

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIA ĐÌNH
KHOA: CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN

XÂY DỰNG HỆ THỐNG QUẢN LÝ BÁN HÀNG TRỰC TUYẾN VỚI SPRING BOOT

MÔN: CÔNG NGHỆ PHÁT TRIỂN PHẦN MỀM MỚI

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **KỸ THUẬT PHẦN MỀM**

Giảng viên hướng dẫn: **TRẦN QUỐC TRƯỜNG**

Sinh viên thực hiện: **NGUYỄN GIA BẢO**

MSSV: **22150450**

Lớp học phần: **221407**

TP. Hồ Chí Minh, tháng 4 năm 2025

MỤC LỤC

MỤC LỤC HÌNH ẢNH	iii
LỜI CẢM ƠN.....	iv
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI	1
1.1. Lý do chọn đề tài	1
1.2. Mục tiêu đề tài	1
1.3. Phương pháp nghiên cứu	2
CHƯƠNG 2: SƠ ĐỒ USE CASE	3
2.1. Giới thiệu sơ lược về sơ đồ Use Case	3
2.2. . Xác định Actors (Tác nhân).....	3
2.3. Xác định Use Cases (Các trường hợp sử dụng)	3
2.3.1. Customer Use Cases.....	3
2.3.2. Admin Use Cases	4
2.4. Sơ đồ Use Case	4
CHƯƠNG 3: TRÌNH BÀY CÔNG NGHỆ THỰC HIỆN	6
3.1. Giới thiệu chung	6
3.2. Công nghệ phía Backend (Backend Technologies)	6
3.2.1. Ngôn ngữ lập trình Java	6
3.2.2. Framework: Spring Boot.....	7
3.2.3. Truy cập dữ liệu (Data Access): Spring Data JPA.....	7
3.2.4. Cơ sở dữ liệu (Database): MySQL.....	8
3.2.5. Bảo mật (Security): Spring Security	8
3.2.6. Công cụ hỗ trợ: Lombok	9
3.3. Công nghệ phía Frontend (Frontend Technologies)	9
3.3.1. Ngôn ngữ Markup & Styling	9
3.3.2. Templating Engine: Thymeleaf.....	9
3.3.3. Framework CSS & UI Components: Bootstrap	10
3.4. Công cụ phát triển và quản lý (Development and Management Tools)	11

3.4.1. Môi trường phát triển tích hợp (IDE - Integrated Development Environment).....	11
3.4.2. Công cụ Build và Quản lý Dependency: Maven.....	11
3.4.3. Hệ quản trị cơ sở dữ liệu (DBMS Tool).....	11
3.4.4. Hệ thống quản lý phiên bản (Version Control System - VCS): Git.....	11
3.4.5. Máy chủ ứng dụng (Application Server): Embedded Tomcat	12
3.5. Tổng kết lý thuyết	12
CHƯƠNG 4: TRÌNH BÀY PROJECT	13
4.1. Giới thiệu	13
4.2. Cơ sở dữ liệu.....	13
4.2.1. Table Users.....	13
4.2.2. Table Customer	14
4.2.3. Table Authorities.....	15
4.2.4. Table Product	15
4.3. Giao diện Khách hàng (Customer Frontend).....	17
4.3.1. Trang chủ	17
4.3.2. Các trang khác.....	18
4.3.3. Trang Chi tiết Sản phẩm	21
4.3.4. Trang Đăng nhập.....	21
4.4. Hệ thống Quản trị	22
4.4.1. Trang Đăng nhập.....	22
4.4.2. Trang Admin	23
4.4.3. Các Trang Quản lý/ báo cáo khác	23
CHƯƠNG 5: TỔNG KẾT.....	29
5.1. Kết quả đạt được	29
5.2. Hạn chế và Điểm cần cải thiện	30
5.3. Hướng phát triển của đề tài.....	31
PHỤ LỤC: TÀI LIỆU THAM KHẢO	32

MỤC LỤC HÌNH ẢNH

Hình 2.4. Sơ đồ UseCase.....	5
Hình 4.3.1. Trang chủ của Customer.....	17
Hình 4.3.2.1. Trang Services	18
Hình 4.3.2.2. Trang About Us	19
Hình 4.3.2.3. Trang Blog.....	20
Hình 4.3.2.4. Trang Contact	20
Hình 4.3.3. Trang Chi tiết sản phẩm	21
Hình 4.3.4. Trang Login	21
Hình 4.4.1.1. Trang Login	22
Hình 4.4.1.2. Trang Chặn thâm nhập	22
Hình 4.4.2. Trang Admin	23
Hình 4.4.3.1. Trang list-customer.....	24
Hình 4.4.3.2. Trang customer-form-insert.....	24
Hình 4.4.3.3. Trang thông báo điền thiếu thông tin trong Customer-form-insert	25
Hình 4.4.3.4. Trang Customer-form-update	25
Hình 4.4.3.5. Trang list-product	26
Hình 4.4.3.6. Trang List-username.....	27
Hình 4.4.3.7. Trang Username-form-update	27
Hình 4.4.3.8. Trang List-user	28
Hình 4.4.3.9. Trang User-form-update	28

LỜI CẢM ƠN

Em xin chân thành cảm ơn sâu sắc và được bày tỏ lòng biết ơn đến với Quý Thầy Cô của khoa Công Nghệ Thông Tin Trường Đại học Gia Định và đặc biệt đối với thầy **ThS. Trần Quốc Trường** đã tận tình hướng dẫn, đồng hành, động viên và chỉ bảo giúp tôi hoàn thành tốt bài báo cáo này cũng như truyền đạt những kiến thức bổ ích và rất quan trọng đối với quá trình đi làm của chúng em sau này.

Trong thời gian nghiên cứu đề tài tiểu luận vừa qua, em đã có nhiều cố gắng trong suốt quá trình thực hiện tiểu luận, tích cực trao đổi thông tin, sưu tầm, tham khảo tài liệu và học hỏi thêm kiến thức từ các Thầy Cô và các học viên khác, không chỉ nhận được những kiến thức đầy bổ ích về chuyên môn mà còn ở những lĩnh vực khác. Những trải nghiệm quý báu đó không chỉ giúp chúng em hoàn thành tốt bài báo cáo mà còn là hành trang quan trọng theo tôi trong suốt thời gian học tập và giúp tôi có thể tự tin bước vào đời làm việc sau này.

Do kiến thức và kinh nghiệm còn hạn chế cho nên bài báo cáo của em còn nhiều thiếu sót, kính mong được sự đánh giá, góp ý của thầy.

Cuối cùng, xin kính chúc thầy luôn luôn khỏe mạnh, vui vẻ và đạt được nhiều thành công cao trong công tác giảng dạy. Chúc trường Đại học Gia Định sẽ luôn là nền tảng vững chắc cho nhiều thế hệ sinh viên tiếp bước trên con đường học tập.

Xin chân thành cảm ơn!

Nguyễn Gia Bảo

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, thương mại điện tử đã trở thành một phần không thể thiếu trong hoạt động kinh doanh của nhiều doanh nghiệp. Hình thức mua sắm trực tuyến không chỉ mang lại sự thuận tiện cho khách hàng mà còn giúp doanh nghiệp tiếp cận được lượng lớn người tiêu dùng mà không bị giới hạn bởi không gian địa lý. Để đáp ứng nhu cầu ngày càng cao của thị trường, các hệ thống quản lý bán hàng trực tuyến cần phải được xây dựng một cách bài bản, đảm bảo tính hiệu quả, bảo mật và dễ dàng mở rộng trong tương lai.

Spring Boot là một trong những framework mạnh mẽ giúp phát triển các ứng dụng web nhanh chóng, hiệu quả và linh hoạt. Nhờ khả năng tự động cấu hình, tích hợp sẵn nhiều thư viện hữu ích và hỗ trợ xây dựng các ứng dụng có cấu trúc rõ ràng, Spring Boot đã trở thành một lựa chọn phổ biến trong phát triển phần mềm hiện đại. Vì vậy, đề tài “Xây dựng hệ thống quản lý bán hàng trực tuyến với Spring Boot” được lựa chọn nhằm ứng dụng các công nghệ tiên tiến vào thực tiễn, đồng thời nâng cao kiến thức về phát triển phần mềm, thiết kế hệ thống và quản lý dữ liệu trong môi trường doanh nghiệp.

1.2. Mục tiêu đề tài

Mục tiêu chính của đề tài là xây dựng một hệ thống quản lý bán hàng trực tuyến với các chức năng cơ bản như quản lý sản phẩm, khách hàng, đơn hàng và thanh toán. Hệ thống cần phải đảm bảo hoạt động ổn định, dễ sử dụng và có thể mở rộng trong tương lai. Ngoài ra, dự án còn hướng đến việc ứng dụng các công nghệ hiện đại như Spring Boot để phát triển backend, Thymeleaf để xây dựng giao diện động, Bootstrap để tối ưu hiển thị trên các thiết bị khác nhau, và XAMPP để hỗ trợ phát triển và kiểm thử hệ thống trên môi trường local.

Bên cạnh mục tiêu xây dựng một hệ thống thực tiễn, đề tài còn giúp nâng cao kỹ năng lập trình, làm việc với framework Spring Boot và các công nghệ liên quan. Việc triển khai hệ thống cũng sẽ giúp người thực hiện hiểu sâu hơn về quy trình phát triển phần mềm, từ phân tích yêu cầu, thiết kế hệ thống, lập trình, kiểm thử cho đến triển khai và bảo trì.

1.3. Phương pháp nghiên cứu

Để thực hiện đề tài này, nhiều phương pháp nghiên cứu khác nhau sẽ được áp dụng. Trước tiên, phương pháp thu thập tài liệu sẽ được sử dụng để nghiên cứu các tài liệu, sách và hướng dẫn liên quan đến Spring Boot, Java, Thymeleaf, Bootstrap và XAMPP. Đây là giai đoạn quan trọng giúp xây dựng nền tảng lý thuyết vững chắc trước khi tiến hành các bước triển khai thực tế.

Tiếp theo, phương pháp thực nghiệm sẽ được áp dụng để xây dựng và triển khai hệ thống thực tế. Trong quá trình này, các tính năng sẽ được phát triển, kiểm thử và tối ưu hóa nhằm đảm bảo hệ thống hoạt động một cách hiệu quả. Bên cạnh đó, phương pháp so sánh cũng sẽ được thực hiện nhằm đánh giá và lựa chọn các công nghệ phù hợp nhất để ứng dụng vào hệ thống. Việc phân tích và thiết kế hệ thống theo mô hình MVC (Model-View-Controller) cũng là một phần quan trọng của nghiên cứu, giúp hệ thống có cấu trúc logic, dễ dàng bảo trì và mở rộng.

CHƯƠNG 2: SƠ ĐỒ USE CASE

2.1. Giới thiệu sơ lược về sơ đồ Use Case

Sơ đồ Use Case là một loại sơ đồ trong UML (Unified Modeling Language) dùng để mô tả sự tương tác giữa người dùng (Actors) và hệ thống. Nó giúp xác định các chức năng chính mà hệ thống cần cung cấp và ai là người sử dụng các chức năng đó.

Trong dự án này, chúng ta có hai phần chính: phần dành cho khách hàng (Final/customer) và phần dành cho quản trị viên (Final/admin). Dựa vào cấu trúc code và các file templates, chúng ta có thể xác định các Actor và Use Case chính.

2.2. . Xác định Actors (Tác nhân)

Dựa trên hai thư mục customer và admin, chúng ta có thể xác định hai Actor chính:

- Customer (Khách hàng): Người dùng cuối tương tác với giao diện công khai của website. Họ có thể xem thông tin, sản phẩm và thực hiện các hành động liên quan đến tài khoản của mình.
- Admin (Quản trị viên): Người quản lý hệ thống, có quyền truy cập vào các chức năng quản trị như quản lý người dùng, xem báo cáo, cấu hình hệ thống.

2.3. Xác định Use Cases (Các trường hợp sử dụng)

Dựa vào các controllers và file templates (.html), chúng ta xác định các chức năng (Use Cases) chính cho từng Actor:

2.3.1. Customer Use Cases

Browse Website: Xem các trang công khai như Trang chủ (home.html, index.html), Giới thiệu (about-us.html), Liên hệ (contact-us.html), Dịch vụ (services.html), Bảng giá (pricing.html), Blog (blog.html).

View Products: Xem danh sách sản phẩm (có thể nằm trong home.html hoặc trang riêng).

View Product Details: Xem thông tin chi tiết của một sản phẩm cụ thể (detail-product.html).

Login/Logout: Đăng nhập vào hệ thống và đăng xuất (login.html, SecurityConfig).

Manage Account: Bao gồm đăng ký (nếu có) và cập nhật thông tin cá nhân (liên quan đến CustomerController trong Final/customer).

2.3.2. Admin Use Cases

Login/Logout: Đăng nhập vào khu vực quản trị và đăng xuất (login.html, SecurityConfig).

Manage Customers: Quản lý thông tin khách hàng, bao gồm xem danh sách (list-customer.html), thêm mới (customer-form-insert.html), cập nhật (customer-form-update.html), và có thể cả xóa (thường nằm trong CustomerController).

View Dashboard/ Reports: Xem trang tổng quan (admin.html, index.html)

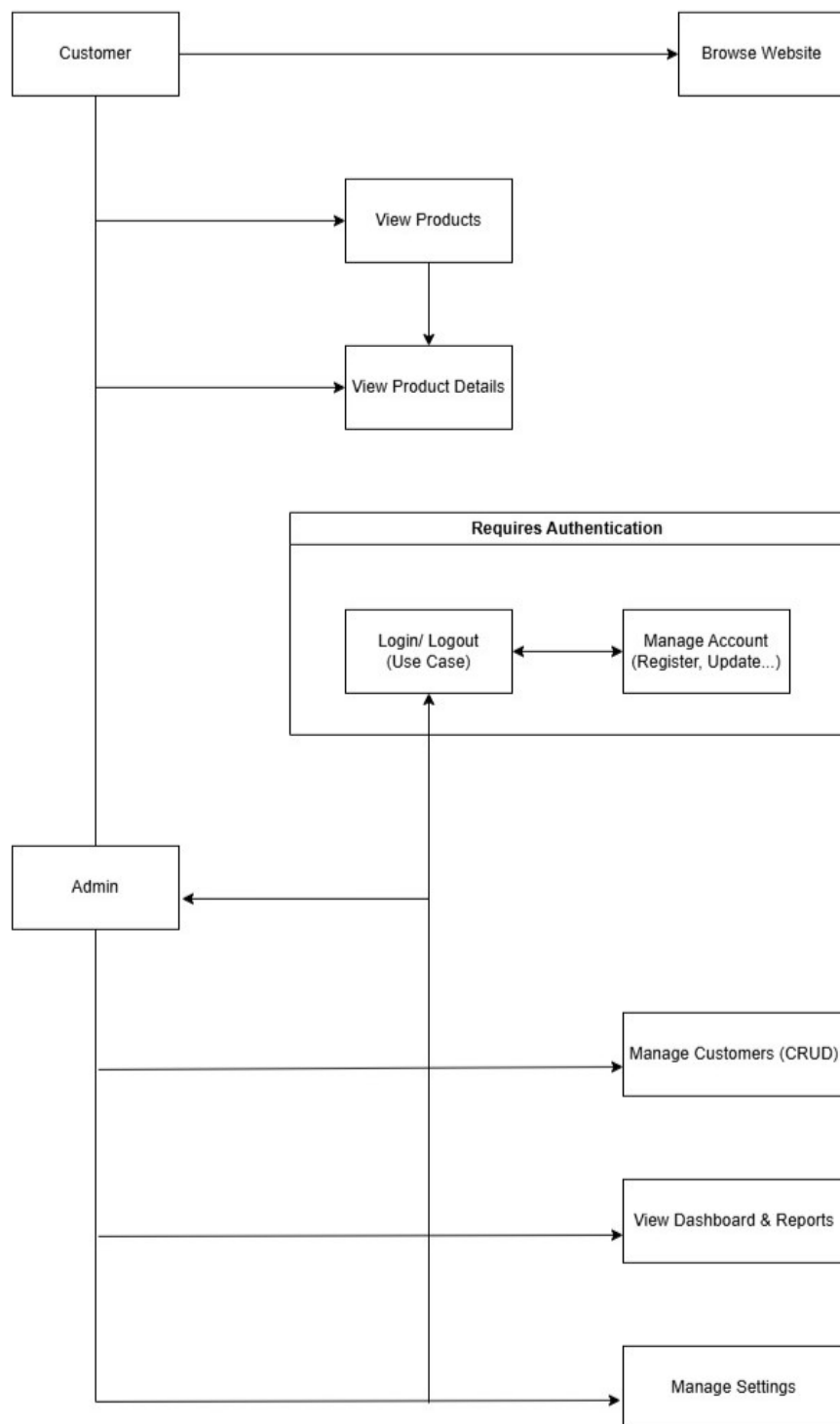
Các báo cáo/thông tin hệ thống khác: các Products (List-product, product-form-insert, product-form-update), Authorities (List-user, user-form-insert, user-form-update), Users (List-username, username-form-insert, username-form-update). Cách hoạt động như thêm, xóa, sửa thông tin đều giống như Manage Customers (nhưng thay đổi về thông tin quản lý)

2.4. Sơ đồ Use Case

Giải thích sơ đồ:

- System Boundary: Đường viền bao quanh thể hiện phạm vi của hệ thống.
- Actors: Customer và Admin nằm bên ngoài System Boundary.
- Use Cases: Các hình chữ nhật/oval bên trong System Boundary đại diện cho các chức năng.
- Associations: Đường nối liền thể hiện Actor nào có thể thực hiện Use Case nào.
- Login/ Logout: Được xem là một Use Case cơ bản mà cả Customer (khi quản lý tài khoản) và Admin đều phải thực hiện để truy cập các chức năng bảo mật. Các Use Case như Manage Account, Manage Customers, View Dashboard/ Reports, Manage Settings,... đều ngầm yêu cầu Login/ Logout (quan hệ <<include>>).
- <<extend>>: Mỗi quan hệ View Products có thể được mở rộng bởi View Product Details, nghĩa là từ việc xem danh sách, người dùng có thể chọn xem chi tiết.

Sơ đồ dưới đây mô tả mối quan hệ giữa Actors và Use Cases chính của hệ thống.



Hình 2.4. Sơ đồ UseCase

CHƯƠNG 3: TRÌNH BÀY CÔNG NGHỆ THỰC HIỆN

3.1. Giới thiệu chung

Chương này tập trung trình bày chi tiết các nền tảng công nghệ, ngôn ngữ lập trình, cơ sở dữ liệu, framework và các thư viện đã được lựa chọn và ứng dụng trong quá trình xây dựng hệ thống. Hệ thống được phát triển bao gồm hai thành phần chính: ứng dụng quản trị (Admin) và ứng dụng dành cho người dùng (Customer), đòi hỏi một tập hợp công nghệ phù hợp để đáp ứng các yêu cầu nghiệp vụ đa dạng.

Việc lựa chọn công nghệ được cân nhắc kỹ lưỡng dựa trên nhiều yếu tố quan trọng. Tính phù hợp với yêu cầu cụ thể của bài toán là tiêu chí hàng đầu, đảm bảo công nghệ có thể giải quyết tốt các vấn đề đặt ra. Bên cạnh đó, sự phổ biến và cộng đồng hỗ trợ lớn mạnh của công nghệ cũng được xem xét, giúp việc tìm kiếm tài liệu, giải quyết vấn đề và học hỏi trở nên dễ dàng hơn. Các yếu tố về hiệu năng, khả năng bảo mật và tiềm năng mở rộng của hệ thống trong tương lai cũng là những tiêu chí không thể bỏ qua khi đưa ra quyết định cuối cùng.

Về tổng thể, hệ thống được thiết kế theo kiến trúc Client-Server, trong đó người dùng tương tác với ứng dụng thông qua trình duyệt web. Cả hai ứng dụng Admin và Customer đều được xây dựng dựa trên mô hình kiến trúc phần mềm Model-View-Controller (MVC), giúp tách biệt rõ ràng giữa các thành phần xử lý logic nghiệp vụ (Model), giao diện người dùng (View) và điều khiển luồng ứng dụng (Controller). Sự tách biệt này không chỉ giúp mã nguồn trở nên có tổ chức, dễ quản lý mà còn tạo điều kiện thuận lợi cho việc bảo trì và phát triển sau này. Hai ứng dụng Admin và Customer được phát triển như hai dự án Spring Boot riêng biệt nhưng có thể chia sẻ một số thành phần chung như lớp Entity và kết nối cùng một cơ sở dữ liệu.

3.2. Công nghệ phía Backend (Backend Technologies)

Phần backend của hệ thống, nơi xử lý logic nghiệp vụ và tương tác dữ liệu, được xây dựng dựa trên các công nghệ và framework Java mạnh mẽ sau:

3.2.1. Ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình hướng đối tượng mạnh mẽ, nổi tiếng với tính ổn định, hiệu năng cao và khả năng chạy trên nhiều nền tảng (Platform Independent). Hệ sinh thái Java vô cùng phong phú với vô số thư viện và framework hỗ trợ, đặc biệt là sự tương

thích tuyệt vời với Spring Framework. Phiên bản Java được sử dụng trong dự án có thể được xác định trong tệp pom.xml.

3.2.2. Framework: Spring Boot

Spring Boot là một framework phát triển ứng dụng Java dựa trên Spring Framework, được thiết kế để đơn giản hóa quá trình khởi tạo và phát triển ứng dụng, đặc biệt là các ứng dụng web và microservices.

Các ưu điểm được tận dụng:

- **Tự động cấu hình (Auto-configuration):** Spring Boot tự động cấu hình nhiều thành phần dựa trên các thư viện có trong classpath, giảm thiểu đáng kể công việc cấu hình thủ công.
- **Starter Dependencies:** Cung cấp các gói "starter" giúp quản lý các nhóm thư viện phụ thuộc một cách dễ dàng. Các starter chính được sử dụng bao gồm:
 - spring-boot-starter-web: Hỗ trợ xây dựng ứng dụng web, bao gồm RESTful API và Spring MVC.
 - spring-boot-starter-data-jpa: Hỗ trợ truy cập dữ liệu sử dụng Java Persistence API.
 - spring-boot-starter-thymeleaf: Tích hợp template engine Thymeleaf để xử lý tầng View.
 - spring-boot-starter-security: Tích hợp Spring Security để xử lý bảo mật.
- **Embedded Server:** Tích hợp sẵn máy chủ web như Tomcat (mặc định), Jetty hoặc Undertow, cho phép chạy ứng dụng như một file JAR độc lập mà không cần triển khai lên một máy chủ web riêng biệt.
- **Quản lý dependency:** Sử dụng Maven (thông qua tệp pom.xml) để khai báo và quản lý các thư viện phụ thuộc một cách nhất quán và hiệu quả.

3.2.3. Truy cập dữ liệu (Data Access): Spring Data JPA

Spring Data JPA là một phần của Spring Data, giúp đơn giản hóa việc triển khai tầng truy cập dữ liệu dựa trên JPA (Java Persistence API). JPA là một tiêu chuẩn đặc tả kỹ thuật cho ORM (Object-Relational Mapping) trong Java.

Spring Data JPA giảm thiểu code boilerplate bằng cách tự động tạo ra các implementation cho các interface Repository (ví dụ: CustomerDAO, ProductDBDAO).

Chỉ cần định nghĩa interface kế thừa từ JpaRepository hoặc các interface cơ sở khác, Spring Data JPA sẽ cung cấp sẵn các phương thức CRUD cơ bản (save, findById, findAll, delete...) và hỗ trợ định nghĩa các phương thức truy vấn tùy chỉnh thông qua tên phương thức hoặc sử dụng annotation @Query.

Hibernate: Được sử dụng như là JPA implementation mặc định bởi Spring Boot. Hibernate thực hiện việc ánh xạ giữa các lớp Java được đánh dấu là @Entity (ví dụ: Customer.java, ProductDB.java) với các bảng tương ứng trong cơ sở dữ liệu quan hệ, giúp lập trình viên thao tác với cơ sở dữ liệu thông qua các đối tượng Java thay vì viết SQL thủ công.

3.2.4. Cơ sở dữ liệu (Database): MySQL

MySQL là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến nhất thế giới, nổi bật với tính ổn định, hiệu năng tốt, cộng đồng hỗ trợ lớn và dễ sử dụng. Nó phù hợp với nhiều loại ứng dụng web, từ nhỏ đến lớn.

Thông tin kết nối đến cơ sở dữ liệu MySQL (như URL, username, password, driver class) được khai báo trong tệp application.properties.

3.2.5. Bảo mật (Security): Spring Security

Spring Security là một framework mạnh mẽ và tùy biến cao, cung cấp các giải pháp toàn diện cho việc xác thực (Authentication - ai là người dùng?) và phân quyền (Authorization - người dùng được làm gì?) trong ứng dụng Java.

Việc cấu hình Spring Security được thực hiện chủ yếu trong lớp SecurityConfig.java. Các cấu hình chính bao gồm:

- Định nghĩa trang đăng nhập tùy chỉnh (ví dụ: /login trỏ đến login.html).
- Cấu hình các quy tắc phân quyền truy cập cho các URL khác nhau dựa trên vai trò người dùng (ví dụ: yêu cầu vai trò ADMIN để truy cập các trang trong /admin/**).
- Cấu hình cơ chế mã hóa mật khẩu (ví dụ: sử dụng BCryptPasswordEncoder để lưu trữ mật khẩu an toàn).
- Bật các cơ chế bảo vệ chống lại các tấn công phổ biến như Cross-Site Request Forgery (CSRF).

3.2.6. Công cụ hỗ trợ: Lombok

Vai trò: (Nếu có dependency lombok trong pom.xml) Project Lombok là một thư viện giúp giảm thiểu mã nguồn Java lặp đi lặp lại (boilerplate code). Bằng cách sử dụng các annotation như: `@Data`, `@Getter`, `@Setter`,... trực tiếp trên các lớp (đặc biệt là các lớp Entity), Lombok sẽ tự động sinh ra các phương thức tương ứng trong quá trình biên dịch, giúp mã nguồn ngắn gọn và dễ đọc hơn.

3.3. Công nghệ phía Frontend (Frontend Technologies)

Giao diện người dùng (UI) của hệ thống được xây dựng bằng các công nghệ và thư viện sau:

3.3.1. Ngôn ngữ Markup & Styling

HTML5: Ngôn ngữ đánh dấu siêu văn bản tiêu chuẩn để cấu trúc nội dung của các trang web.

CSS3: Ngôn ngữ định dạng dùng để mô tả cách trình bày (styling) của các phần tử HTML, bao gồm màu sắc, bố cục, font chữ, hiệu ứng...

SCSS/SASS: (Nếu có sử dụng các tệp .scss trong thư mục static/admin/scss hoặc static/default/scss) Là các bộ tiền xử lý CSS (CSS Preprocessor) mạnh mẽ, cho phép viết CSS với cú pháp nâng cao như sử dụng biến, mixin, kế thừa selector, nesting... giúp mã CSS trở nên có cấu trúc, dễ quản lý và tái sử dụng hơn. Các tệp SCSS sau đó sẽ được biên dịch thành CSS thông thường.

3.3.2. Templating Engine: Thymeleaf

Thymeleaf là một template engine phía server hiện đại cho Java, được thiết kế để xử lý và tạo ra các định dạng như HTML, XML, JavaScript, CSS và text.

Thymeleaf tích hợp rất tốt với Spring Boot, cho phép dễ dàng truy cập và hiển thị dữ liệu từ Model của Spring MVC vào trong các tệp template HTML. Nó sử dụng cú pháp "natural templating", nghĩa là các tệp template HTML vẫn có thể hiển thị đúng cấu trúc trên trình duyệt mà không cần chạy server, thuận tiện cho việc thiết kế. Thymeleaf hỗ trợ mạnh mẽ các layout dialect (ví dụ: sử dụng `layout:decorate` kết hợp với các fragment `th:insert`, `th:replace`) giúp xây dựng các layout trang dùng chung (ví dụ: `default/layout.html`, `admin/layout.html`) một cách hiệu quả.

3.3.3. Framework CSS & UI Components: Bootstrap

Bootstrap là một framework CSS/JavaScript mã nguồn mở và miễn phí, rất phổ biến để phát triển giao diện website responsive (thích ứng trên nhiều thiết bị) và mobile-first. Nó cung cấp sẵn một hệ thống lưới (grid system) mạnh mẽ, các kiểu định dạng cho typography, form, button, table, navigation, modal và nhiều thành phần giao diện khác, cùng với các plugin JavaScript tùy chọn.

Sự hiện diện của các tệp như bootstrap.min.css và bootstrap.bundle.min.js trong thư mục static/admin/vendor/bootstrap và static/default/css, static/default/js cho thấy Bootstrap đã được sử dụng để xây dựng giao diện cho cả hai phần Admin và Customer. Phiên bản cụ thể có thể được xác định qua tên tệp hoặc nội dung.

3.3.4. Thư viện JavaScript:

jQuery: Là một thư viện JavaScript nhanh, nhỏ gọn và giàu tính năng. Nó đơn giản hóa việc duyệt và thao tác tài liệu HTML (DOM traversing and manipulation), xử lý sự kiện (event handling), tạo hiệu ứng động (animation) và thực hiện các yêu cầu Ajax. Sự có mặt của jquery.min.js (hoặc tương tự) trong static/admin/vendor/jquery và có thể được nhúng trong các layout cho thấy jQuery được sử dụng làm nền tảng cho nhiều tương tác phía client.

DataTables: (Chủ yếu trong phần Admin) Là một plugin mạnh mẽ cho thư viện jQuery, dùng để tăng cường chức năng cho các bảng HTML. Nó cung cấp các tính năng như phân trang (pagination), tìm kiếm tức thời (instant search), sắp xếp theo nhiều cột (multi-column ordering) và dễ dàng tùy biến. Các tệp như jquery.dataTables.min.js,.dataTables.bootstrap4.min.js,.dataTables.bootstrap4.min.css và tệp khởi tạo datatables-demo.js trong static/admin/vendor/datatables và static/admin/js/demo cho thấy thư viện này được dùng để hiển thị dữ liệu dạng bảng trong trang Admin.

Chart.js: (Chủ yếu trong phần Admin) Là một thư viện JavaScript đơn giản nhưng linh hoạt để vẽ các loại biểu đồ đẹp mắt (như biểu đồ đường, cột, tròn...) bằng cách sử dụng thẻ <canvas> của HTML5. Các tệp Chart.min.js, Chart.bundle.min.js và các tệp demo (chart-area-demo.js, chart-pie-demo.js, chart-bar-demo.js),... để chỉ ra việc sử dụng thư viện này để hiển thị dữ liệu trực quan trên dashboard của trang Admin.

Font Awesome: (Nếu có thư mục fontawesome-free trong static/admin/vendor) Là một bộ công cụ cung cấp các biểu tượng vector (vector icons) và logo phổ biến, dễ dàng tùy chỉnh kích thước, màu sắc thông qua CSS.

Custom JavaScript: Bên cạnh các thư viện phổ biến, dự án còn sử dụng các tệp JavaScript tùy chỉnh (như custom.js, scripts.js, loadContent.js...) để triển khai các logic xử lý sự kiện, tương tác người dùng hoặc các hiệu ứng đặc thù riêng của ứng dụng.

3.4. Công cụ phát triển và quản lý (Development and Management Tools)

Quá trình phát triển và quản lý dự án được hỗ trợ bởi các công cụ sau:

3.4.1. Môi trường phát triển tích hợp (IDE - Integrated Development Environment)

Các IDE phổ biến như IntelliJ IDEA, Eclipse hoặc Visual Studio Code thường được sử dụng để viết mã, gỡ lỗi và quản lý dự án Java/Spring Boot. Các IDE này cung cấp nhiều tính năng hỗ trợ như gợi ý mã, kiểm tra lỗi cú pháp, tích hợp với các công cụ build và hệ thống quản lý phiên bản.

3.4.2. Công cụ Build và Quản lý Dependency: Maven

Maven là một công cụ quản lý và tự động hóa quy trình build dự án mạnh mẽ. Nó quản lý các thư viện phụ thuộc của dự án (dependencies), biên dịch mã nguồn, chạy kiểm thử, đóng gói ứng dụng (thành file JAR hoặc WAR) và thực hiện nhiều tác vụ khác trong vòng đời build. Tất cả cấu hình của Maven được định nghĩa trong tệp pom.xml.

3.4.3. Hệ quản trị cơ sở dữ liệu (DBMS Tool)

Các công cụ như MySQL Workbench, DBeaver, hoặc phpMyAdmin thường được sử dụng để thiết kế, quản lý cơ sở dữ liệu MySQL, thực thi các câu lệnh SQL và xem dữ liệu.

3.4.4. Hệ thống quản lý phiên bản (Version Control System - VCS): Git

Git là hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay, giúp theo dõi các thay đổi trong mã nguồn, quản lý các nhánh phát triển (branches), phối hợp làm việc nhóm và quay lại các phiên bản trước đó khi cần. Mã nguồn dự án thường được lưu trữ trên các nền tảng như GitHub, GitLab hoặc Bitbucket.

3.4.5. Máy chủ ứng dụng (Application Server): Embedded Tomcat

Nhờ Spring Boot, ứng dụng được đóng gói kèm với một máy chủ web nhúng (mặc định là Tomcat). Điều này cho phép ứng dụng tự chạy độc lập mà không cần cài đặt và cấu hình một máy chủ Tomcat riêng biệt, đơn giản hóa quá trình phát triển và triển khai.

3.5. Tổng kết lý thuyết

Chương này đã trình bày một cách tổng quan và chi tiết về các công nghệ, framework, thư viện và công cụ đã được lựa chọn và sử dụng để xây dựng hệ thống ứng dụng web bao gồm phần quản trị (Admin) và phần người dùng (Customer). Các công nghệ phía backend chủ đạo bao gồm Java, Spring Boot, Spring Data JPA, Hibernate, MySQL và Spring Security, tạo nên một nền tảng vững chắc cho việc xử lý logic nghiệp vụ và quản lý dữ liệu.

P phía frontend, sự kết hợp giữa HTML5, CSS3, Thymeleaf, Bootstrap, jQuery và các thư viện JavaScript chuyên dụng khác như DataTables và Chart.js đã giúp xây dựng giao diện người dùng chức năng, thân thiện và đáp ứng tốt trên nhiều thiết bị. Các công cụ như Maven, Git và IDE hiện đại đóng vai trò quan trọng trong việc hỗ trợ quá trình phát triển diễn ra hiệu quả và có tổ chức.

Sự lựa chọn và phối hợp các công nghệ này được đánh giá là phù hợp với quy mô và yêu cầu của đề tài, tận dụng được ưu điểm của từng công nghệ để tạo ra một hệ thống hoạt động ổn định, dễ bảo trì và có khả năng mở rộng. Mối liên kết chặt chẽ giữa các thành phần, ví dụ như Spring Boot điều phối hoạt động, Spring Data JPA làm cầu nối với MySQL, Spring Security đảm bảo an toàn, và Thymeleaf cùng Bootstrap hiển thị dữ liệu ra giao diện, đã tạo nên một giải pháp công nghệ hoàn chỉnh cho hệ thống.

CHƯƠNG 4: TRÌNH BÀY PROJECT

4.1. Giới thiệu

Chương này trình bày tổng quan về ứng dụng web đã được xây dựng hoàn chỉnh, bao gồm hai thành phần chính: giao diện dành cho Khách hàng (Customer Frontend) và hệ thống quản trị dành cho Quản trị viên (Admin Backend). Dự án được phát triển bằng công nghệ Java Spring Boot, sử dụng Thymeleaf cho tầng giao diện, Spring Security để phân quyền và Hibernate/JPA để tương tác cơ sở dữ liệu. Phần cơ sở dữ liệu đóng vai trò lưu trữ thông tin người dùng, khách hàng và quyền truy cập

4.2. Cơ sở dữ liệu

4.2.1. Table Users

```
CREATE TABLE `users` (  
  `username` varchar(50) NOT NULL,  
  `password` varchar(100) NOT NULL,  
  `enabled` tinyint(4) NOT NULL  
)  
  
INSERT INTO `users` (`username`, `password`, `enabled`) VALUES  
(  
  ('hung', '$2a$12$9Ewqv90SvoNi0mkWDGiJoeainfM09OrgJ47a3cyO2Gpf8tinzJ0gK',  
    1),  
  ('Lan',  
    '$2a$10$KomfbYJKLQF4ZXTlwNzBT.yFXWk4eAe16gKNet75nK3CYa8d7rpL.', 1),  
  ('tru', '$2a$12$11fmFFTG7x9wU2yIExaj3et31m.flPY8YeGMifD5zjbVw3hiIYoc2',  
    1),  
  ('truong',  
    '$2a$12$ROki/QX8tZD0rvbW72QOk.uxk7vOiU1uO8PtyUNWe4YeyV1M8uWNm',  
    1);  
  
ALTER TABLE `users` ADD PRIMARY KEY (`username`);
```

4.2.2. Table Customer

```
CREATE TABLE `customer` (  
  `id` int(11) NOT NULL,  
  `name_company` varchar(255) NOT NULL,  
  `street_address` varchar(255) NOT NULL,  
  `city` varchar(50) NOT NULL,  
  `region` varchar(20) NOT NULL,  
  `post_code` char(5) NOT NULL,  
  `country` varchar(20) NOT NULL  
)  
  
INSERT INTO `customer` (`id`, `name_company`, `street_address`, `city`, `region`,  
  `post_code`, `country`) VALUES  
(1, 'Nguyễn Văn A', '123 Đường A', 'Hà Nội', 'Miền Bắc', '10000', 'Việt Nam'),  
(2, 'Trần Thị B', '456 Đường B', 'TP.HCM', 'Miền Nam', '70000', 'Việt Nam'),  
(3, 'Lê Văn C', '789 Đường C', 'Đà Nẵng', 'Miền Trung', '55000', 'Việt Nam'),  
(4, 'Phạm Thị D', '101 Đường D', 'Hải Phòng', 'Miền Bắc', '18000', 'Việt Nam'),  
(5, 'Hoàng Văn E', '202 Đường E', 'Cần Thơ', 'Miền Tây', '90000', 'Việt Nam'),  
(6, 'van', '123 thđ', 'Ho Chi Minh', 'Vietnam', '123vn', 'VietNam'),  
(7, 'e1T', '123 thđ', 'Ha Noi', 'Vie', 'E1TVN', 'Vietnam'),  
(8, 'VNG', '12 VNG', 'HCM', 'VietNam', '12VNG', 'Vietnam'),  
(9, 'eIT', '123 thđ', 'Ha Noi', 'Viet Nam', '123IT', 'Vietnam'),  
(10, 'AWS', '12 Silicon Valley', 'Texas', 'Viet Nam', 'AWS35', 'USA'),  
(11, 'ShopCom', 'Phan Van Tri', 'Ho Chi Minh', 'Viet Nam', 'E1TSC', 'Vietnam'),  
(12, 'Nguyen Van A', '123 Duong A', 'Ha Noi', 'VietNam', '101vA', 'Vietnam');  
  
ALTER TABLE `customer` ADD PRIMARY KEY (`id`);
```

4.2.3. Table Authorities

```
CREATE TABLE `authorities` (  
    `id` int(10) NOT NULL,  
    `username` varchar(50) NOT NULL,  
    `authority` varchar(50) NOT NULL  
)  
  
INSERT INTO `authorities` (`id`, `username`, `authority`) VALUES  
  
(1, 'hung', 'ROLE_ADMIN'),  
(2, 'hung', 'ROLE_MANAGER'),  
(3, 'hung', 'ROLE_STUDENT'),  
(4, 'tru', 'ROLE_STUDENT'),  
(5, 'truong', 'ROLE_MANAGER'),  
(6, 'truong', 'ROLE_STUDENT');  
  
ALTER TABLE `authorities` ADD PRIMARY KEY (`id`);
```

4.2.4. Table Product

```
CREATE TABLE `product` (  
    `id` int(11) NOT NULL,  
    `name` varchar(255) DEFAULT NULL,  
    `description` text DEFAULT NULL,  
    `price` decimal(10,2) DEFAULT NULL,  
    `image_url` varchar(1024) DEFAULT NULL  
)  
  
INSERT INTO `product` (`id`, `name`, `description`, `price`, `image_url`) VALUES  
  
(1, 'Peanut Butter', 'Bơ đậu phộng béo ngậy với hương vị tuyệt hảo', 45.00,  
'https://food.fnr.sndimg.com/content/dam/images/food/fullset/2007/10/15/0/EA1112_Peanut_Butter.jpg.rend.hgtvcom.1280.1280.suffix/1382375731291.webp'),
```

(2, 'Pumpkin Pie', 'Bánh bí ngô truyền thống thơm ngon', 35.00,
'https://assets.tmeccosys.com/image/upload/t_web767x639/img/recipe/ras/Assets/e54fa10d818b5a75debc944b79bc6e7e/Derivates/eccbf500e6e1e8f9dc4a69adf311500746fa593d.jpg'),

(3, 'Jambalaya', 'Món ăn cay nóng kiểu New Orleans', 50.00,
'https://assets.tmeccosys.com/image/upload/t_web767x639/img/recipe/ras/Assets/e0c688bb1afa8f4b7e6912bca2c3b3fc/Derivates/a72cefbf839e118d5f4ef36f15a1b06d6b7e1720.jpg'),

(4, 'Pizza', 'Delicious Italy Pizza', 9.99, 'https://daylambanh.edu.vn/wp-content/uploads/2024/04/cach-lam-banh-pizza.jpg');

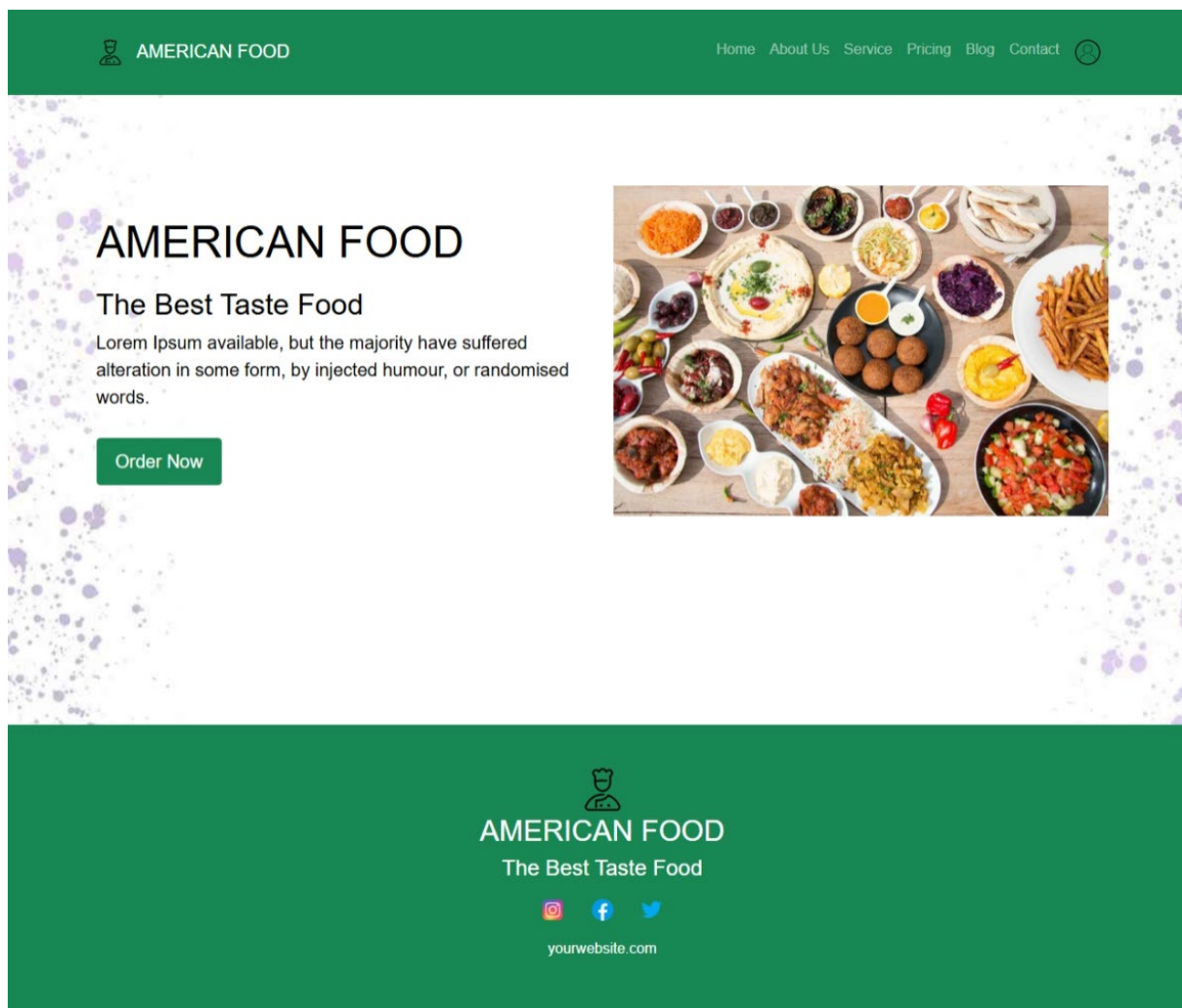
ALTER TABLE `product` ADD PRIMARY KEY (`id`);

4.3. Giao diện Khách hàng (Customer Frontend)

Phần giao diện khách hàng được xây dựng với mục tiêu cung cấp trải nghiệm thân thiện, dễ sử dụng, cho phép người dùng tương tác với các dịch vụ/sản phẩm của hệ thống.

4.3.1. Trang chủ

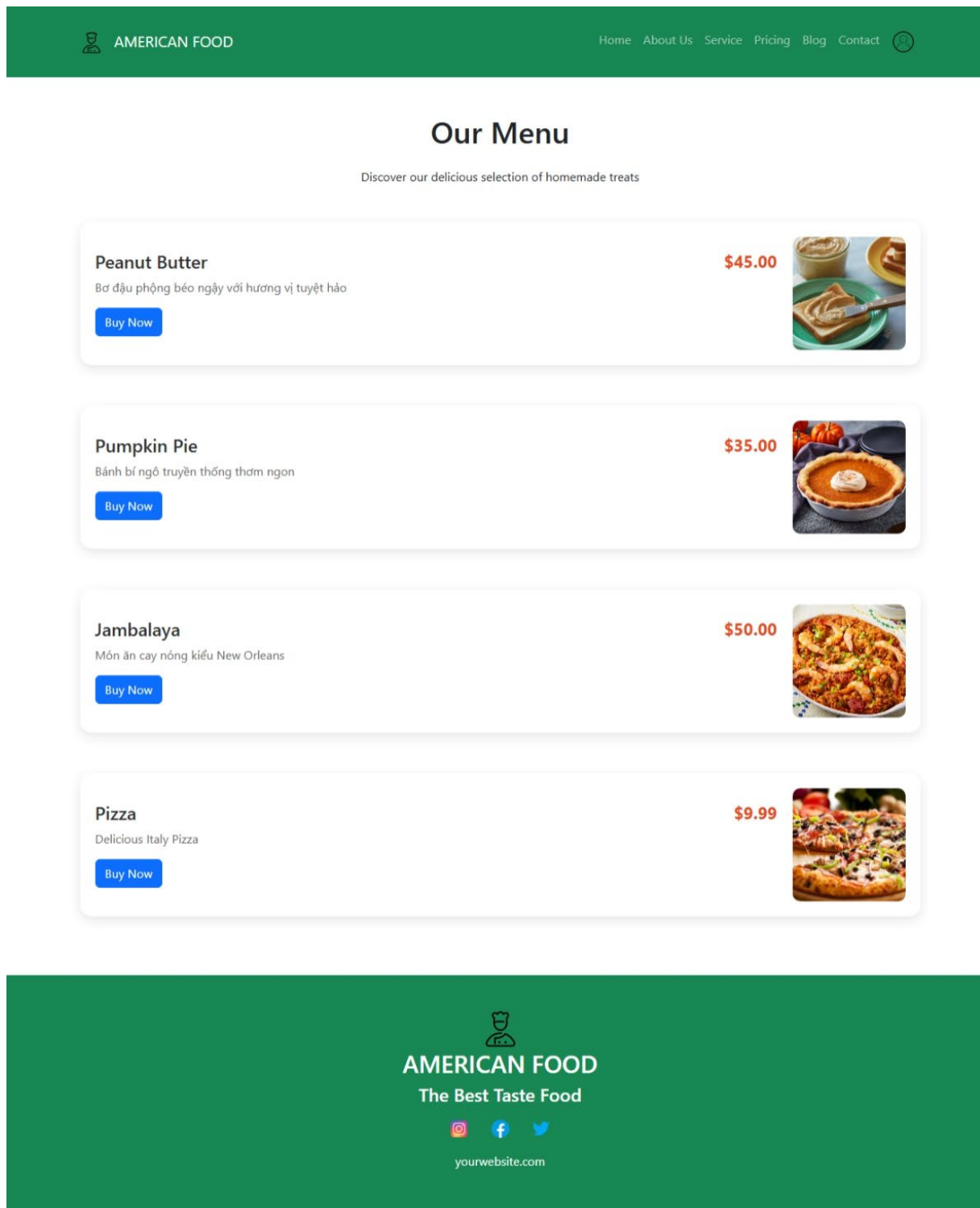
Là trang đầu tiên người dùng thấy khi truy cập. Thường hiển thị các thông tin tổng quan, sản phẩm nổi bật, tin tức mới hoặc các banner quảng cáo. Giao diện sử dụng template từ Final/customer/src/main/resources/templates/default/layout.html và các file CSS/JS trong Final/customer/src/main/resources/static/default/.



Hình 4.3.1. Trang chủ của Customer

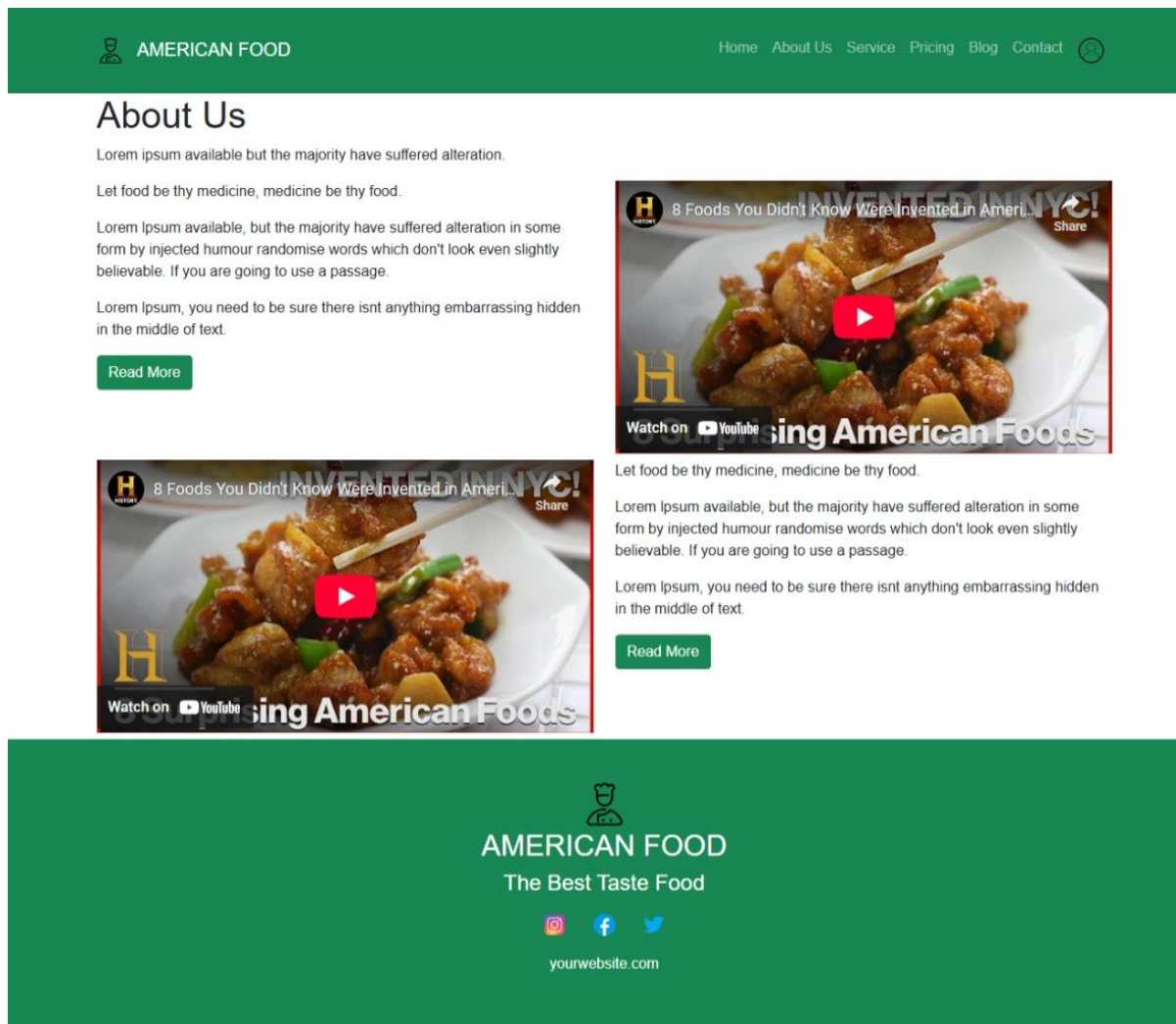
4.3.2. Các trang khác

Hiển thị danh sách các sản phẩm hoặc dịch vụ mà hệ thống cung cấp. Người dùng có thể xem thông tin cơ bản và hình ảnh.

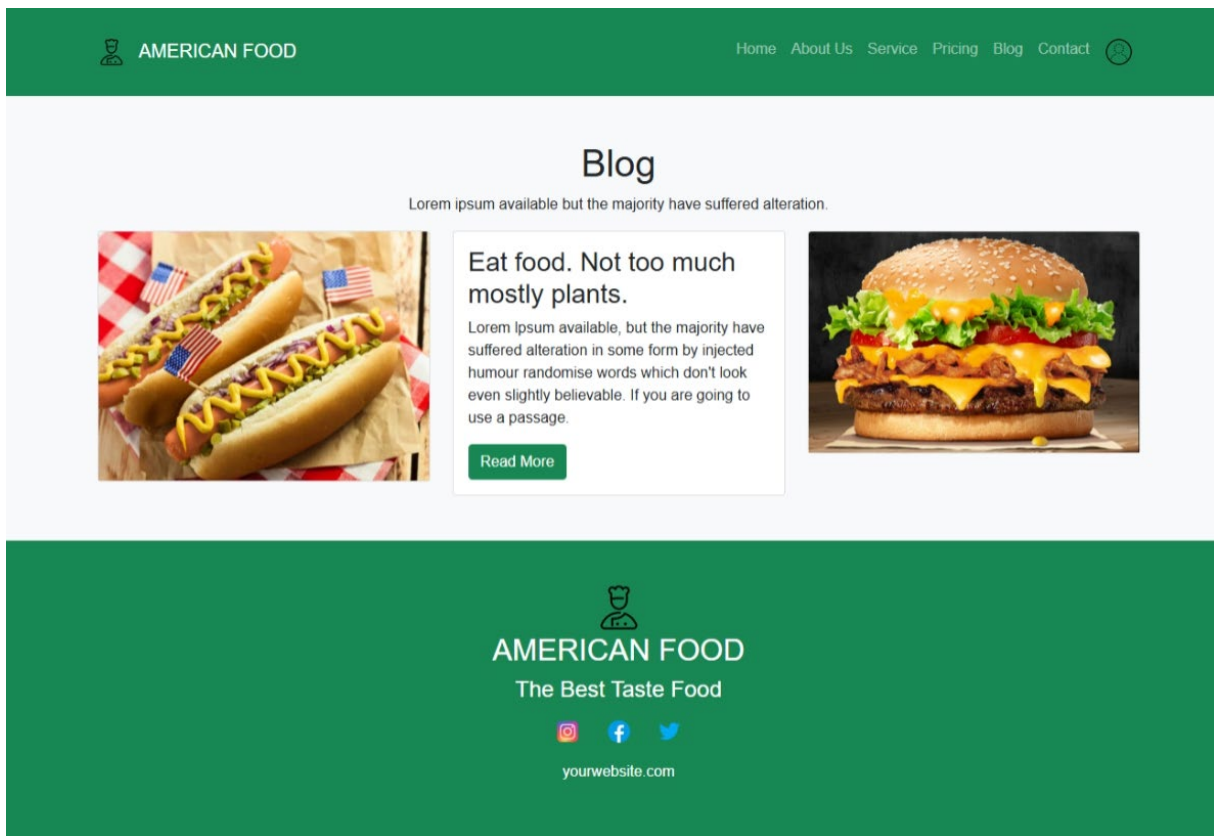


Hình 4.3.2.1. Trang Services

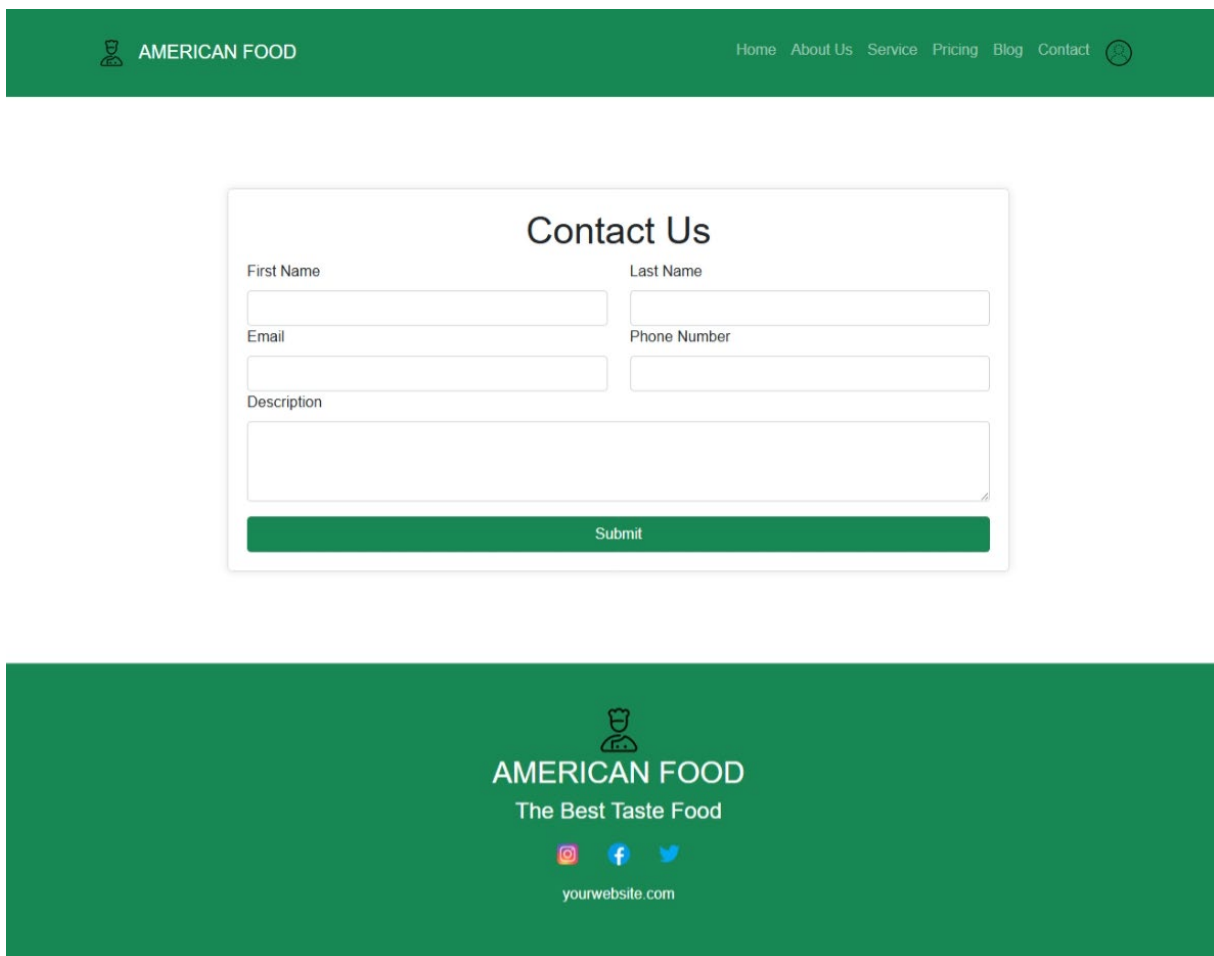
Ngoài ra còn có các trang gồm Giới thiệu (about-us.html), Liên hệ (contact-us.html), Bảng giá (pricing.html), Blog (blog.html), cung cấp các thông tin hỗ trợ cho người dùng. Giao diện các trang này thường đồng bộ với trang chủ.



Hình 4.3.2.2. Trang About Us



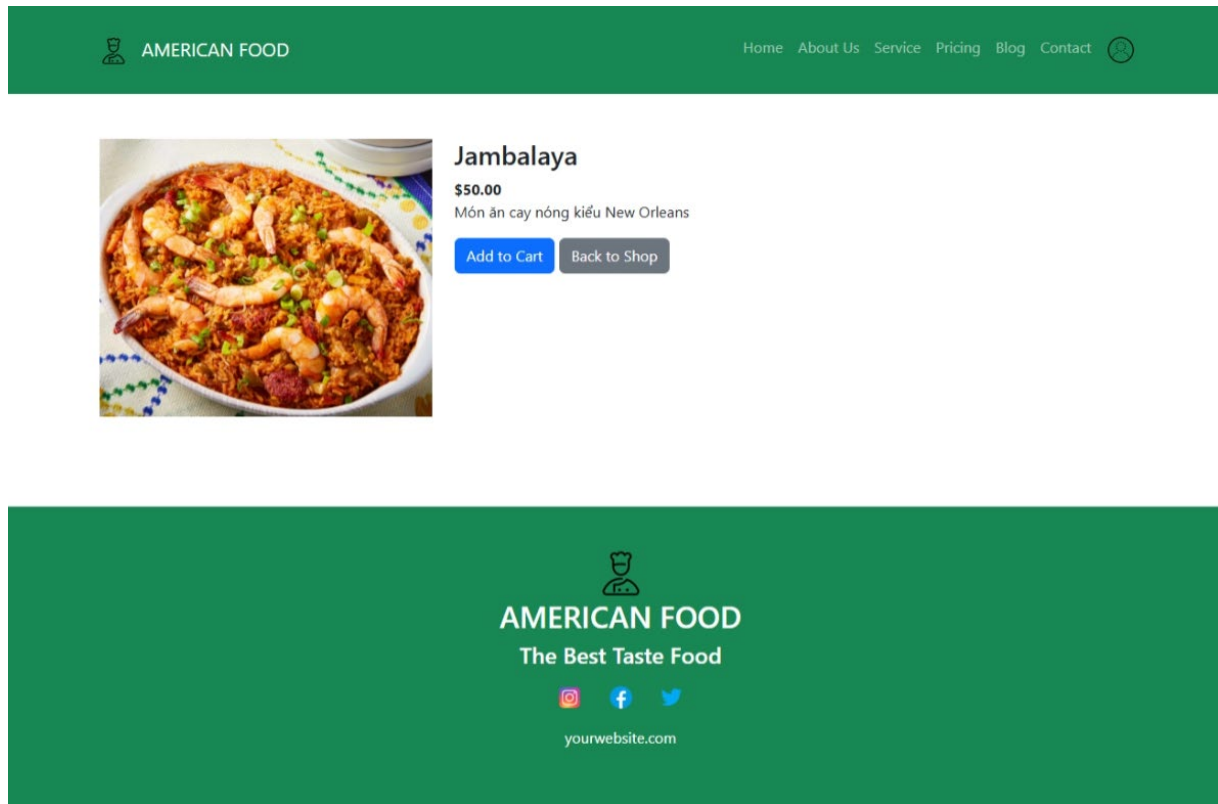
Hình 4.3.2.3. Trang Blog



Hình 4.3.2.4. Trang Contact

4.3.3. Trang Chi tiết Sản phẩm

Khi người dùng nhấp vào một sản phẩm, trang này sẽ hiển thị thông tin chi tiết hơn, bao gồm mô tả, hình ảnh lớn, giá cả (nếu có), và các thông tin liên quan khác.



Hình 4.3.3. Trang Chi tiết sản phẩm

4.3.4. Trang Đăng nhập

Form cho phép khách hàng đã có tài khoản đăng nhập vào hệ thống để truy cập các chức năng cá nhân (nếu có, ví dụ: quản lý đơn hàng, thông tin cá nhân).

Login Page

User

Password

login

Hình 4.3.4. Trang Login

4.4. Hệ thống Quản trị

Phần quản trị được thiết kế để Quản trị viên (Admin) có thể quản lý các khía cạnh khác nhau của hệ thống một cách hiệu quả. Giao diện quản trị thường có cấu trúc dashboard với menu điều hướng bên trái hoặc trên cùng. Giao diện sử dụng template từ Final/admin/src/main/resources/templates/admin/layout.html và các file CSS/JS trong Final/admin/src/main/resources/static/admin/.

4.4.1. Trang Đăng nhập

Form đăng nhập riêng dành cho Admin để truy cập vào khu vực quản trị.

Login Page

User

Password

login

Hình 4.4.1.1. Trang Login

Trong trường hợp user đó không có vai trò (Role) Admin mà cố tình vào thì sẽ bị chặn trang như vậy

Access Denied Page

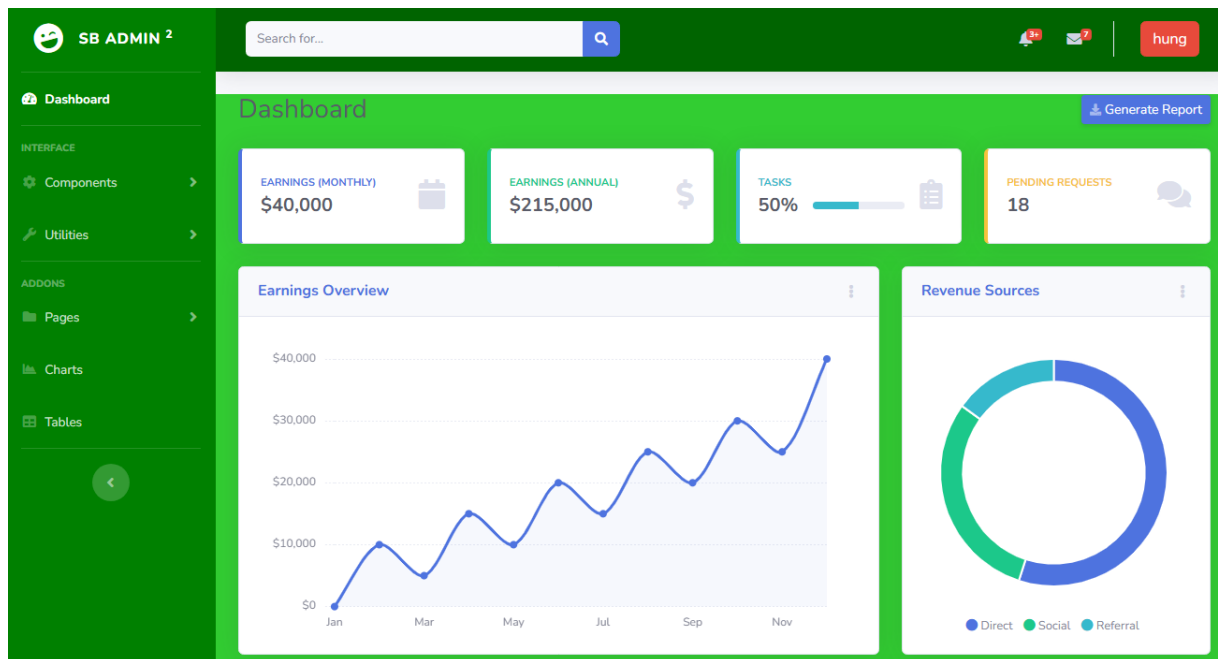
Cannot Access

Back Home

Hình 4.4.1.2. Trang Chặn thâm nhập

4.4.2. Trang Admin

Trang tổng quan sau khi Admin đăng nhập thành công. Thường hiển thị các số liệu thống kê nhanh, biểu đồ, hoặc các thông báo quan trọng.



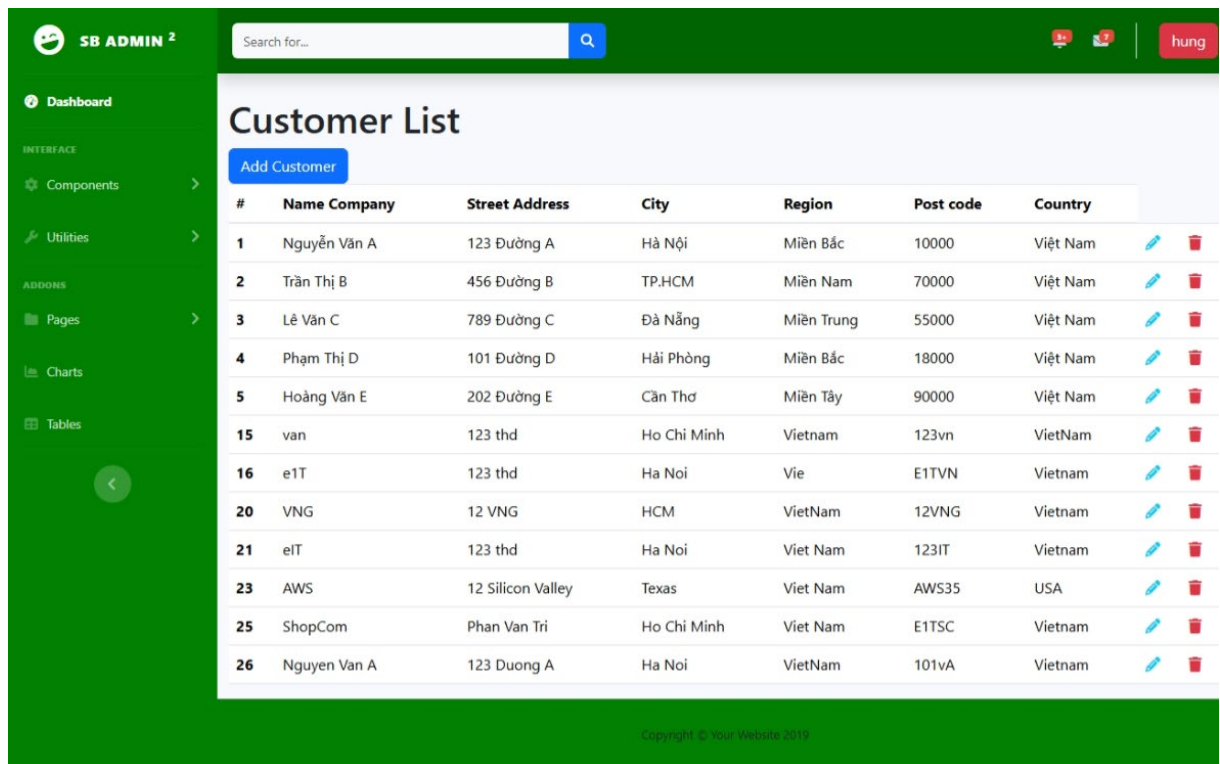
Hình 4.4.2. Trang Admin

4.4.3. Các Trang Quản lý/ báo cáo khác

Bao gồm các trang hiển thị dữ liệu dưới dạng sản phẩm (list-product), thông tin khách hàng (list-customer), quản lý User (Users), quản lý Vai trò (Authorities)

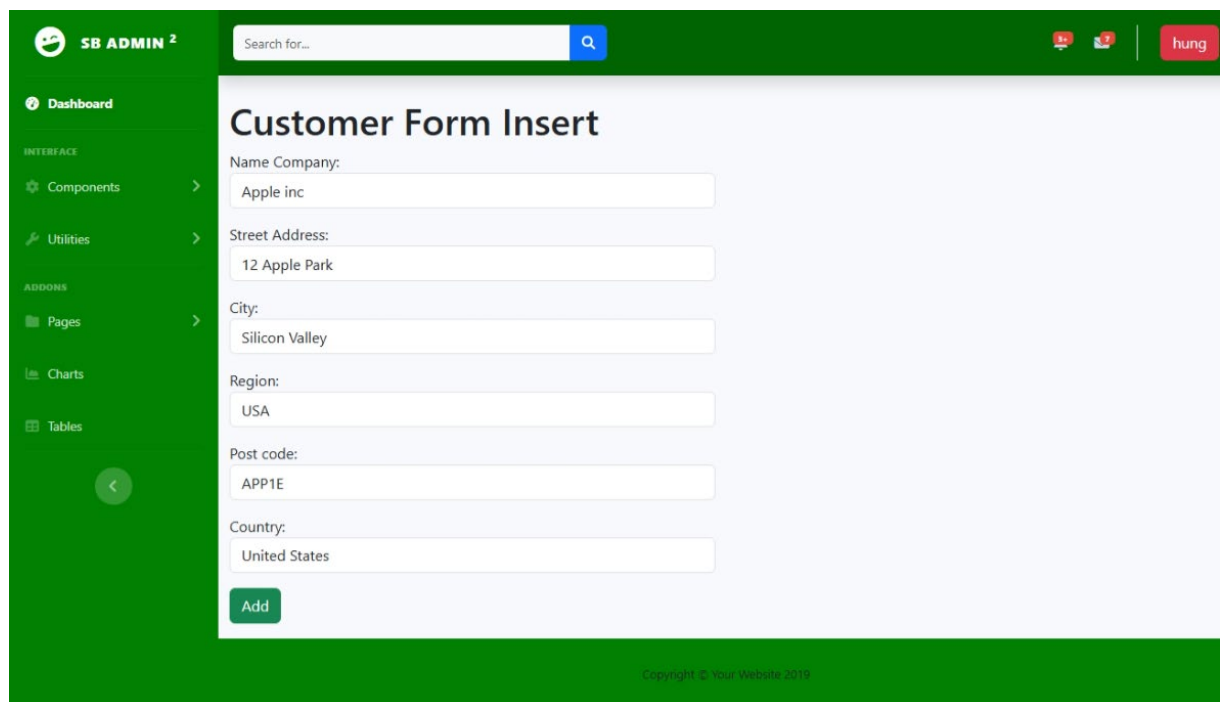
a). Quản lý Khách hàng

Chức năng cốt lõi cho phép Admin xem danh sách tất cả khách hàng, thêm khách hàng mới, chỉnh sửa thông tin khách hàng hiện có và xóa khách hàng (thông qua action trong controller). Giao diện thường là dạng bảng dữ liệu với các nút chức năng tương ứng.



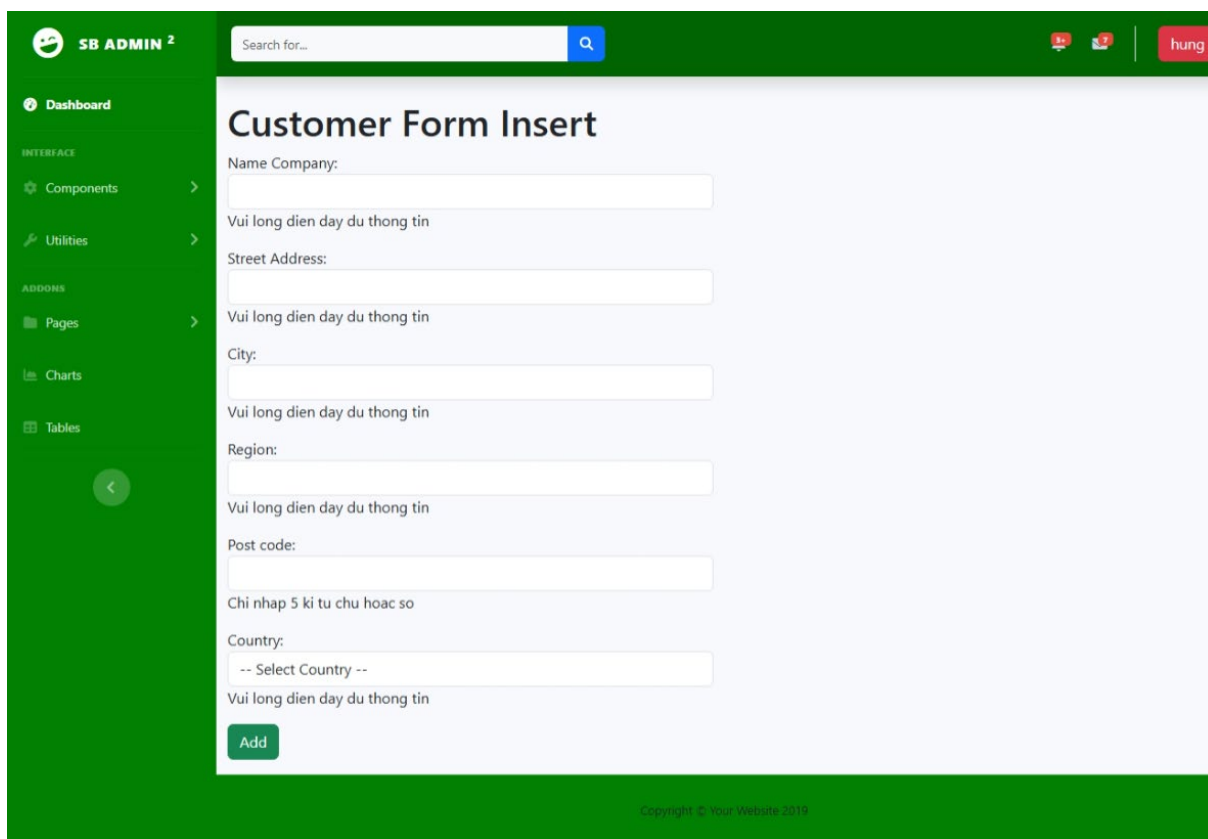
Hình 4.4.3.1. Trang list-customer

Khi nhấn button Add Customer thì sẽ chuyển sang trang Customer-form-insert



Hình 4.4.3.2. Trang customer-form-insert

Trong trường hợp nếu không nhập đầy đủ thông tin, hoặc nhập sai thông tin thì sẽ hiện cảnh báo dưới Label

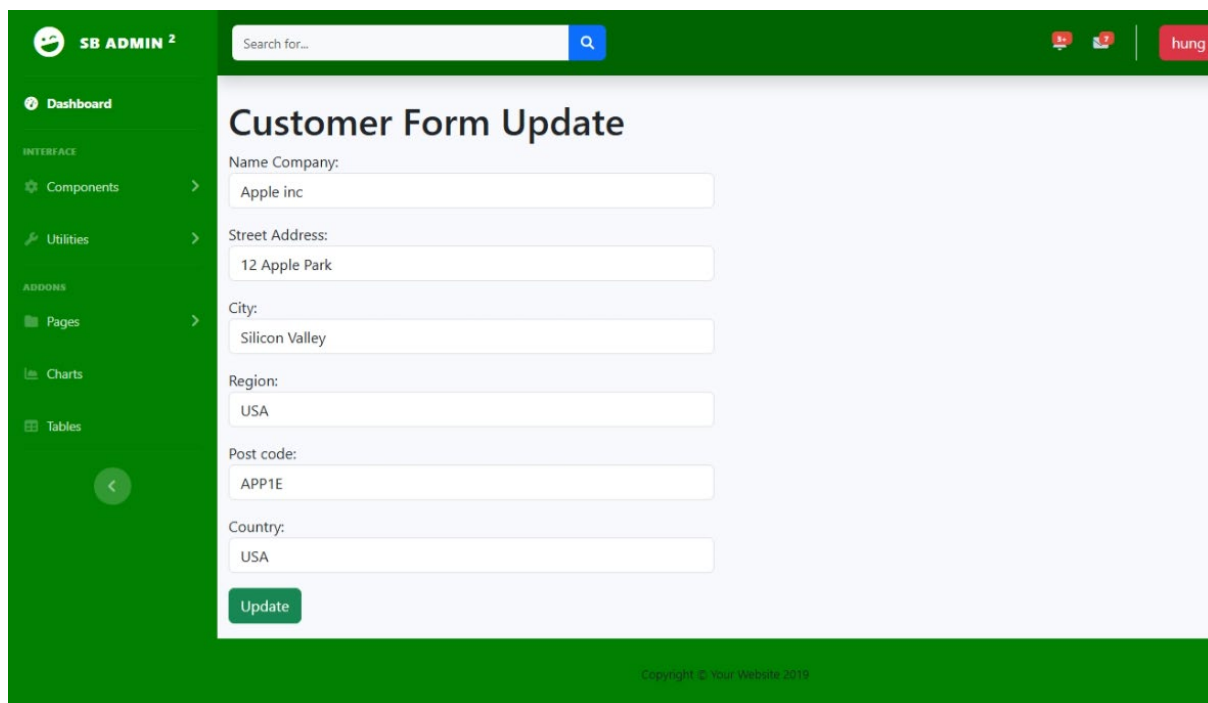


The screenshot shows the 'Customer Form Insert' page in the SB ADMIN 2 interface. The left sidebar contains a menu with 'Dashboard', 'INTERFACE' (Components, Utilities), and 'ADDONS' (Pages, Charts, Tables). The main content area has a search bar and a user profile 'hung'. The form fields are:

- Name Company:
- Street Address:
- City:
- Region:
- Post code:
- Country:

Each field has a placeholder message 'Vui lòng điền đầy đủ thông tin' (Please fill in the information completely). A green 'Add' button is at the bottom. The footer says 'Copyright © Your Website 2019'.

Hình 4.4.3.3. Trang thông báo điền thiếu thông tin trong Customer-form-insert



The screenshot shows the 'Customer Form Update' page in the SB ADMIN 2 interface. The left sidebar contains a menu with 'Dashboard', 'INTERFACE' (Components, Utilities), and 'ADDONS' (Pages, Charts, Tables). The main content area has a search bar and a user profile 'hung'. The form fields are:

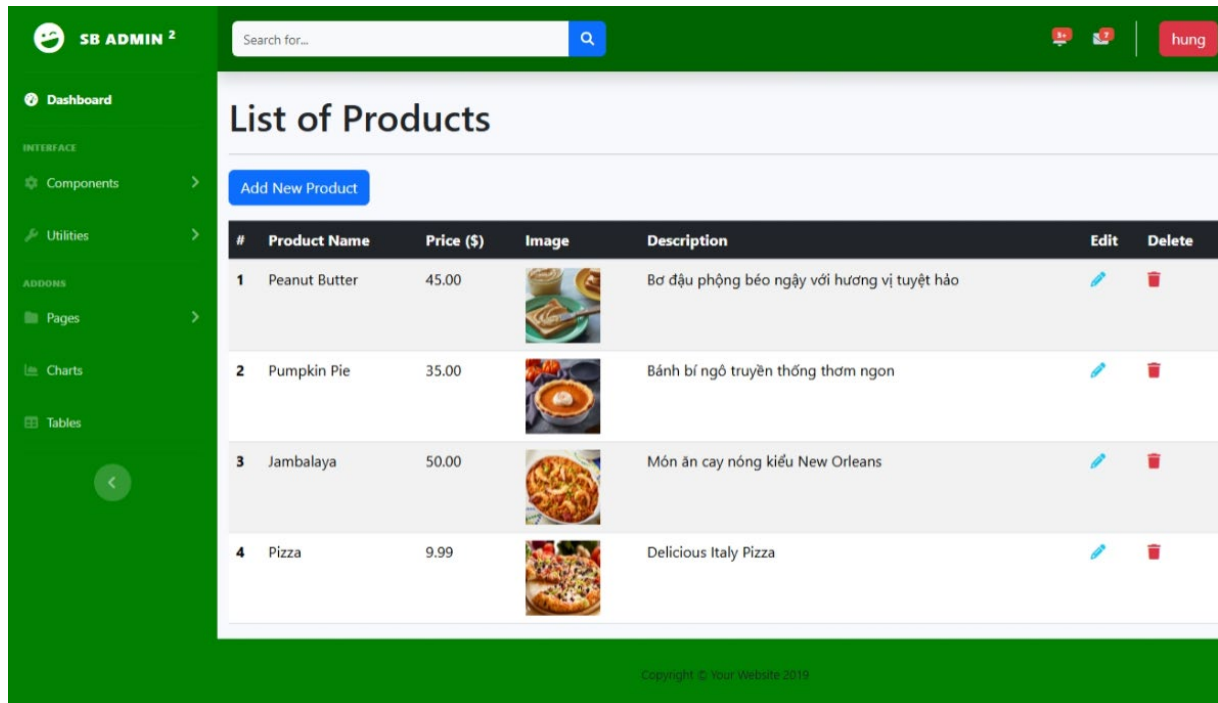
- Name Company:
- Street Address:
- City:
- Region:
- Post code:
- Country:

Each field has a placeholder message 'Vui lòng điền đầy đủ thông tin' (Please fill in the information completely). A green 'Update' button is at the bottom. The footer says 'Copyright © Your Website 2019'.

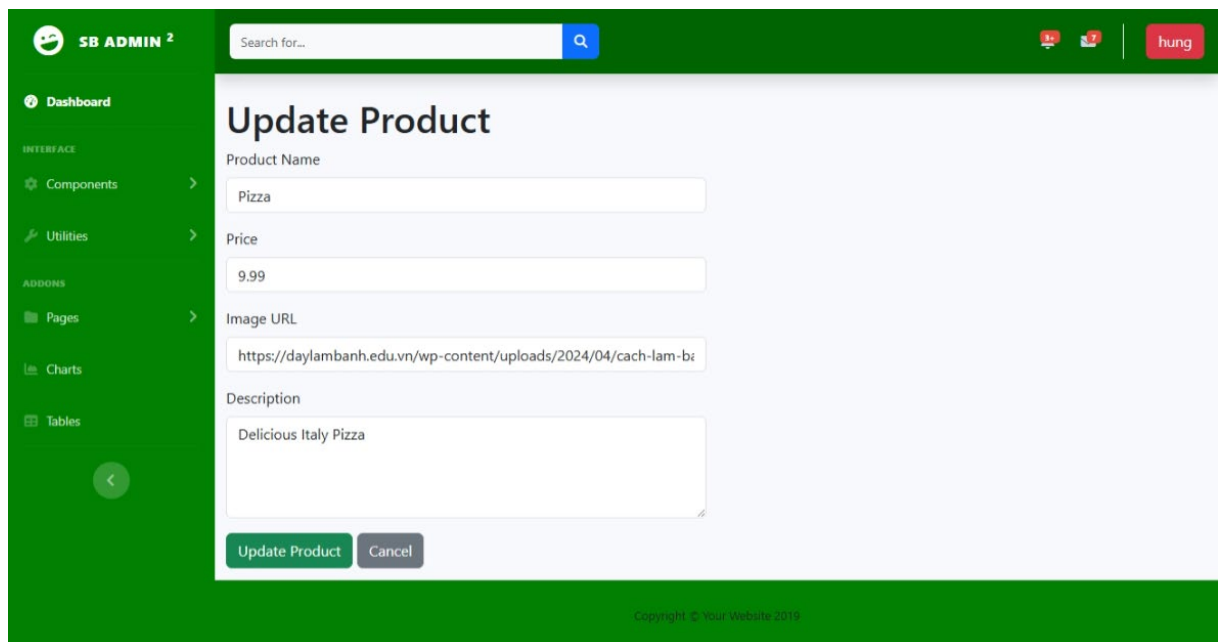
Hình 4.4.3.4. Trang Customer-form-update

b) Quản lý sản phẩm

Chức năng cốt lõi cho phép Admin xem danh sách tất cả món ăn, thêm món ăn mới, chỉnh sửa thông tin món ăn hiện có và xóa món ăn (thông qua action trong controller). Giao diện thường là dạng bảng dữ liệu với các nút chức năng tương ứng.



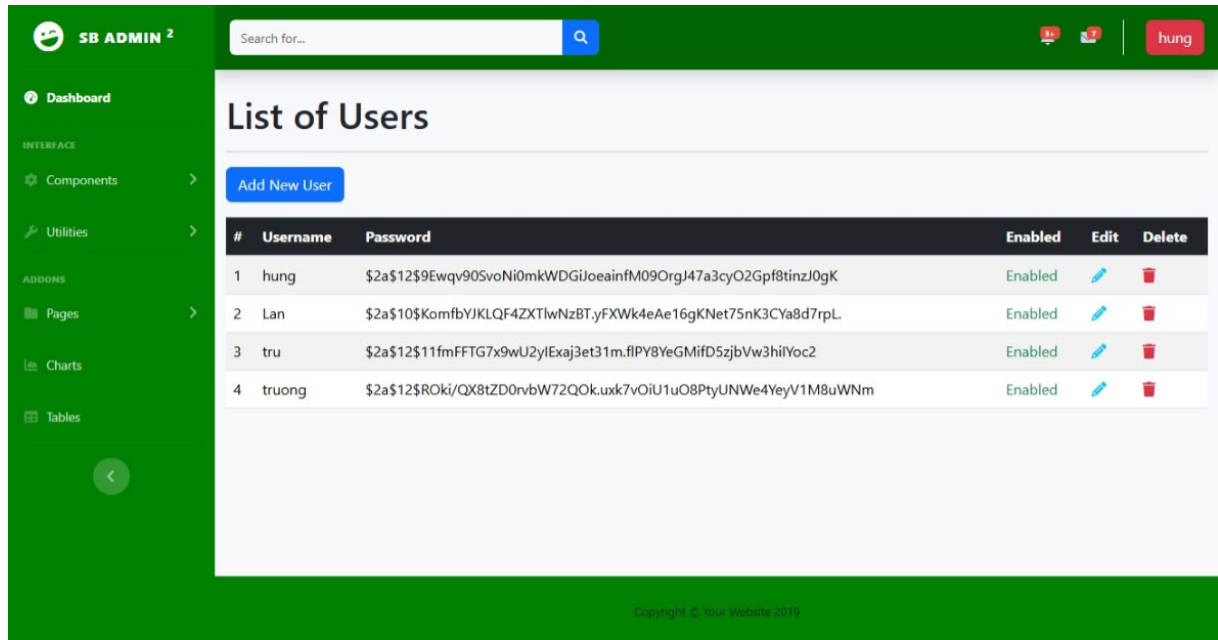
Hình 4.4.3.5. Trang list-product



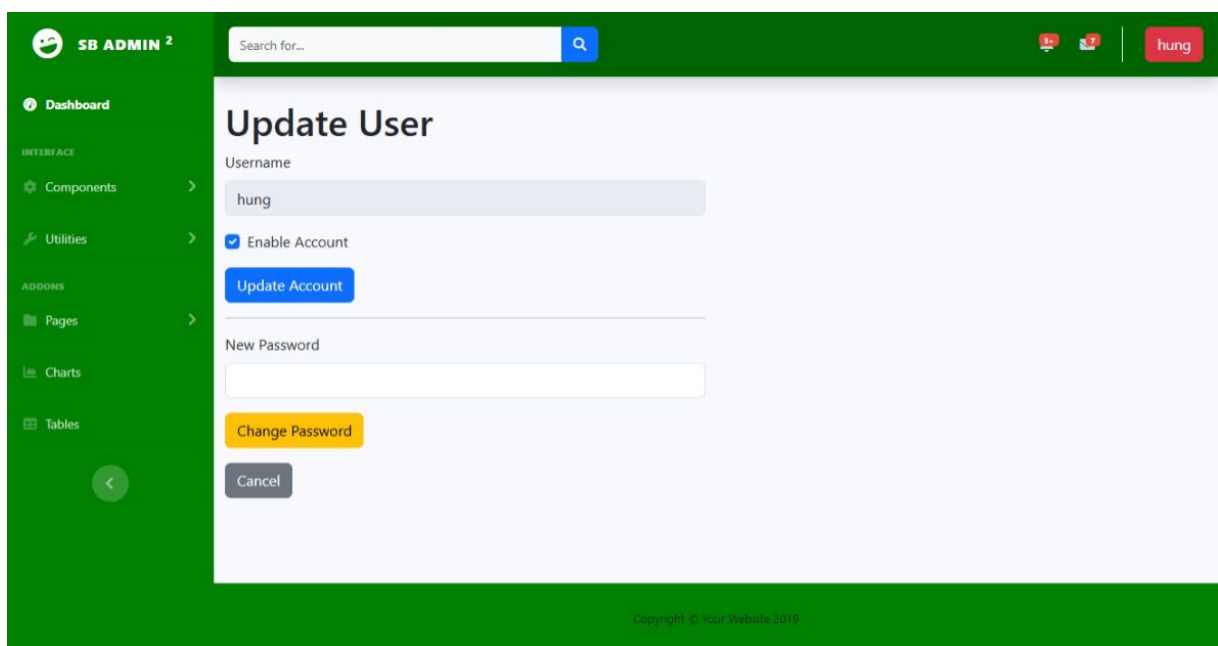
Hình 4.4.3.6. Trang product-form-update

c) Quản lý người dung (Users)

Chức năng cốt lõi cho phép Admin xem danh sách tất cả Users, thêm Users mới, chỉnh sửa thông tin Users hiện có và xóa Users (thông qua action trong controller). Giao diện thường là dạng bảng dữ liệu với các nút chức năng tương ứng



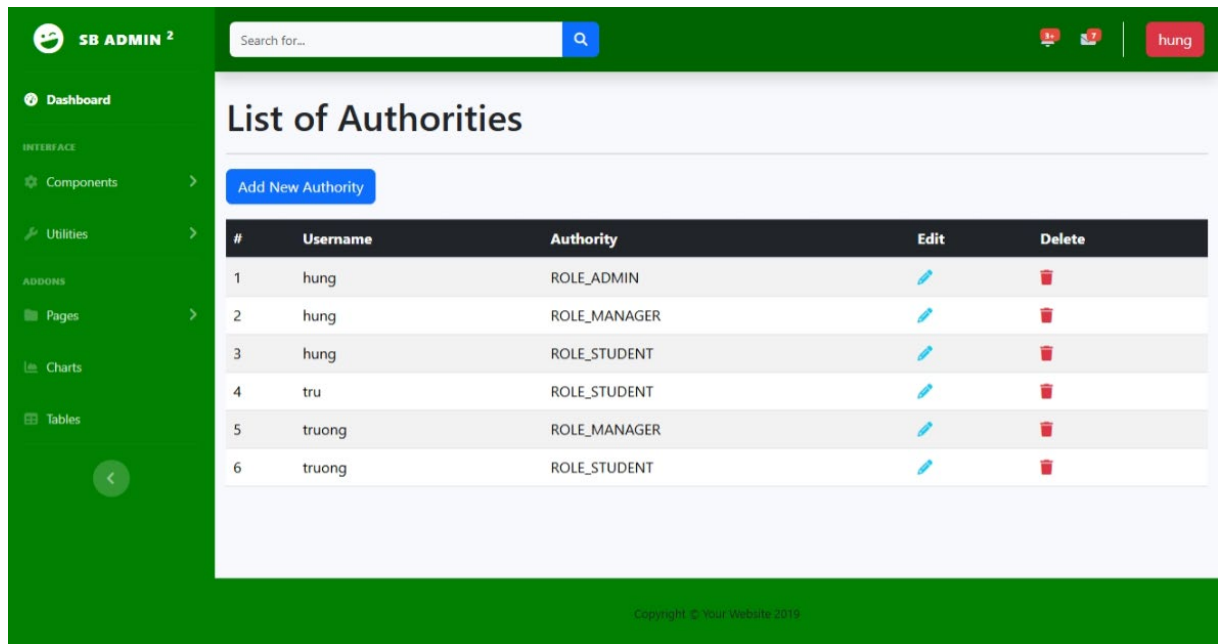
Hình 4.4.3.6. Trang List-username



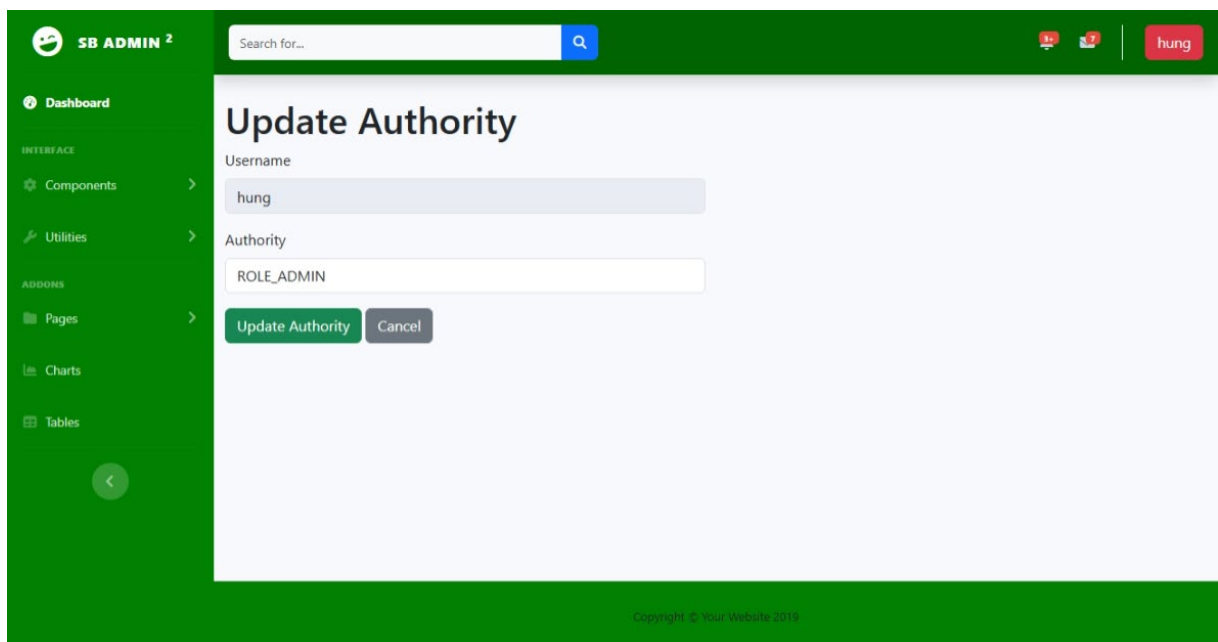
Hình 4.4.3.7. Trang Username-form-update

d) Quản lý vai trò (Authorities)

Chức năng cốt lõi cho phép Admin xem danh sách tất cả Authorities, thêm Authorities cho Users, chỉnh sửa thông tin Authorities hiện có và xóa Authorities (thông qua action trong controller). Giao diện thường là dạng bảng dữ liệu với các nút chức năng tương ứng



Hình 4.4.3.8. Trang List-user



Hình 4.4.3.9. Trang User-form-update

CHƯƠNG 5: TỔNG KẾT

5.1. Kết quả đạt được

Đề tài đặt ra mục tiêu xây dựng một hệ thống quản lý bán hàng trực tuyến với Spring Boot hoàn chỉnh, bao gồm giao diện quản trị dành cho người điều hành và giao diện người dùng thân thiện. Nhìn chung, mục tiêu này đã được hoàn thành ở mức độ cơ bản. Hệ thống đã xây dựng thành công hai ứng dụng web độc lập dựa trên nền tảng Spring Boot, thể hiện rõ sự phân tách vai trò giữa trang Admin và trang Customer. Việc áp dụng kiến trúc Model-View-Controller (MVC) cùng với các công nghệ như Spring Data JPA để tương tác với cơ sở dữ liệu MySQL và Spring Security để bảo mật đã được thực hiện thành công, tạo nên một cấu trúc ứng dụng rõ ràng và có khả năng mở rộng.

Cụ thể hơn, trang quản trị (Admin) đã cung cấp được các chức năng cốt lõi cho người quản lý. Chức năng đăng nhập bảo mật đã được thiết lập, đảm bảo chỉ quản trị viên mới có quyền truy cập. Module quản lý thông tin khách hàng đã được hoàn thiện với đầy đủ các thao tác thêm, sửa, xóa và xem danh sách, thể hiện qua các Controller và View tương ứng (Customers, Products, Authorities, Users). Bên cạnh đó, các yếu tố trực quan hóa dữ liệu như bảng biểu (sử dụng DataTables) và biểu đồ (sử dụng Chart.js) cũng đã được tích hợp, cung cấp một cái nhìn tổng quan ban đầu về hoạt động của hệ thống. Giao diện quản trị được tổ chức khoa học với layout nhất quán và thanh điều hướng tiện lợi.

Đối với trang người dùng (Customer), hệ thống đã xây dựng thành công giao diện chính, bao gồm trang chủ (index.html, home.html) và các trang hiển thị thông tin cần thiết như giới thiệu (about-us.html), liên hệ (contact-us.html), dịch vụ (services.html). Chức năng hiển thị sản phẩm hoặc thông tin liên quan cũng đã được triển khai (ví dụ: ProductDBService, detail-product.html), cho phép người dùng xem và tìm hiểu về các sản phẩm/dịch vụ mà hệ thống cung cấp. Giao diện người dùng được thiết kế hướng tới sự thân thiện và đảm bảo tính responsive nhờ vào việc sử dụng Bootstrap và các kỹ thuật CSS hiện đại, mang lại trải nghiệm tốt trên nhiều loại thiết bị. Tóm lại, hệ thống đã hình thành và hoạt động được với các chức năng cơ bản theo đúng yêu cầu đề ra ban đầu.

5.2. Hạn chế và Điểm cần cải thiện

Mặc dù đã đạt được những kết quả khả quan, hệ thống vẫn còn tồn tại một số hạn chế cần được nhìn nhận và cải thiện. Về mặt chức năng, nhiều tính năng nâng cao vẫn chưa được triển khai hoặc chỉ ở mức độ sơ khai. Ví dụ, chức năng tìm kiếm và lọc sản phẩm/thông tin còn khá đơn giản, chưa đáp ứng được các nhu cầu phức tạp. Các quy trình nghiệp vụ quan trọng như quản lý giỏ hàng, đặt hàng, và đặc biệt là tích hợp thanh toán trực tuyến vẫn chưa được hiện thực hóa. Phần báo cáo và thống kê trong trang Admin tuy đã có nhưng còn hạn chế về mặt thông tin và tùy biến.

Bên cạnh đó, trải nghiệm người dùng (UI/UX) vẫn còn không gian để cải thiện. Mặc dù đã sử dụng Bootstrap, giao diện ở một số trang có thể được thiết kế lại để trở nên trực quan, hấp dẫn và tối ưu hơn nữa cho người dùng cuối. Tính tương thích hoàn toàn trên mọi kích thước màn hình và trình duyệt cũng cần được kiểm tra kỹ lưỡng hơn. Về mặt kỹ thuật, hiệu năng của hệ thống khi đối mặt với lượng dữ liệu lớn hoặc số lượng truy cập đồng thời cao chưa được đánh giá và tối ưu một cách bài bản. Các khía cạnh bảo mật nâng cao, ngoài những gì Spring Security cung cấp mặc định, cũng cần được xem xét bổ sung để tăng cường khả năng phòng thủ cho hệ thống.

Một trong những hạn chế lớn khi không xuất được API và tạo microservices là việc thiếu tính mở rộng và khả năng tái sử dụng. Khi hệ thống không được thiết kế dưới dạng API hoặc microservices, việc thêm chức năng mới hoặc thay đổi hệ thống sẽ trở nên phức tạp và tốn thời gian. Điều này cũng làm giảm khả năng duy trì, vì việc cập nhật một phần của hệ thống có thể ảnh hưởng đến các phần khác, dẫn đến nguy cơ xảy ra lỗi không mong muốn. Ngoài ra, việc thiếu phân tách rõ ràng các dịch vụ trong hệ thống khiến cho việc kiểm tra và triển khai trở nên khó khăn, đặc biệt khi quy mô hệ thống ngày càng lớn. Một điểm cần cải thiện là cần phải có chiến lược rõ ràng hơn trong việc chia nhỏ ứng dụng thành các dịch vụ độc lập, dễ dàng bảo trì và phát triển mà không gây ảnh hưởng quá lớn đến toàn bộ hệ thống. Điều này không chỉ giúp tăng cường khả năng mở rộng mà còn giúp cải thiện hiệu suất, giảm thiểu sự phụ thuộc và nâng cao độ linh hoạt trong quá trình phát triển phần mềm.

Cuối cùng, quy trình phát triển cũng có những điểm cần khắc phục. Việc thiếu các quy trình kiểm thử tự động (Unit Test, Integration Test) khiến việc đảm bảo chất lượng và phát hiện lỗi sớm trở nên khó khăn hơn, chủ yếu dựa vào kiểm thử thủ công. Tài liệu kỹ

thuật đi kèm còn sơ sài. Một số phần mã nguồn có thể cần được tái cấu trúc (refactoring) để tuân thủ tốt hơn các nguyên tắc thiết kế, giúp mã nguồn dễ đọc, dễ bảo trì và mở rộng hơn trong tương lai. Việc xử lý lỗi và ghi log (logging) cũng cần được chuẩn hóa và chi tiết hơn trên toàn bộ hệ thống.

5.3. Hướng phát triển của đề tài

Từ những kết quả đã đạt được và nhận diện các hạn chế, đề tài mở ra nhiều hướng phát triển tiềm năng trong tương lai. Trước hết, việc hoàn thiện và mở rộng các chức năng hiện có là ưu tiên hàng đầu. Điều này bao gồm việc xây dựng đầy đủ các module quản lý còn thiếu như quản lý đơn hàng, quản lý kho, quản lý nội dung (blog/tin tức), và phát triển các tính năng phức tạp hơn như tìm kiếm nâng cao, bộ lọc đa dạng, hệ thống mã giảm giá. Tích hợp các cổng thanh toán trực tuyến phổ biến tại Việt Nam như MoMo hay VNPAY sẽ là một bước tiến quan trọng để hoàn thiện quy trình kinh doanh (nếu có).

Song song với việc bổ sung chức năng, các cải tiến về mặt kỹ thuật cũng cần được chú trọng. Tối ưu hóa hiệu năng thông qua việc rà soát và cải thiện các câu lệnh truy vấn CSDL, áp dụng kỹ thuật caching, và tối ưu tài nguyên frontend là rất cần thiết. Việc tăng cường bảo mật bằng cách triển khai các biện pháp phòng chống tấn công tiên tiến và thực hiện kiểm tra lỗ hổng định kỳ sẽ giúp hệ thống an toàn hơn. Xây dựng một bộ kiểm thử tự động toàn diện (Unit Test, Integration Test) sẽ nâng cao chất lượng mã nguồn và độ tin cậy của hệ thống. Nghiên cứu áp dụng các mẫu thiết kế (Design Patterns) phù hợp và tái cấu trúc mã nguồn sẽ cải thiện khả năng bảo trì và mở rộng.

Về lâu dài, có thể xem xét các hướng đi mang tính chiến lược hơn. Việc cải thiện đáng kể giao diện người dùng (UI/UX) theo các xu hướng thiết kế hiện đại sẽ nâng cao trải nghiệm người dùng. Nghiên cứu và áp dụng các quy trình CI/CD sẽ tự động hóa việc tích hợp và triển khai, giúp đưa các thay đổi mới đến người dùng nhanh chóng và ổn định hơn. Đối với các hệ thống có tiềm năng phát triển quy mô lớn, việc nghiên cứu chuyển đổi sang kiến trúc microservices hoặc đóng gói ứng dụng bằng Docker để triển khai linh hoạt trên các nền tảng đám mây (cloud) là những hướng đi đáng cân nhắc. Cuối cùng, việc phát triển ứng dụng di động song song với ứng dụng web cũng là một khả năng để tiếp cận người dùng một cách hiệu quả hơn. Những hướng phát triển này không chỉ khắc phục các hạn chế hiện tại mà còn mở ra tiềm năng to lớn để hệ thống trở nên hoàn thiện, mạnh mẽ và đáp ứng tốt hơn nhu cầu thực tế.

PHỤ LỤC: TÀI LIỆU THAM KHẢO

1. Spring boot. (n.d.). Spring Boot. <https://spring.io/projects/spring-boot>
2. Spring Framework Documentation :: Spring Framework. (n.d.).
<https://docs.spring.io/spring-framework/reference/>
3. Tutorial: Using Thymeleaf. (n.d.).
<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>
4. Jakarta Persistence | Jakarta EE | The Eclipse Foundation. (n.d.). Jakarta® EE:
The New Home of Cloud Native Java.
<https://jakarta.ee/specifications/persistence/>
5. Contributors, M. O. J. T. a. B. (n.d.). Bootstrap. <https://getbootstrap.com/>
6. OpenJS Foundation - openjsf.org. (n.d.). *JQuery*. <https://jquery.com/>
7. *Font awesome*. (n.d.). Font Awesome. <https://fontawesome.com/>
8. *DataTables* | *Javascript table library*. (n.d.). <https://datatables.net/>
9. Chart.js. (n.d.). Open Source HTML5 Charts for Your Website.
<https://www.chartjs.org/>
10. *ganlanyuan/tiny-slider: Vanilla javascript slider for all purposes*. (n.d.). GitHub.
<https://github.com/ganlanyuan/tiny-slider>