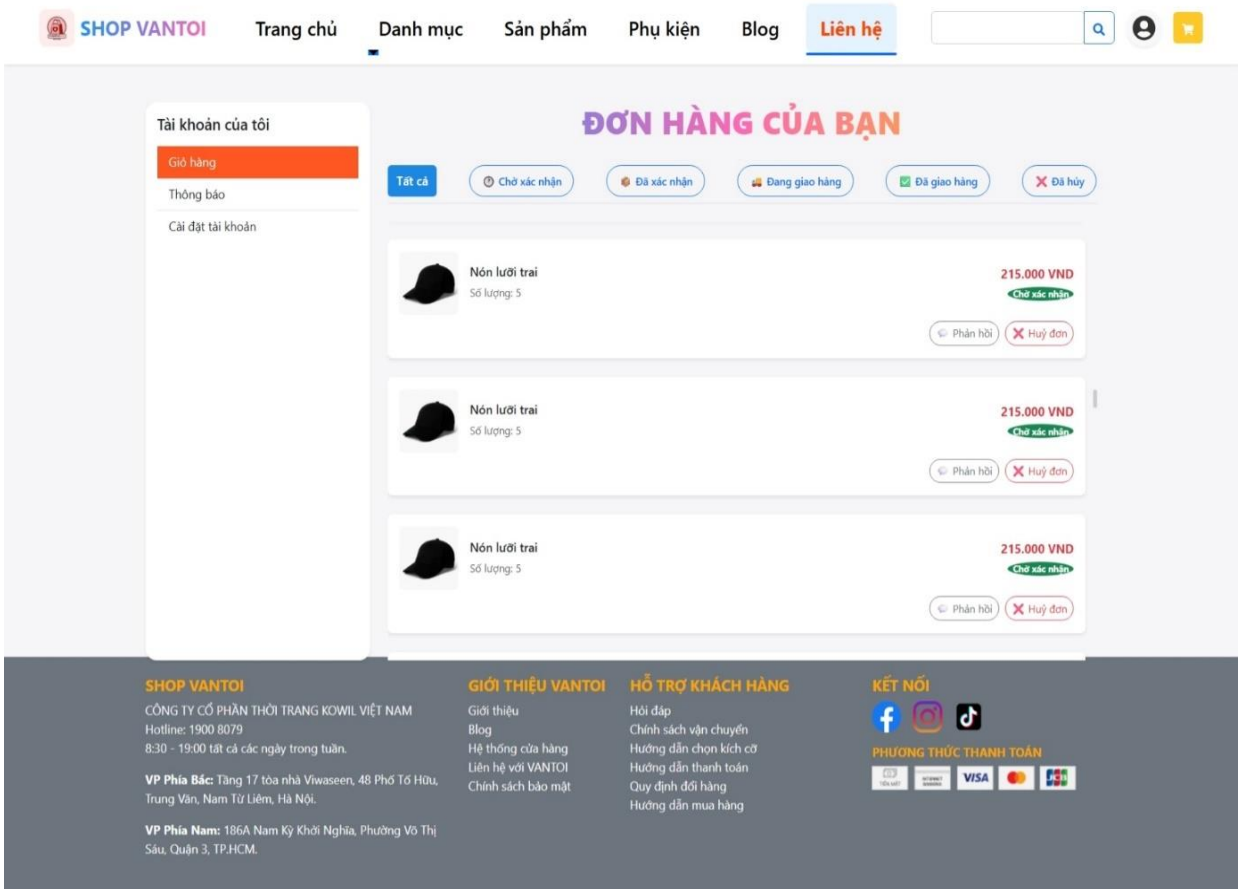
Sau khi khách hàng tiến hành xác nhận thanh toán các sản phẩm trong giỏ hàng, sẽ được chuyển đến trang thông tin đơn hàng, tại đây khách hàng cần chọn địa giao hàng nếu chưa có khách hàng có thể tạo mới để quá trình đặt hàng có thể được tiếp tục.



Hình 4.8 Giao diện thanh toán

4.1.8. Giao diện kiểm tra đơn hàng

Sau khi đặt hàng thành công, khách hàng có thể kiểm tra lại đơn hàng đã đặt bằng cách truy cập vào tài khoản của mình. Tại đây khách hàng có thể thấy được tất cả đơn hàng đã đặt, hoặc tùy chọn theo dõi đơn hàng theo các trạng thái giao hàng: chờ xác nhận, đang giao, đã giao, đã hủy. Khi đơn hàng đang ở trạng thái chờ xác nhận.



Hình 4.9 Giao diện theo dõi đơn hàng

4.1.9. Giao diện chat với trợ lí ảo AI

Chức năng Chat với trợ lý ảo AI mang đến cho khách hàng một trải nghiệm mua sắm trực tuyến tiện lợi và thông minh. Trợ lý ảo AI đóng vai trò như một người bạn đồng hành, sẵn sàng hỗ trợ khách hàng trong suốt quá trình mua sắm. Khách hàng có thể nhận được những gợi ý sản phẩm phù hợp với nhu cầu giúp tiết kiệm thời gian tìm kiếm và dễ dàng tìm thấy những sản phẩm đáp ứng đúng yêu cầu. Không chỉ tư vấn sản phẩm, trợ lý ảo AI còn giải đáp mọi thắc mắc về cách phối đồ, gu thời trang và trang phục phù hợp trong các bữua tiệc, hướng dẫn chi tiết từ những bước lựa chọn trang phục một cách hiệu quả và tối ưu nhất. Bên cạnh đó, trợ lý ảo AI cập nhật nhanh chóng và liên tục.

Hình 4.10 Giao diện Chat với trợ lý ảo AI

4.2. Giao diện quản trị

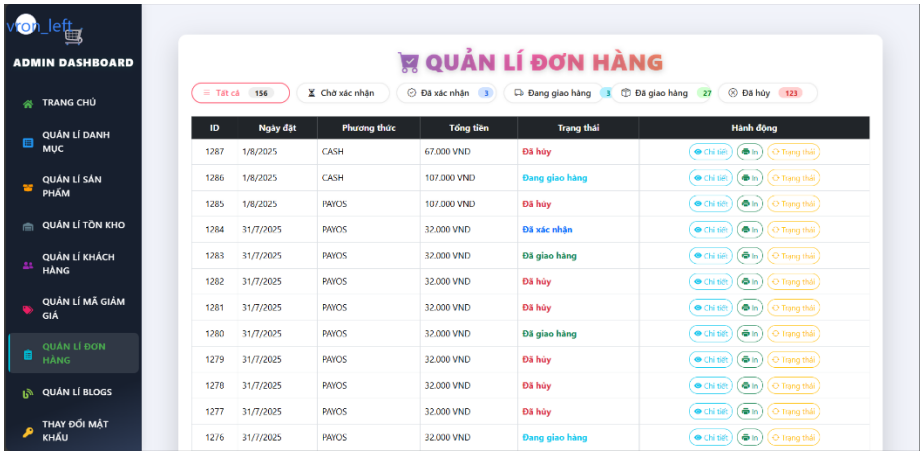
4.2.1. Giao diện thống kê

Giao diện thống kê này giúp quản trị viên dễ dàng theo dõi và tổng hợp các chỉ số kinh doanh quan trọng của hệ thống. Trên màn hình chính, người dùng có thể nhanh chóng xem được số lượng sản phẩm đã bán, doanh thu từ khách hàng mới, tổng chi phí và lợi nhuận thực tế. Ngoài ra, giao diện còn cung cấp các bộ lọc thời gian, cho phép người dùng tùy chỉnh khoảng thời gian thống kê để phân tích hiệu quả hoạt động. Các biểu đồ trực quan như doanh thu theo thời gian, doanh thu từ khách hàng mới, top sản phẩm bán chạy và doanh thu theo từng danh mục giúp quản trị viên nắm bắt tình hình kinh doanh một cách toàn diện.

Hình 4.11 Giao diện thống kê

4.2.2. Giao diện quản lý đơn hàng

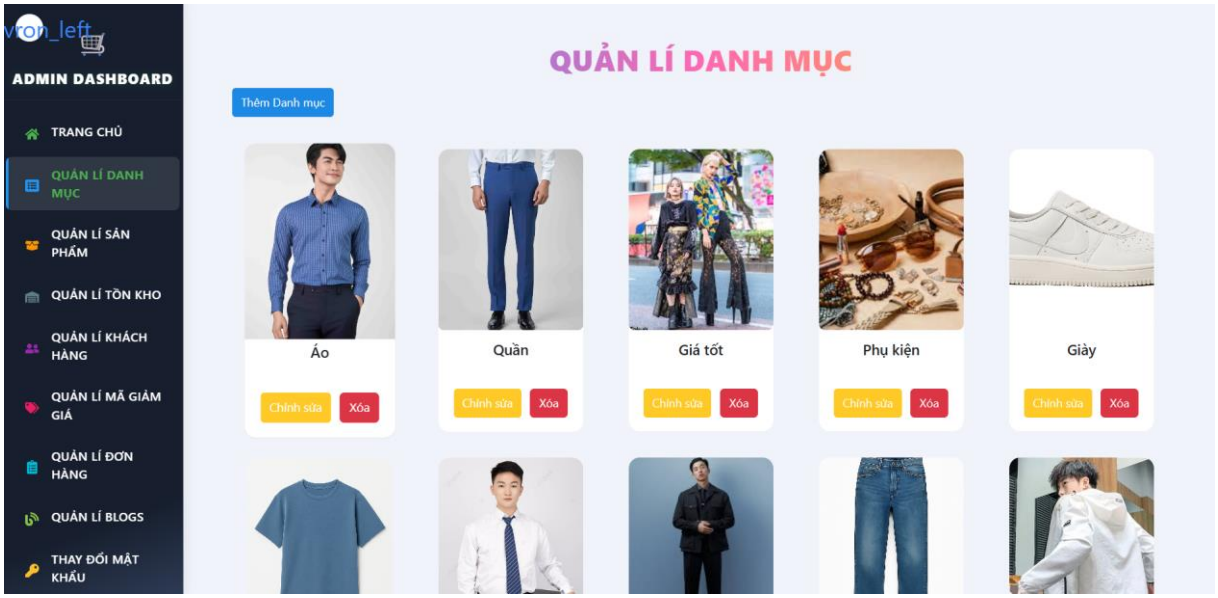
Trang quản lý đơn hàng cho phép quản trị viên theo dõi và kiểm soát toàn bộ quá trình xử lý đơn hàng của hệ thống. Giao diện hiển thị danh sách các đơn hàng với đầy đủ thông tin như mã đơn, ngày đặt, phương thức thanh toán, tổng tiền và trạng thái xử lý. Người dùng có thể dễ dàng lọc đơn hàng theo các trạng thái như chờ xác nhận, đã xác nhận, đang giao hàng, đã giao hàng hoặc đã hủy. Bên cạnh đó, các nút chức năng như xem chi tiết, in đơn hàng và cập nhật trạng thái giúp việc quản lý đơn hàng trở nên thuận tiện, nhanh chóng và chính xác.



Hình 4.12 Giao diện quản lý đơn hàng

4.2.3. Giao diện quản lý danh mục

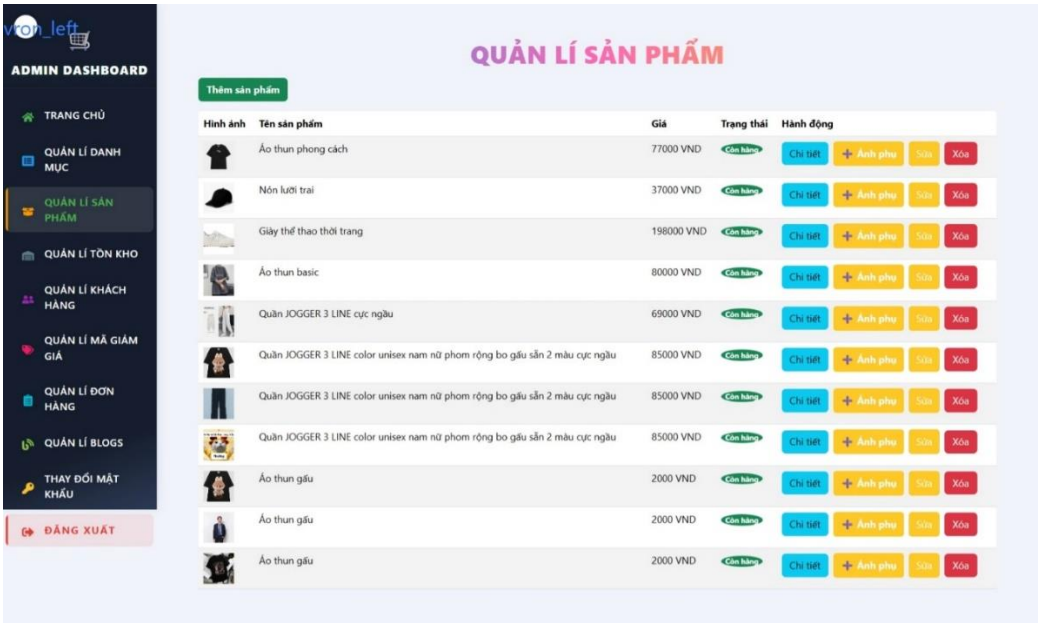
Trang quản lý danh mục cho phép quản trị viên dễ dàng theo dõi, chỉnh sửa và xóa các danh mục sản phẩm trong hệ thống. Giao diện hiển thị trực quan từng danh mục kèm hình ảnh đại diện, tên gọi và các nút chức năng như "Chỉnh sửa" và "Xóa" giúp thao tác quản lý trở nên nhanh chóng và thuận tiện. Ngoài ra, trang còn cung cấp tính năng thêm mới danh mục, hỗ trợ mở rộng hoặc cập nhật các nhóm sản phẩm một cách linh hoạt, đảm bảo hệ thống luôn phù hợp với nhu cầu kinh doanh hiện tại.



Hình 4.13 Giao diện quản lý danh mục

4.2.4. Giao diện quản lý sản phẩm

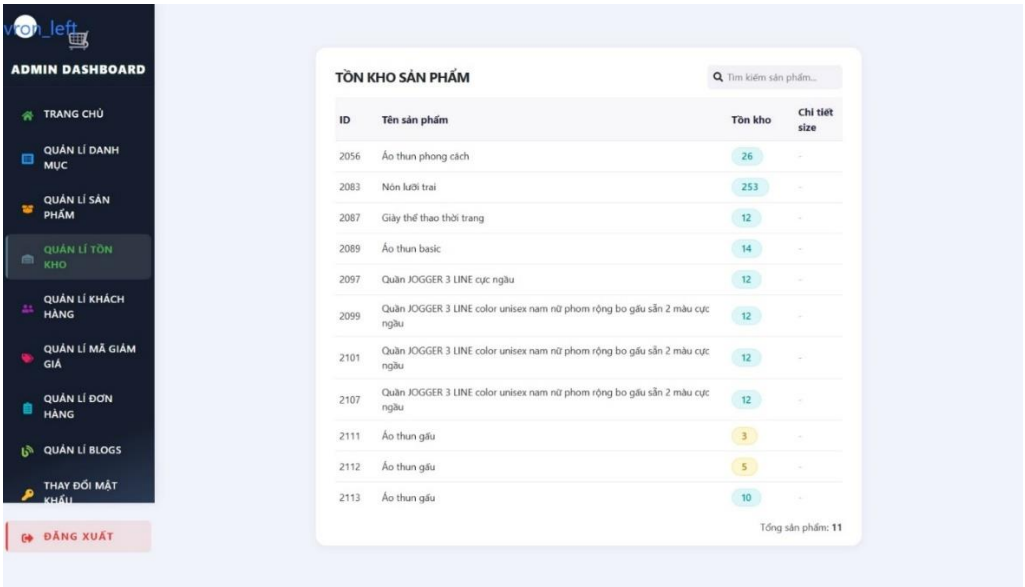
Trang quản lý sản phẩm cung cấp cho quản trị viên khả năng theo dõi, chỉnh sửa và kiểm soát toàn bộ danh sách sản phẩm trên hệ thống. Tại đây, mỗi sản phẩm đều được hiển thị với hình ảnh, tên, giá bán, trạng thái tồn kho và các nút chức năng như xem chi tiết, thêm ảnh phụ, sửa hoặc xóa sản phẩm. Ngoài ra, giao diện còn hỗ trợ chức năng thêm mới sản phẩm, giúp việc cập nhật và mở rộng danh mục hàng hóa diễn ra dễ dàng, nhanh chóng và hiệu quả.



Hình 4.14 Giao diện quản lý sản phẩm

4.2.5. Giao diện quản lý tồn kho

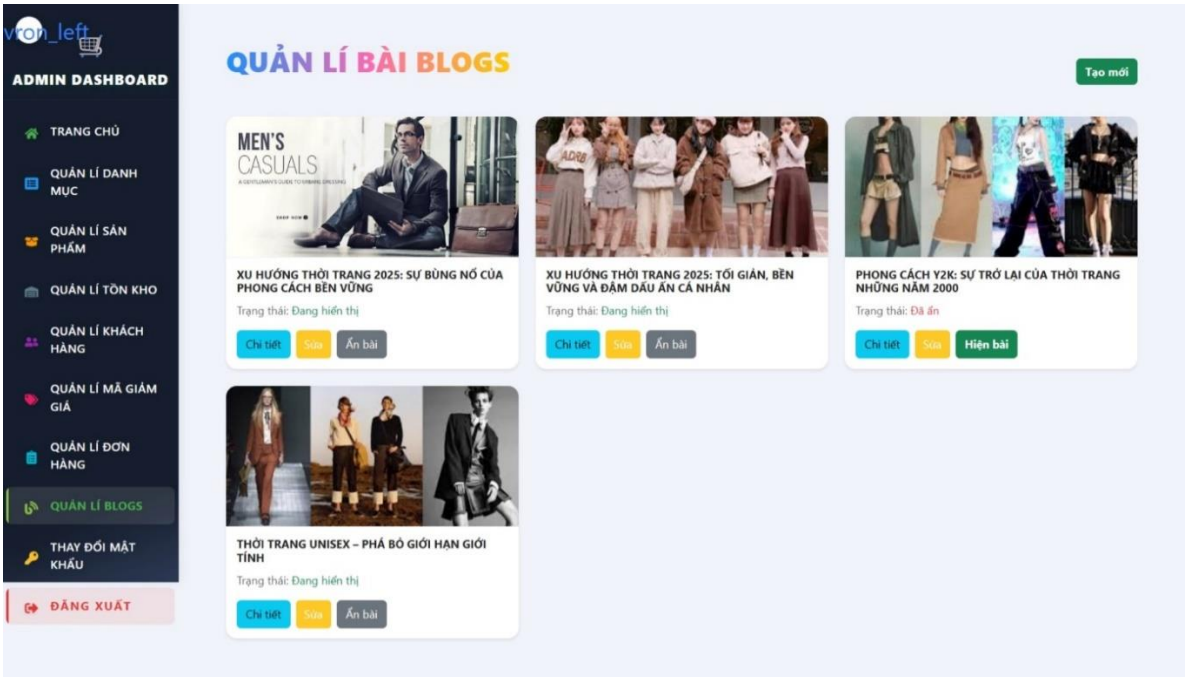
Trang quản lý tồn kho sản phẩm giúp quản trị viên kiểm soát số lượng hàng hóa hiện có trong kho một cách nhanh chóng và hiệu quả. Giao diện hiển thị danh sách các sản phẩm kèm theo ID, tên sản phẩm, số lượng tồn kho, cho phép người dùng dễ dàng theo dõi và cập nhật tình trạng hàng hóa. Chức năng tìm kiếm sản phẩm hỗ trợ việc tra cứu nhanh, đảm bảo việc quản lý kho luôn chính xác và thuận tiện, góp phần tối ưu quá trình vận hành và cung ứng sản phẩm.



Hình 4.15 Giao diện quản lý tồn kho

4.2.6. Giao diện quản lý Blogs

Trang quản lý bài blogs cho phép quản trị viên kiểm soát, chỉnh sửa và quản lý các bài viết về thời trang trên hệ thống. Giao diện hiển thị danh sách các bài blog với tiêu đề, hình ảnh đại diện, trạng thái hiển thị và các nút chức năng như xem chi tiết, sửa, ẩn bài hoặc hiện bài. Ngoài ra, quản trị viên có thể dễ dàng tạo mới bài blog, cập nhật nội dung và trạng thái bài viết để phù hợp với mục tiêu truyền thông của cửa hàng. Chức năng này giúp việc quản lý nội dung trở nên linh hoạt, đảm bảo các thông tin về xu hướng thời trang luôn được cập nhật nhanh chóng và hiệu quả.



Hình 4.16 Giao diện quản lý blog

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Sau quá trình nghiên cứu và triển khai, đề tài đã đạt được nhiều kết quả đáng ghi nhận. Hệ thống website bán thời trang đã được xây dựng thành công với giao diện hiện đại, trực quan và thân thiện với người dùng, cho phép thực hiện đầy đủ các chức năng cơ bản như đăng ký, đăng nhập, xem chi tiết sản phẩm, thêm vào giỏ hàng, đặt hàng và theo dõi trạng thái đơn hàng một cách dễ dàng. Một điểm nổi bật là việc tích hợp mô hình AI thử đồ ảo, giúp người dùng có thể tải ảnh chân dung lên để xem trước trang phục trên chính cơ thể mình, từ đó giảm thiểu tâm lý e ngại khi mua sắm trực tuyến.

Ngoài ra, hệ thống còn tích hợp chatbot AI sử dụng công nghệ xử lý ngôn ngữ tự nhiên (NLP), hỗ trợ tư vấn và giải đáp thắc mắc của khách hàng mà không cần sự can thiệp của nhân viên. Về mặt kỹ thuật, đề tài đã ứng dụng kiến trúc Clean Architecture kết hợp MediatR và Entity Framework Core, giúp hệ thống có tính mở rộng, dễ bảo trì và thuận tiện trong việc phát triển thêm các tính năng trong tương lai. Các chức năng quản lý sản phẩm, khách hàng, đơn hàng, mã giảm giá được triển khai đầy đủ và hoạt động ổn định, góp phần đảm bảo trải nghiệm người dùng mượt mà. Đồng thời, giao diện quản trị dành cho quản trị viên cũng đã được thiết kế và hoàn thiện, hỗ trợ hiệu quả các thao tác quản lý sản phẩm, duyệt đơn hàng và thống kê hoạt động kinh doanh.

5.2. Kết quả chưa đạt được

Mặc dù đề tài đã đạt được nhiều kết quả tích cực, nhưng vẫn còn một số điểm chưa hoàn thiện cần được cải thiện trong tương lai. Cụ thể, tính năng thử đồ ảo AI hiện tại còn khá đơn giản, chủ yếu chỉ thực hiện việc ghép ảnh quần áo lên ảnh người mà chưa thể xử lý các yếu tố phức tạp như đa góc nhìn, dáng đứng linh hoạt hoặc mô phỏng chất liệu vải có độ rủ như trong các mô hình hiện đại hơn như IDMVTON hay MCVTON. Tính năng này cũng yêu cầu người dùng phải tải ảnh có sẵn từ thiết bị, chưa hỗ trợ việc chụp ảnh trực tiếp từ webcam hoặc điện thoại trong trình duyệt. Bên cạnh đó, chatbot AI còn hạn chế trong việc hiểu ngữ cảnh sâu, phản hồi đôi khi chưa sát với nhu cầu của người dùng và chưa được huấn luyện chuyên biệt cho ngành thời trang tại Việt Nam. Hệ thống thanh toán trực tuyến như VNPay, MoMo và PayPal chưa được tích hợp đầy đủ do hạn chế về thời gian và tài nguyên trong quá trình thực hiện đề tài.

Ngoài ra, khả năng đề xuất sản phẩm thông minh dựa trên hành vi người dùng cùng với hệ thống đánh giá xếp hạng AI vẫn chưa được triển khai. Cuối cùng, các yếu tố liên quan đến bảo mật và kiểm thử hiệu năng chưa được đầu tư đúng mức do giới hạn về môi trường triển khai và thiết bị thử nghiệm, ảnh hưởng phần nào đến khả năng đánh giá tổng thể về hiệu quả hoạt động của hệ thống.

5.3. Hướng phát triển

Ứng dụng công nghệ AI và Machine Learning để phân tích hành vi, lịch sử mua hàng và sở thích của từng khách hàng nhằm đề xuất sản phẩm phù hợp, từ đó nâng cao trải nghiệm mua sắm và tăng khả năng quay lại của khách hàng.

Xây dựng các chương trình tích điểm, thẻ thành viên, coupon giảm giá để thu hút và giữ chân khách hàng lâu dài, đồng thời thúc đẩy doanh số bán hàng hiệu quả hơn.

Mở rộng các phương thức thanh toán tiện lợi và an toàn hơn như ví điện tử, QR Pay hoặc hỗ trợ mua trả góp. Đồng thời, tích hợp hệ thống vận chuyển với tính năng theo dõi đơn hàng, dự đoán thời gian giao hàng giúp nâng cao trải nghiệm khách hàng.

Xây dựng ứng dụng mobile cho các nền tảng iOS và Android để mở rộng phạm vi tiếp cận khách hàng và tạo thuận tiện trong việc mua sắm mọi lúc mọi nơi.

Tích hợp các tính năng chia sẻ sản phẩm, đánh giá và bình luận trên mạng xã hội để tăng tương tác và thu hút khách hàng mới.

Nếu điều kiện cho phép, tôi sẽ tiếp tục nghiên cứu cũng như tìm hiểu chi tiết hơn về nghiệp vụ bán hàng cũng như cách hình thành trang web bán hàng, tìm hiểu những kiến thức mới góp phần cải tiến website bán hàng hoàn chỉnh. Mục tiêu là không ngừng cải tiến trang web bán hàng của mình, đưa vào hoạt động những chức năng mới và tối ưu hóa hiệu suất để mang lại trải nghiệm tốt nhất cho người sử dụng.

Tôi tin rằng với sự nỗ lực và kiên trì, cùng với việc không ngừng học hỏi và áp dụng những kiến thức mới, tôi sẽ đạt được mục tiêu xây dựng một website bán hàng chuyên nghiệp, hiệu quả và mang lại giá trị thực sự cho khách hàng.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. "Comparing the Top 4 Open Source Virtual Try On (VITON) Models," Fashn.ai Blog. [Online]. Available: https://fashn.ai/blog/comparing-the-top-4-open-source-virtual-try-on-viton-models.

[2]. VOER. (n.d.). Giới thiệu cơ bản về ngôn ngữ ASP.NET. *Thư viện Học liệu Mở Việt Nam*. https://voer.edu.vn/m/gioi-thieu-co-ban-ve-ngon-ngu-aspnet/0da1b890

[3]. Troelsen, A., & Japikse, P. (2017). *Pro C# 7: With .NET and .NET Core* (8th ed.). Apress.

[4]. .NET Core VN. (2019, 13 tháng 9). Entity Framework là gì?. *.NET Core VN*. https://netcore.vn/bai-viet/kien-thuc-entity-framwork/entity-framework-la-gi

[5]. H. N. Thắng, "Triển khai trong .NET Core với MediatR CQRS Pattern," Viblo, [Online]. Available: https://viblo.asia/p/trien-khai-trong-net-core-voi-mediatr-cqrs-pattern-Yym40KKoV91.

[6]. TopDev, "RESTful API là gì? Tổng quan về kiến trúc REST & RESTful API," TopDev Blog, [Online]. Available: https://topdev.vn/blog/restful-api-la-gi/.

[7]. FPT Aptech, "RESTful API có ưu và nhược điểm gì?", Slides, [Online]. Available: https://slides.com/fptaptech/restful-api-co-uu-va-nhuoc-diem-gi#/2.

[8]. Martin, R. C. (2017). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

[9]. Trần, V. (2022, 25 tháng 5). Clean Architecture là gì – Ưu nhược và cách dùng hợp lý. *200Lab*. https://200lab.io/blog/clean-architecture-uu-nhuoc-va-cach-dung-hop-ly

[10]. DocumentProcessing.com, "MimeKit – Thư viện xử lý email cho .NET," DocumentProcessing.com, [Online]. Available: https://products.documentprocessing.com/vi/editor/net/mimekit/

[11]. 200Lab, "Ngrok là gì? Hướng dẫn sử dụng Ngrok chi tiết," 200Lab Blog, [Online]. Available: https://200lab.io/blog/ngrok-la-gi.

[12]. React. (n.d.). Getting started – React. *React Documentation*. Retrieved August 9, 2025, from https://react.dev/learn

[13] W3Schools Tech, "ReactJS: Ưu điểm và nhược điểm," W3Schools Tech, [Online]. Available: https://w3schools.tech/vi/tutorial/reactjs/reactjs_advantages_and_disadvantages.

[14]. Viettel IDC. (n.d.). Toàn tập về SQL Server cho người mới bắt đầu. *Viettel IDC*. https://viettelidc.com.vn/tin-tuc/toan-tap-ve-sql-server-cho-nguoi-moi-bat-dau

[15]. Ichi, "EmailJS: Cách kết nối biểu mẫu phản ứng nhiều bước với EmailJS," Ichi.pro, [Online]. Available: https://ichi.pro/vi/emailjs-cach-ket-noi-bieu-mau-phan-ung-nhieu-buoc-voi-emailjs-177585784782141.

[16]. SmartBear Software. (n.d.). *Swagger Documentation*. Retrieved June 8, 2025, from https://swagger.io/docs/

[17]. Postman. (n.d.). *Postman documentation overview*. Retrieved June 8, 2025, from https://learning.postman.com/docs/introduction/overview/

[18] C. T. Quang and T. P. Thành, “Tìm hiểu framework Spring cho backend, ReactJS cho frontend và xây dựng website bán giày minh họa,” Báo cáo thực tập Tốt nghiệp, Đại học Giao thông Vận tải TP. HCM, 2020.

[19] T. T. Phát, “Phát triển hệ thống quảng cáo dựa trên web sử dụng ReactJS và NodeJS,” Luận văn Tốt nghiệp, Đại học Cần Thơ, 2022.