

Đề tài: Xây dựng hệ thống bán đồ ăn trực tuyến

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	6
1.1 Giới thiệu về Hệ thống thương mại điện tử và Kiến trúc Microservices.....	6
1.1.1 Tổng quan về Hệ thống Thương mại điện tử (E-commerce Systems).....	6
1.1.2 Sự chuyển dịch từ Kiến trúc Monolithic sang Microservices.....	7
1.1.3 Các thành phần trong Kiến trúc Microservices áp dụng.....	8
1.1.4 Lý do lựa chọn Microservices cho Đề tài.....	9
1.2 Lý do chọn đề tài.....	10
1.3 Phân tích bài toán.....	10
1.3.1 Thực trạng quy trình nghiệp vụ thủ công và các hạn chế.....	10
1.3.2 Nhu cầu chuẩn hóa và tự động hóa quy trình	11
1.3.3 Các thách thức kỹ thuật cần giải quyết.....	11
1.4 Mục tiêu và Yêu cầu của đề tài.....	12
1.4.1 Mục tiêu đề tài	12
1.4.2 Yêu cầu chức năng (Functional Requirements).....	13
1.4.3 Yêu cầu phi chức năng (Non-functional Requirements).....	14
1.4.4 Phạm vi thực hiện (Scope).....	15
1.5 Phương pháp thực hiện.....	16
CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ VÀ HỆ THỐNG.....	17
2.1 Các tác nhân của hệ thống (Actors).....	17
2.2 Biểu đồ Use Case tổng quát.....	18
2.3 Đặc tả các Use Case chính.....	19
2.3.1 Quản Lý Nhân Viên.....	19
2.3.2 Biểu đồ Use Case Quản lý khách hàng.....	23
2.3.3 Biểu đồ Use Case Quản lý sản phẩm.....	26
2.4 Biểu đồ hoạt động (Activity Diagram).....	29
2.5 Biểu đồ tuần tự (Sequence Diagram).....	45
2.6 Thiết kế cơ sở dữ liệu	55

2.6.1 Phần khách hàng.....	55
2.6.2 Phần nhân viên.....	56
2.6.3 Phần sản phẩm.....	57
2.7 Thiết kế mô hình hướng dịch vụ.....	59
2.7.1 Định hướng thiết kế.....	59
2.7.2. Các dịch vụ chính trong hệ thống.....	59
2.7.3 Giao tiếp giữa các dịch vụ	60
2.7.4. Mối liên hệ giữa giao diện và các dịch vụ.....	60
2.7.5. Ưu điểm của mô hình đã triển khai.....	60
2.8 Giao diện mẫu.....	61
2.8.1 Giao diện phía khách hàng.....	61
2.8.2 Giao diện phía nhân viên	65
2.8.3. Giao diện phía quản trị viên.....	67
CHƯƠNG 3. CÔNG CỤ, CÔNG NGHỆ VÀ TRIỂN KHAI HỆ THỐNG	68
3.1 Công cụ và công nghệ sử dụng	68
3.1.1 Ngôn ngữ lập trình và Framework (Backend)	68
3.1.2 Giao diện người dùng (Frontend)	68
3.1.3 Hệ quản trị Cơ sở dữ liệu (Database).....	69
3.1.4. Các công cụ hỗ trợ phát triển (DevTools).....	69
3.2 Các bước cài đặt hệ thống.....	70
3.3. Mô tả quy trình Demo (Kịch bản kiểm thử)	71
3.3.1 Kịch bản 1: Khách hàng mua hàng thành công.....	71
3.3.2 Kịch bản 2: Quản trị viên quản lý	74
3.3.3. Kiểm thử hộp đen.....	75
CHƯƠNG 4. KẾT QUẢ VÀ ĐÁNH GIÁ	90
4.1 Tóm tắt kết quả và lợi ích đạt được	90
4.1.1 Về mặt kỹ thuật.....	90
4.1.2 Về mặt thực tiễn.....	90
4.2 Đánh giá kết quả thực hiện	90
4.3 Hạn chế và hướng phát triển.....	91
TÀI LIỆU THAM KHẢO	92

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Giới thiệu về Hệ thống thương mại điện tử và Kiến trúc Microservices

1.1.1 Tổng quan về Hệ thống Thương mại điện tử (E-commerce Systems)

a. Khái niệm và Phân loại

Thương mại điện tử (E-commerce) được định nghĩa là việc tiến hành các giao dịch thương mại (mua bán hàng hóa, dịch vụ, thông tin) thông qua mạng Internet và các phương tiện điện tử khác. Trong kỷ nguyên số, E-commerce không chỉ đơn thuần là việc đưa sản phẩm lên website, mà là một hệ sinh thái phức tạp bao gồm quản lý chuỗi cung ứng, xử lý giao dịch trực tuyến, trao đổi dữ liệu điện tử (EDI), quản lý quan hệ khách hàng (CRM) và các hệ thống thu thập dữ liệu tự động.

Hệ thống mà nhóm phát triển thuộc mô hình B2C (Business-to-Consumer). Đây là loại hình giao dịch thương mại giữa doanh nghiệp (nhà hàng, cửa hàng đồ ăn) và người tiêu dùng cuối cùng. Trong mô hình này, hệ thống thông tin đóng vai trò trung gian quan trọng, giúp doanh nghiệp tiếp cận khách hàng mọi lúc, mọi nơi, đồng thời cung cấp cho khách hàng sự tiện lợi tối đa trong việc lựa chọn và thanh toán.

b. Đặc thù của Hệ thống Thương mại điện tử ngành F&B (Food & Beverage)

Khác với các hệ thống thương mại điện tử bán lẻ thông thường (như bán quần áo, đồ điện tử), hệ thống bán đồ ăn nhanh trực tuyến có những yêu cầu khắt khe và đặc thù riêng biệt về mặt xử lý thông tin:

- **Tính tức thời (Real-time constraints):** Vòng đời của một đơn hàng trong ngành F&B rất ngắn, thường chỉ kéo dài từ 30 đến 60 phút kể từ lúc khách đặt món đến khi hoàn thành giao hàng. Do đó, hệ thống đòi hỏi độ trễ cực thấp (Low Latency). Thông tin đơn hàng sau khi được xác nhận phải được đồng bộ ngay lập tức xuống bộ phận bếp và bộ phận giao hàng. Bất kỳ sự chậm trễ nào trong luồng dữ liệu cũng sẽ ảnh hưởng trực tiếp đến chất lượng món ăn (nguyên, hỏng) và trải nghiệm khách hàng.
- **Quản lý biến thể sản phẩm phức tạp (Complex Product Attributes):** Sản phẩm đồ ăn thường không tồn tại ở dạng đơn lẻ mà đi kèm với nhiều tùy chọn tùy biến (Customization). Ví dụ: Một chiếc Pizza có thể có nhiều kích cỡ (S, M, L), nhiều loại đế (dày, mỏng), và hàng chục loại topping thêm bớt khác nhau. Cấu trúc dữ liệu của hệ thống phải đủ linh hoạt để lưu trữ và xử lý chính xác các biến thể này trong giỏ hàng và hóa đơn, đảm bảo bếp nhận đúng yêu cầu chế biến.
- **Khả năng chịu tải cao điểm (High Concurrency handling):** Đặc thù của ngành ăn uống là nhu cầu tập trung dồn dập vào các khung giờ vàng (bữa trưa từ 11h-12h, bữa tối từ 18h-19h). Hệ thống phải có khả năng xử lý hàng nghìn yêu cầu đồng thời (Concurrent Requests) trong khoảng thời gian ngắn mà không bị tắc nghẽn hay sập nguồn. Đây là thách thức lớn đối với các kiến trúc phần mềm truyền thống.

c. Các thành phần chức năng cốt lõi

Một hệ thống thương mại điện tử B2C tiêu chuẩn cho ngành đồ ăn nhanh bao gồm các phân hệ chính:

- **Front-office (Storefront):** Giao diện tương tác với người dùng (Web/App), nơi hiển thị Catalog sản phẩm, công cụ tìm kiếm, giỏ hàng và cổng thanh toán.
- **Back-office (Admin Panel):** Giao diện dành cho nhà quản trị và nhân viên để quản lý thực đơn, thiết lập giá, quản lý kho (Inventory), và xử lý đơn hàng (Order Fulfillment).
- **Middleware & API:** Lớp trung gian kết nối giữa Front-office và Back-office, cũng như tích hợp với các dịch vụ bên thứ 3 (Công thanh toán ngân hàng, Dịch vụ vận chuyển, SMS Gateway).

1.1.2 Sự chuyển dịch từ Kiến trúc Monolithic sang Microservices

Để xây dựng một hệ thống đáp ứng được các yêu cầu đặc thù nêu trên, việc lựa chọn kiến trúc phần mềm là quyết định quan trọng nhất.

a. Hạn chế của Kiến trúc Nguyên khối (Monolithic Architecture)

Theo truyền thống, các ứng dụng web thường được xây dựng theo kiến trúc Monolithic. Trong mô hình này, tất cả các module chức năng (Quản lý người dùng, Quản lý sản phẩm, Đặt hàng, Thanh toán, Báo cáo...) được đóng gói trong một khối mã nguồn duy nhất (ví dụ: một file .war hoặc .jar không lòi) và sử dụng chung một cơ sở dữ liệu (Single Database).

Mặc dù đơn giản trong giai đoạn đầu phát triển, kiến trúc Monolithic bộc lộ nhiều nhược điểm chí mạng khi hệ thống mở rộng quy mô:

- **Điểm chết duy nhất (Single Point of Failure):** Nếu một module nhỏ gặp lỗi (ví dụ: module Xuất báo cáo bị tràn bộ nhớ), nó có thể kéo sập toàn bộ ứng dụng, khiến khách hàng không thể đặt món.
- **Khó mở rộng (Hard to Scale):** Khi lưu lượng truy cập tăng, ta buộc phải nhân bản toàn bộ ứng dụng lên nhiều server, gây lãng phí tài nguyên. Không thể chỉ mở rộng riêng module "Đặt hàng" mà giữ nguyên module "Quản trị".
- **Công nghệ bị khóa chặt (Technology Lock-in):** Toàn bộ hệ thống phải sử dụng cùng một ngôn ngữ lập trình và framework. Việc nâng cấp hay thay đổi công nghệ cực kỳ rủi ro và tốn kém.
- **Sự phức tạp trong bảo trì:** Codebase quá lớn khiến việc debug, kiểm thử và deploy trở nên chậm chạp. Các lập trình viên mới mất nhiều thời gian để hiểu luồng đi của dữ liệu.

b. Sự ra đời của Kiến trúc Vi dịch vụ (Microservices Architecture)

Microservices là một phong cách kiến trúc phần mềm, trong đó một ứng dụng lớn được phân rã thành một tập hợp các dịch vụ nhỏ, độc lập (Services). Các dịch vụ này được tổ chức xoay quanh các năng lực nghiệp vụ (Business Capabilities) cụ thể.

Các đặc điểm nhận diện chính của Microservices:

- **Độc lập (Independent Deployment):** Mỗi dịch vụ có thể được build, deploy và update riêng biệt mà không ảnh hưởng đến các dịch vụ khác.
- **Cơ sở dữ liệu riêng (Database per Service):** Đây là đặc điểm quan trọng nhất. Mỗi dịch vụ sở hữu và quản lý dữ liệu của riêng mình. Điều này đảm bảo tính lỏng lẻo (Loose Coupling) giữa các dịch vụ.
- **Giao tiếp qua mạng:** Các dịch vụ giao tiếp với nhau thông qua các giao thức mang nhẹ (lightweight protocols), phổ biến nhất là HTTP/REST (cho giao tiếp đồng bộ) hoặc Message Queue như Kafka/RabbitMQ (cho giao tiếp bất đồng bộ).

1.1.3 Các thành phần trong Kiến trúc Microservices áp dụng

Trong khuôn khổ đề tài này, nhóm áp dụng hệ sinh thái **Spring Cloud** để triển khai kiến trúc Microservices. Dưới đây là phân tích chi tiết các thành phần kỹ thuật cấu thành nên hệ thống:

a. Service Discovery (*Cơ chế phát hiện dịch vụ*)

Trong môi trường phân tán, các dịch vụ thường được triển khai trên các container (Docker) với địa chỉ IP thay đổi động. Các dịch vụ không thể biết trước địa chỉ IP của nhau để gọi API.

- **Giải pháp:** Nhóm sử dụng Netflix Eureka làm Service Registry.
- **Cơ chế:** Khi một service (ví dụ: Product Service) khởi động, nó sẽ tự động đăng ký địa chỉ IP và Port của mình lên Eureka Server. Khi Order Service cần gọi Product Service, nó sẽ hỏi Eureka để lấy danh sách địa chỉ khả dụng. Điều này giúp hệ thống linh hoạt, dễ dàng thêm/bớt server mà không cần cấu hình lại code.

b. API Gateway (*Cổng giao tiếp tập trung*)

Client (trình duyệt, mobile app) không nên gọi trực tiếp đến hàng chục microservices riêng lẻ. Điều này gây khó khăn trong việc quản lý bảo mật và tối ưu hiệu năng.

- **Giải pháp:** Nhóm sử dụng **Spring Cloud Gateway**.
- **Vai trò:** API Gateway đóng vai trò là "người gác cổng" duy nhất. Mọi yêu cầu từ Client đều phải đi qua Gateway. Tại đây, Gateway thực hiện:

1. *Routing:* Định tuyến yêu cầu đến đúng service đích (ví dụ: /api/products -> Product Service).
2. *Authentication:* Kiểm tra Token xác thực người dùng.

3. *Load Balancing*: Cân bằng tải giữa các instance của cùng một service.
c. *Centralized Configuration (Quản lý cấu hình tập trung)*

Việc quản lý file cấu hình (application.properties) cho từng service riêng lẻ rất khó khăn khi số lượng service tăng lên.

- **Giải pháp:** Nhóm sử dụng **Spring Cloud Config**.
- **Cơ chế:** Toàn bộ cấu hình của hệ thống được lưu trữ tập trung tại một Git Repository. Config Server sẽ đọc cấu hình từ Git và cung cấp cho các microservices khi chúng khởi động. Việc này cho phép thay đổi cấu hình hệ thống (ví dụ: thay đổi timeout, bật/tắt tính năng) mà không cần build lại code.

d. *Inter-service Communication (Giao tiếp giữa các dịch vụ)*

- **Giao tiếp đồng bộ (Synchronous):** Sử dụng **OpenFeign** (một declarative REST client) để các service gọi nhau như gọi hàm nội bộ. Ví dụ: Order Service gọi Payment Service để xác nhận thanh toán.
- **Xử lý lỗi (Fault Tolerance):** Tích hợp **Resilience4j** để cài đặt cơ chế Circuit Breaker (Ngắt mạch). Nếu Payment Service bị lỗi hoặc phản hồi chậm, Circuit Breaker sẽ ngắt kết nối tạm thời để tránh làm treo Order Service, đồng thời trả về thông báo lỗi thân thiện hoặc dữ liệu cache cho người dùng.

1.1.4 Lý do lựa chọn Microservices cho Đề tài

Việc áp dụng kiến trúc Microservices cho "Hệ thống bán đồ ăn trực tuyến" không chỉ là sự thử nghiệm công nghệ mà xuất phát từ những lợi ích thực tiễn giải quyết bài toán nghiệp vụ:

a. *Khả năng mở rộng theo chiều ngang (Horizontal Scalability)*

Trong giờ cao điểm, dịch vụ "Đặt hàng" (Order Service) và "Tìm kiếm sản phẩm" (Product Service) sẽ chịu tải rất lớn, trong khi dịch vụ "Quản lý nhân viên" (Employee Service) hầu như không có truy cập. Với Microservices, nhóm có thể thiết lập để tự động khởi chạy thêm 5 instance cho Order Service mà giữ nguyên 1 instance cho Employee Service. Điều này giúp hệ thống luôn mượt mà với chi phí tài nguyên tối ưu nhất.

b. *Tăng tốc độ phát triển và bảo trì (Agility)*

Hệ thống được chia nhỏ giúp việc phân chia công việc trong nhóm dễ dàng hơn.

- Thành viên A phụ trách *Product Service* có thể tự do chỉnh sửa cấu trúc bảng Sản phẩm, viết API và deploy mà không sợ làm hỏng code của thành viên B đang làm *Order Service*, miễn là API contract (đầu vào/đầu ra) không thay đổi.
- Điều này phù hợp với phương pháp làm việc nhóm Agile/Scrum mà nhóm đang áp dụng.

c. Tính sẵn sàng cao (High Availability)

Đối với một hệ thống bán hàng, "downtime" (thời gian chết) đồng nghĩa với việc mất doanh thu. Với kiến trúc Microservices, các dịch vụ nghiệp vụ được tách biệt hoàn toàn giúp cô lập lỗi (Fault Isolation). Ví dụ: Nếu **Employee Service** (Dịch vụ Quản lý nhân viên) gặp sự cố hoặc đang trong quá trình bảo trì, các chức năng quản trị nội bộ có thể bị gián đoạn, nhưng khách hàng vẫn có thể truy cập website, xem thực đơn và đặt hàng bình thường thông qua sự phối hợp của **Product Service** và **Order Service**. Điều này đảm bảo luồng doanh thu cốt lõi của cửa hàng không bị ảnh hưởng bởi các sự cố của các phân hệ phụ trợ hoặc quản lý nội bộ.

d. Khả năng tích hợp chức năng mới linh hoạt

Trong tương lai, nếu cửa hàng muốn tích hợp thêm tính năng "Gợi ý món ăn bằng AI" viết bằng Python, hệ thống Microservices cho phép dễ dàng tích hợp service mới này vào hệ sinh thái hiện có (đang chạy Java) thông qua giao tiếp REST API mà không cần viết lại hệ thống cũ.

1.2 Lý do chọn đề tài

Sự phát triển của thương mại điện tử và thói quen tiêu dùng thay đổi sau đại dịch đã khiến việc đặt đồ ăn trực tuyến trở thành một phần không thể thiếu trong đời sống hiện đại. Các cửa hàng F&B truyền thống đang đứng trước áp lực phải chuyển đổi số để cạnh tranh.

Tuy nhiên, các giải pháp quản lý bán hàng thủ công hiện tại bộc lộ quá nhiều yếu điểm: dễ sai sót, khó kiểm soát và không tối ưu được quy trình vận hành. Việc xây dựng một hệ thống bán hàng trực tuyến chuyên nghiệp không chỉ là nhu cầu cấp thiết của doanh nghiệp mà còn là cơ hội để nhóm sinh viên áp dụng những công nghệ mới nhất (Java Spring Boot, Microservices, ReactJS) vào thực tế.

Đề tài "**Xây dựng hệ thống bán đồ ăn trực tuyến**" được lựa chọn vì tính thực tiễn cao, độ phức tạp nghiệp vụ vừa đủ để triển khai mô hình Microservices, giúp nhóm rèn luyện kỹ năng phân tích thiết kế hệ thống và lập trình hướng dịch vụ.

1.3 Phân tích bài toán

1.3.1 Thực trạng quy trình nghiệp vụ thủ công và các hạn chế

Trong mô hình kinh doanh đồ ăn nhanh truyền thống, quy trình từ lúc khách hàng gọi món đến khi hoàn tất thanh toán thường diễn ra theo phương thức thủ công hoặc sử dụng các phần mềm đơn lẻ, rời rạc. Qua khảo sát thực tế, nhóm nhận thấy quy trình này tồn tại nhiều bất cập nghiêm trọng :

- **Quy trình ghi nhận đơn hàng (Ordering):** Nhân viên phục vụ phải ghi chép order bằng giấy hoặc nhập liệu vào các hệ thống POS cục bộ (offline). Việc này dễ dẫn đến

sai sót như: nghe nhầm món, quên ghi chú tùy chọn (ít đường, không hành...), hoặc thất lạc phiếu order khi chuyển xuống bếp.

- **Quy trình kiểm soát tồn kho (Inventory):** Việc trừ kho thường chỉ được thực hiện vào cuối ngày hoặc cuối ca làm việc. Điều này dẫn đến tình trạng "lệch tồn kho" thực tế: trên menu vẫn còn món nhưng thực tế trong bếp đã hết nguyên liệu, buộc nhân viên phải quay lại xin lỗi khách và yêu cầu đổi món, gây trải nghiệm rất tệ.
- **Quy trình thanh toán và xuất hóa đơn:** Khách hàng phải xếp hàng chờ đợi tại quầy thu ngân để thanh toán. Việc xuất hóa đơn giấy tốn thời gian và chi phí in ấn. Hơn nữa, việc tra cứu lại hóa đơn cũ khi có khiếu nại là cực kỳ khó khăn.
- **Khả năng chịu tải:** Vào các khung giờ cao điểm (như giờ nghỉ trưa), lượng khách dồn về cửa hàng đông. Quy trình thủ công tạo ra các "nút thắt cổ chai" (bottlenecks) tại quầy thu ngân và khu vực bếp, khiến tốc độ phục vụ giảm sút nghiêm trọng.

1.3.2 Nhu cầu chuẩn hóa và tự động hóa quy trình

Từ những hạn chế trên, bài toán đặt ra là phải xây dựng một hệ thống thông tin khép kín, tự động hóa các luồng nghiệp vụ cốt lõi sau:

- **Số hóa thực đơn và đặt hàng:**

1. Cần một cơ chế cho phép quản lý hàng trăm món ăn với các thuộc tính phức tạp (Size, Topping) một cách có cấu trúc.
2. Khách hàng phải có khả năng tự thao tác chọn món, xem giá tiền theo thời gian thực (Real-time pricing) mà không cần nhân viên hỗ trợ.

- **Đồng bộ hóa dữ liệu tồn kho:**

Hệ thống phải giải quyết bài toán trừ tồn kho ngay lập tức khi đơn hàng được khởi tạo (hoặc xác nhận). Nếu kho hết, hệ thống phải tự động chặn không cho đặt món đó trên giao diện người dùng.

- **Tích hợp quy trình Thanh toán và Hóa đơn:**

1. Đây là bài toán nghiệp vụ phức tạp mà nhóm cần giải quyết. Hệ thống không chỉ đơn thuần ghi nhận trạng thái "Đã thanh toán", mà phải tự động kích hoạt quy trình sinh hóa đơn điện tử (Invoice Generation).
2. Cụ thể: Ngay khi **Payment Service** xác nhận giao dịch thành công, nó phải gửi tín hiệu sang **Order Service** để thay đổi trạng thái đơn hàng, đồng thời hệ thống phải tự động trích xuất dữ liệu để tạo file hóa đơn PDF lưu trữ, sẵn sàng cho khách hàng tải về.

1.3.3 Các thách thức kỹ thuật cần giải quyết

Việc chuyển đổi từ quy trình thủ công sang hệ thống trực tuyến đặt ra nhiều thách thức kỹ thuật, đặc biệt đối với ngành F&B có tốc độ giao dịch nhanh:

a. Bài toán về tính toàn vẹn dữ liệu trong môi trường phân tán

Do hệ thống được chia nhỏ thành các dịch vụ riêng biệt (Order Service, Product Service, Payment Service), việc đảm bảo tính nhất quán của dữ liệu (Data Consistency) là một thách thức lớn.

Ví dụ: Khách hàng đã thanh toán tiền (tại Payment Service) nhưng quá trình cập nhật trạng thái đơn hàng (tại Order Service) bị lỗi mạng. Hệ thống cần có cơ chế xử lý để đảm bảo khách hàng không bị mất tiền oan và đơn hàng vẫn được ghi nhận đúng.

c. Bài toán về hiệu năng và khả năng chịu tải (Scalability)

Hệ thống phải có khả năng phục vụ đồng thời hàng trăm, hàng nghìn lượt truy cập vào giờ cao điểm mà không bị sập nguồn (Crash). Điều này đòi hỏi kiến trúc hệ thống phải cho phép mở rộng linh hoạt (Scaling) các module quan trọng như *Tìm kiếm sản phẩm* hay *Đặt hàng* mà không lãng phí tài nguyên cho các module quản trị ít dùng.

d. Bài toán về bảo mật thông tin

Khi đưa việc bán hàng lên môi trường trực tuyến, hệ thống phải đối mặt với các nguy cơ tấn công mạng. Bài toán đặt ra là phải xây dựng cơ chế xác thực mạnh mẽ (Authentication & Authorization) để đảm bảo chỉ nhân viên có quyền mới được truy cập dữ liệu nhạy cảm và thông tin khách hàng được bảo vệ tuyệt đối.

1.4 Mục tiêu và Yêu cầu của đề tài

1.4.1 Mục tiêu đề tài

a. Mục tiêu tổng quát

Xây dựng thành công một hệ thống thương mại điện tử hoàn chỉnh phục vụ cho mô hình kinh doanh đồ ăn nhanh (Fast Food), giải quyết triệt để các hạn chế của quy trình quản lý thủ công hiện tại. Hệ thống không chỉ là một website bán hàng đơn thuần mà là một nền tảng quản lý toàn diện từ khâu đặt món, kiểm soát tồn kho đến thanh toán và xuất hóa đơn điện tử.

Đồng thời, đề tài hướng đến việc nghiên cứu và áp dụng thực tiễn kiến trúc **Microservices** – một trong những xu hướng công nghệ phổ biến nhất hiện nay, giúp sinh viên làm chủ quy trình phát triển phần mềm hiện đại, quy mô lớn và có tính linh hoạt cao.

b. Mục tiêu cụ thể về mặt nghiệp vụ

- **Tự động hóa quy trình bán hàng:** Loại bỏ hoàn toàn thao tác ghi chép thủ công. Đảm bảo luồng dữ liệu thông suốt từ Khách hàng -> Hệ thống -> Nhà bếp -> Thu ngân.
- **Minh bạch hóa tài chính:** Tích hợp quy trình thanh toán và xuất hóa đơn tự động, giúp khách hàng an tâm và chủ cửa hàng dễ dàng đối soát doanh thu, tránh thất thoát.

- **Nâng cao trải nghiệm người dùng:** Cung cấp giao diện trực quan, tốc độ phản hồi nhanh, giúp khách hàng dễ dàng tìm kiếm món ăn và đặt hàng trong thời gian ngắn nhất.

c. *Mục tiêu cụ thể về mặt kỹ thuật*

- Xây dựng được các **Microservices** cốt lõi hoạt động độc lập: Customer, Product, Order, Payment, Employee.
- Triển khai thành công các thành phần hạ tầng của hệ sinh thái **Spring Cloud**:
 1. **Service Discovery (Eureka):** Quản lý định danh và trạng thái sống (health check) của các dịch vụ.
 2. **API Gateway:** Cổng giao tiếp tập trung, điều hướng request và bảo mật.
 3. **Config Server:** Quản lý cấu hình tập trung cho toàn bộ hệ thống.
- Thực hiện giao tiếp giữa các dịch vụ (Inter-service communication) thông qua **RESTful API** (sử dụng OpenFeign).
- Đảm bảo an toàn bảo mật hệ thống bằng cơ chế xác thực và phân quyền **JWT (JSON Web Token)**.

1.4.2 Yêu cầu chức năng (Functional Requirements)

Hệ thống được phân quyền và cung cấp các nhóm chức năng chi tiết cho từng đối tượng người dùng dựa trên phân tích Use Case:

a. *Nhóm chức năng dành cho Khách hàng (Customer)*

- **Đăng ký và Đăng nhập:** Cho phép người dùng tạo tài khoản mới bằng email và đăng nhập để sử dụng dịch vụ. Hệ thống thực hiện xác thực và cấp token phiên làm việc.
- **Quản lý hồ sơ (Profile):** Cập nhật thông tin cá nhân (họ tên, số điện thoại) và địa chỉ giao hàng mặc định.
- **Tìm kiếm và Xem sản phẩm:** Tìm kiếm món ăn theo tên, lọc theo danh mục (Pizza, Đồ uống, Khai vị...), xem chi tiết giá, mô tả và hình ảnh trực quan.
- **Quản lý Giỏ hàng (Cart):** Thêm món vào giỏ, cập nhật số lượng (tăng/giảm), xóa món khỏi giỏ hàng. Hệ thống tự động kiểm tra tồn kho trước khi thêm.
- **Đặt hàng (Place Order):** Xác nhận thông tin đơn hàng, chọn phương thức thanh toán và tiến hành đặt hàng (Checkout).
- **Thanh toán và Nhận hóa đơn:**
 1. Thực hiện thanh toán đơn hàng (qua Payment Service).
 2. Sau khi thanh toán thành công, khách hàng có thể xem và tải về **Hóa đơn điện tử (PDF)**. Lưu ý: *Chức năng này được Order Service tự động xử lý và lưu trữ link tải sau khi nhận tín hiệu từ Payment Service.*
- **Lịch sử đơn hàng:** Xem lại danh sách các đơn đã đặt và theo dõi trạng thái hiện tại (Đang xử lý, Đã giao, Đã hủy).

b. Nhóm chức năng dành cho Nhân viên (Employee)

- **Tiếp nhận đơn hàng:** Xem danh sách đơn hàng mới đầy đủ từ hệ thống theo thời gian thực.
- **Xử lý đơn hàng:** Cập nhật trạng thái đơn hàng theo quy trình thực tế (Xác nhận -> Đang chuẩn bị -> Đang giao -> Hoàn thành).
- **Xử lý hủy/hoàn đơn:** Thực hiện hủy đơn hàng khi có yêu cầu hoặc sự cố, hệ thống tự động hoàn tác số lượng tồn kho tương ứng.
- **Quản lý Sản phẩm:** Thêm mới món ăn, sửa giá, cập nhật hình ảnh, xóa món ăn ngừng kinh doanh.
- **Gửi thông báo:** Gửi email xác nhận hoặc thông báo trạng thái đơn hàng tới khách hàng.

c. Nhóm chức năng dành cho Quản trị viên (Admin).

- **Quản lý Sản phẩm:** tương tự nhân viên
- **Quản lý Danh mục:** Tạo mới và chỉnh sửa các nhóm món ăn (Category).
- **Quản lý Nhân sự:** Tạo tài khoản cho nhân viên mới, phân quyền truy cập và khóa tài khoản khi cần thiết.
- **Báo cáo thống kê:** Xem báo cáo doanh thu theo ngày/tháng, thống kê món ăn bán chạy nhất để hỗ trợ ra quyết định kinh doanh.

1.4.3 Yêu cầu phi chức năng (Non-functional Requirements)

Để đảm bảo hệ thống hoạt động ổn định trong môi trường thực tế, các yêu cầu chất lượng sau cần được đáp ứng:

a. Hiệu năng (Performance)

- **Thời gian phản hồi (Response Time):** Các thao tác xem danh sách sản phẩm, thêm vào giỏ hàng phải phản hồi nhanh (dưới 2 giây). Quy trình đặt hàng không quá 5 giây.
- **Khả năng chịu tải:** Hệ thống phải đảm bảo hoạt động ổn định khi có nhiều người dùng truy cập và đặt hàng cùng lúc nhờ cơ chế cân bằng tải (Load Balancing) tích hợp trong API Gateway.

b. Tính tin cậy và sẵn sàng (Reliability & Availability)

- Hệ thống phải hoạt động liên tục 24/7.
- Áp dụng cơ chế **Fault Tolerance** (Chịu lỗi): Nếu một dịch vụ phụ (ví dụ: Gửi email thông báo) bị lỗi, các dịch vụ chính (Đặt hàng, Thanh toán) vẫn phải hoạt động bình thường, không gây sập toàn bộ hệ thống.

c. Bảo mật (Security)

- Mật khẩu người dùng phải được mã hóa (Hashing) trước khi lưu vào cơ sở dữ liệu.
- Mọi API truy cập dữ liệu nhạy cảm phải yêu cầu xác thực qua Token (JWT).
- Ngăn chặn các lỗi bảo mật web phổ biến.

d. Khả năng bảo trì và mở rộng (Maintainability & Scalability)

- Mã nguồn (Source code) phải được tổ chức rõ ràng, tuân thủ các nguyên tắc lập trình sạch (Clean Code).
- Dễ dàng thêm mới một Microservice (ví dụ: thêm dịch vụ Khuyến mãi) mà không cần viết lại các dịch vụ cũ.
- Dễ dàng triển khai (Deploy) trên các môi trường khác nhau nhờ công nghệ Containerization (Docker).

e. Giao diện người dùng (UI/UX)

- Giao diện thân thiện, hiện đại, màu sắc hài hòa với thương hiệu đồ ăn nhanh.
- Tương thích hiển thị (Responsive) trên các thiết bị khác nhau (Laptop, Máy tính bảng).

1.4.4 Phạm vi thực hiện (Scope)

a. Phạm vi và Quy mô hệ thống

Hệ thống được xây dựng ở quy mô thử nghiệm (Prototype) nhằm minh họa cho kiến trúc Microservices, bao gồm 5 dịch vụ lõi:

Customer Service: Quản lý người dùng và xác thực.

Product Service: Quản lý danh mục và sản phẩm.

Order Service: Quản lý đơn hàng và tích hợp logic tạo hóa đơn (Invoice Generation).

Payment Service: Xử lý thanh toán (Giả lập) và kích hoạt xuất hóa đơn.

Employee Service: Quản lý nhân sự nội bộ.

Các dịch vụ này hoạt động độc lập, có cơ sở dữ liệu tách riêng (Database per Service) và giao tiếp với nhau thông qua API Gateway để đảm bảo tính bảo mật và định tuyến thống nhất.

b. Các tính năng không nằm trong phạm vi (Out of Scope)

Để tập trung vào việc hoàn thiện kiến trúc hệ thống và các luồng nghiệp vụ chính trong thời gian có hạn, đề tài không bao gồm các tính năng sau:

- **Tích hợp thanh toán trực tuyến thực tế:** Không tích hợp với các cổng thanh toán ngân hàng (Visa/Mastercard) hoặc ví điện tử (Momo, VNPay). Payment Service chỉ thực hiện mô phỏng quá trình xử lý giao dịch (thành công/thất bại).
- **Tích hợp vận chuyển:** Không kết nối với API thời gian thực của các đơn vị giao hàng (Grab, GHTK).

Hệ thống được xây dựng ở quy mô thử nghiệm với các service chính: **Customer Service**, **Order Service**, **Payment Service**, **Product Service** và **Employee Service**. Các dịch vụ giao tiếp thông qua **API Gateway**, cơ sở dữ liệu được tách riêng cho từng service để đảm bảo tính độc lập theo đúng nguyên tắc microservice. Đề tài không bao gồm các tính năng như thanh toán trực tuyến, tích hợp vận chuyển hoặc đánh giá sản phẩm.

1.5 Phương pháp thực hiện

Đề tài được thực hiện theo các bước:

1. **Khảo sát và phân tích yêu cầu**, xác định rõ các thành phần chính và luồng xử lý nghiệp vụ giữa khách hàng, nhân viên và hệ thống sản phẩm.
2. **Thiết kế hệ thống** theo kiến trúc microservice, xác định các service, API tương tác và mô hình dữ liệu liên quan.
3. **Xây dựng từng dịch vụ độc lập**, bao gồm xử lý nghiệp vụ, kết nối cơ sở dữ liệu và giao tiếp API.
4. **Tích hợp và kiểm thử toàn hệ thống**, đảm bảo các service hoạt động đúng, dữ liệu thống nhất và luồng xử lý diễn ra mạch lạc.
5. **Hoàn thiện tài liệu, báo cáo, và trình bày kết quả mô phỏng**.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ VÀ HỆ THỐNG

2.1 Các tác nhân của hệ thống (Actors)

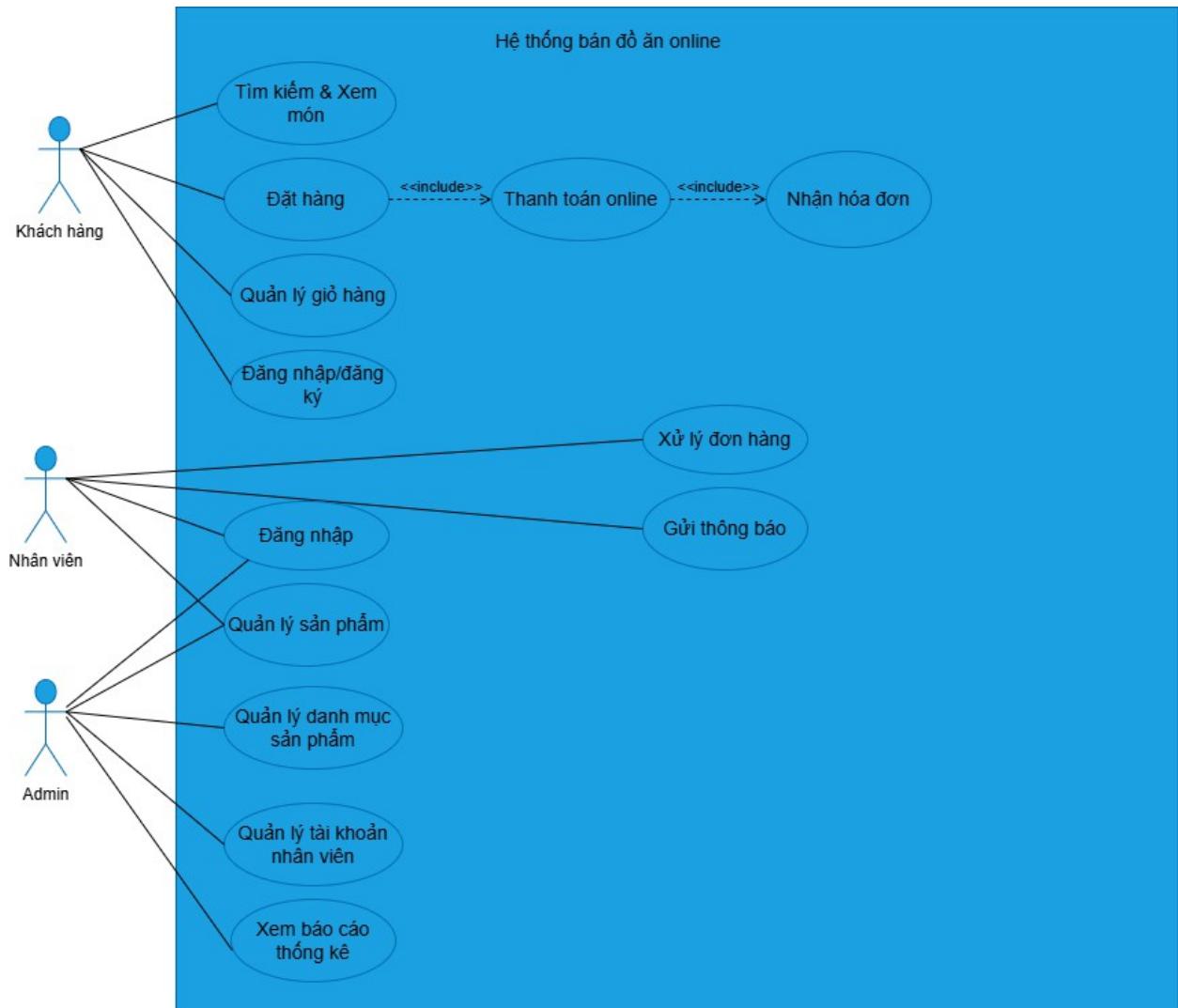
Hệ thống được thiết kế để phục vụ 3 nhóm đối tượng chính, tương tác trực tiếp với các dịch vụ Microservices:

Khách hàng (Customer): Là người dùng cuối truy cập vào hệ thống (Website) để xem thực đơn, đăng ký thành viên, đặt món ăn, thực hiện thanh toán trực tuyến và nhận hóa đơn.

Nhân viên (Employee): Là người làm việc tại cửa hàng, chịu trách nhiệm tiếp nhận đơn hàng, cập nhật trạng thái đơn (đang chuẩn bị, đang giao), quản lý sản phẩm (thêm, sửa, xóa), xem báo cáo thống kê và gửi thông báo hỗ trợ khách hàng.

Quản trị viên (Admin): Là người có quyền hạn cao nhất, chịu trách nhiệm quản lý danh mục sản phẩm, quản lý tài khoản nhân viên và xem báo cáo thống kê doanh thu toàn hệ thống.

2.2 Biểu đồ Use Case tổng quát



Khách hàng (Customer):

1. Đăng ký, Đăng nhập.
2. Tìm kiếm món ăn, Xem chi tiết món ăn.
3. Quản lý giỏ hàng (Thêm/Sửa/Xóa).
4. Đặt hàng (Checkout).
5. Thanh toán trực tuyến (Make Payment).
6. Xem và Tải hóa đơn điện tử (View/Download Invoice).
7. Xem lịch sử đơn hàng.

Nhân viên (Employee):

1. Đăng nhập.
2. Xem danh sách đơn hàng mới.

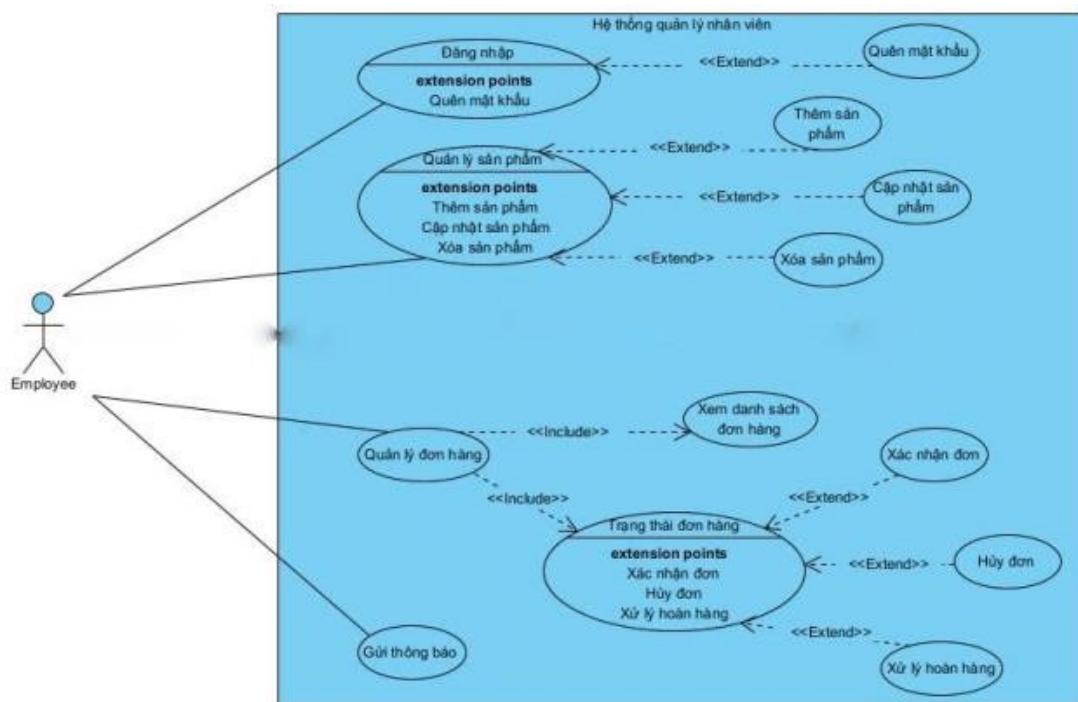
3. Cập nhật trạng thái đơn hàng.
4. Quản lý sản phẩm (Thêm/Sửa/Xóa).
5. Gửi thông báo (thủ công).

Quản trị viên (Admin):

1. Quản lý sản phẩm (Thêm/Sửa/Xóa).
2. Quản lý danh mục.
3. Quản lý tài khoản nhân viên.
4. Xem báo cáo thống kê.

2.3 Đặc tả các Use Case chính

2.3.1 Quản Lý Nhân Viên



• Đặc Tả Use Case

1. USE CASE 1: Đăng nhập (AuthService)
- Actor: Employee

Mô tả sơ lược: Nhân viên sử dụng giao diện đăng nhập để xác thực với hệ thống. AuthService kiểm tra thông tin và cấp token nếu hợp lệ.

Tiền điều kiện:

Nhân viên có tài khoản hợp lệ.

AuthService hoạt động ổn định.

Hậu điều kiện:

Nhân viên nhận access token (JWT).

Token được frontend lưu trữ để sử dụng các chức năng khác.

Luồng chính:

Nhân viên truy cập trang đăng nhập.

Nhập tên đăng nhập và mật khẩu.

Giao diện gửi yêu cầu qua API Gateway đến AuthService.

AuthService kiểm tra thông tin:

Nếu đúng → trả token.

Nếu sai → trả lỗi.

Frontend lưu token và chuyển hướng đến dashboard.

Ngoại lệ:

Mã lỗi	Nguyên nhân	Thông báo cho khách hàng
400	Không nhập hoặc nhập thiếu tài khoản/mật khẩu	“Vui lòng nhập đầy đủ tên đăng nhập và mật khẩu.”
401	Thông tin đăng nhập sai	“Tên đăng nhập hoặc mật khẩu không đúng.”
403	Tài khoản bị khóa	“Tài khoản của bạn đã bị khóa. Vui lòng liên hệ quản trị viên.”
503	Dịch vụ xác thực không phản hồi	“Hệ thống đang bảo trì. Vui lòng thử lại sau.”
408	Mạng yếu, phản hồi chậm	“Kết nối chậm. Vui lòng thử lại.”

2. USE CASE 2: Quên mật khẩu (AuthService)

Actor: Employee

Mô tả sơ lược: Nhân viên yêu cầu hệ thống gửi liên kết đặt lại mật khẩu đến email đăng ký.

Tiền điều kiện:

Tài khoản có email hợp lệ.

Email service hoạt động.

Hậu điều kiện:

Email khôi phục được gửi thành công.

Luồng chính:

Nhân viên nhấn “Quên mật khẩu”.

Nhập địa chỉ email.

Frontend gửi email đến AuthService qua API Gateway.

AuthService xác minh và gửi liên kết qua EmailService.

Ngoại lệ:

Mã	Nguyên nhân	Thông báo cho khách hàng
----	-------------	--------------------------

Lỗi		
400	Không nhập email hoặc sai định dạng	“Vui lòng nhập địa chỉ email hợp lệ.”
404	Email không tồn tại	“Email này không tồn tại trong hệ thống.”
429	Gửi quá nhiều yêu cầu	“Bạn đã yêu cầu quá nhiều lần. Vui lòng đợi vài phút.”
503	Dịch vụ email không phản hồi	“Không thể gửi email lúc này. Vui lòng thử lại sau.”

3. USE CASE 3: Quản lý sản phẩm (ProductService)

Actor: Employee

Mô tả sơ lược: Nhân viên thực hiện thêm, sửa hoặc xóa sản phẩm thông qua ProductService.

Tiền điều kiện:

Nhân viên đã đăng nhập và có quyền.

ProductService hoạt động.

Hậu điều kiện:

Thông tin sản phẩm được cập nhật vào cơ sở dữ liệu.

Luồng chính:

Nhân viên mở giao diện quản lý sản phẩm.

Chọn hành động (thêm/sửa/xóa).

Gửi dữ liệu qua Gateway đến ProductService.

ProductService xử lý và phản hồi.

Ngoại lệ:

Mã lỗi	Nguyên nhân	Thông báo cho khách hàng
400	Thiếu thông tin sản phẩm (tên, giá...)	“Vui lòng nhập đầy đủ thông tin sản phẩm.”
403	Không có quyền thao tác	“Bạn không có quyền thực hiện chức năng này.”
409	Mã sản phẩm bị trùng	“Mã sản phẩm đã tồn tại.”
404	Sản phẩm không tồn tại khi sửa/xóa	“Không tìm thấy sản phẩm.”
500	Lỗi ghi dữ liệu	“Hệ thống gặp sự cố. Vui lòng thử lại.”

4. USE CASE 4: Quản lý đơn hàng (OrderService)

Actor: Employee

Mô tả sơ lược: Nhân viên xác nhận, hủy hoặc hoàn đơn hàng thông qua OrderService.

Tiền điều kiện:

Đơn hàng tồn tại.

Nhân viên có quyền xử lý đơn.

Hậu điều kiện:

Trạng thái đơn hàng được cập nhật.

Luồng chính:

Nhân viên chọn đơn hàng cần xử lý.

Chọn hành động (xác nhận, hủy, hoàn).

Gửi yêu cầu đến OrderService.

Hệ thống cập nhật đơn hàng.

Ngoại lệ:

Mã lỗi	Nguyên nhân	Thông báo cho khách hàng
400	Đơn hàng ở trạng thái không cho phép xử lý	“Không thể thực hiện thao tác này với đơn hàng hiện tại.”
403	Không có quyền quản lý đơn hàng	“Bạn không được phép xử lý đơn hàng này.”
404	Đơn hàng không tồn tại	“Không tìm thấy đơn hàng.”
409	Tồn kho không đủ để xác nhận	“Không đủ số lượng hàng để xử lý đơn hàng.”
500	Lỗi hệ thống khi xử lý đơn	“Có lỗi xảy ra khi cập nhật đơn hàng.”

5. USE CASE 5: Gửi thông báo (NotificationService)

Actor: Employee

Mô tả sơ lược: Nhân viên gửi thông báo đến các khách hàng khác trong hệ thống.

Tiền điều kiện:

Nhân viên đăng nhập hợp lệ.

NotificationService hoạt động.

Hậu điều kiện:

Thông báo được gửi thành công đến người nhận.

Luồng chính:

Nhân viên truy cập chức năng gửi thông báo.

Nhập nội dung và chọn người nhận.

Gửi thông tin đến NotificationService.

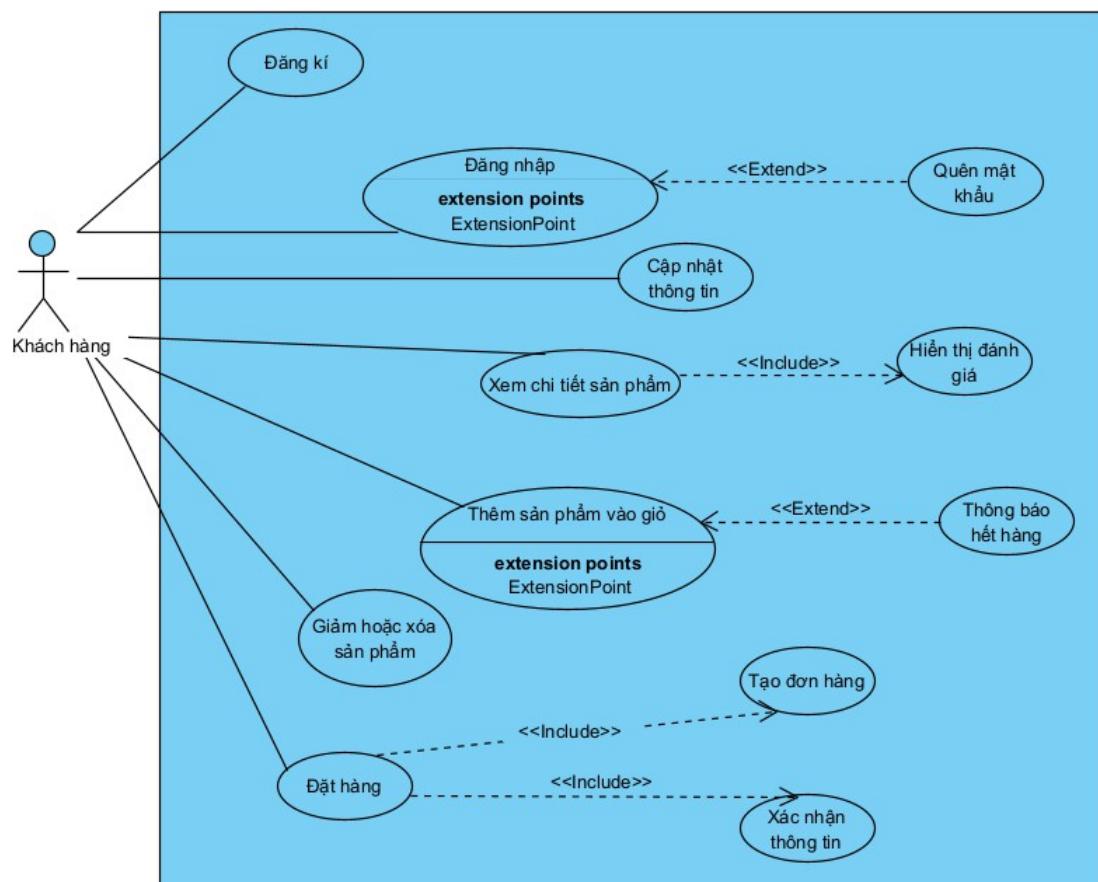
Hệ thống lưu và gửi thông báo.

Ngoại lệ:

Mã lỗi	Nguyên nhân	Thông báo cho khách hàng
400	Nội dung thông báo trống hoặc thiếu người nhận	“Vui lòng nhập nội dung và chọn người nhận.”
401	Token hết hạn hoặc không hợp lệ	“Phiên làm việc đã hết. Vui lòng đăng nhập lại.”
404	Người nhận không tồn tại	“Người nhận không tồn tại trong hệ thống.”
500	Lỗi khi gửi hoặc lưu thông báo	“Không thể gửi thông báo. Vui lòng thử lại.”

2.3.2 Biểu đồ Use Case Quản lý khách hàng

- Quản Lý Khách Hàng



- Đặc Tả Use Case**

- USE CASE 1: Đăng ký
Actor: Khách hàng

Mô tả: Cho phép khách hàng tạo tài khoản mới.

Điều kiện tiên quyết: Khách hàng chưa có tài khoản.

Kết quả: Tài khoản mới được tạo thành công.

Luồng chính:

Khách hàng chọn chức năng 'Đăng ký' từ giao diện chính.

- Hệ thống hiển thị biểu mẫu đăng ký.
- Khách hàng nhập các thông tin yêu cầu: họ tên, email, mật khẩu,...
- Khách hàng nhấn nút 'Gửi' hoặc 'Đăng ký'.
- Hệ thống kiểm tra tính hợp lệ và duy nhất của email.
- Nếu hợp lệ, hệ thống tạo tài khoản mới và thông báo đăng ký thành công.
- Hệ thống chuyển đến trang đăng nhập hoặc trang chính của khách hàng.

Ngoại lệ:

- Email đã tồn tại -> hiển thị thông báo lỗi.
- Nhập thiếu thông tin -> yêu cầu nhập lại.

2. USE CASE 2: Đăng nhập

Actor: Khách hàng

Mô tả: Cho phép khách hàng đăng nhập hệ thống.

Điều kiện tiên quyết: Đã có tài khoản hợp lệ.

Kết quả: Khách hàng được đăng nhập thành công.

Luồng chính:

- Khách hàng chọn chức năng 'Đăng nhập'.
- Hệ thống hiển thị biểu mẫu đăng nhập.
- Khách hàng nhập tên đăng nhập và mật khẩu.
- Khách hàng nhấn nút 'Đăng nhập'.
- 5. Hệ thống kiểm tra thông tin đăng nhập.
- Nếu thông tin hợp lệ, hệ thống cho phép truy cập và chuyển đến trang chính.
 - Nếu không, hiển thị thông báo lỗi và cho phép thử lại.

Ngoại lệ:

- Sai thông tin -> hiển thị thông báo lỗi.

3. USE CASE 3: Cập nhật thông tin

Actor: Khách hàng

Mô tả: Cho phép cập nhật hồ sơ cá nhân.

Điều kiện tiên quyết: Đã đăng nhập.

Kết quả: Thông tin được cập nhật.

Luồng chính:

- Khách hàng đăng nhập và truy cập vào mục 'Thông tin cá nhân'.
- 2. Hệ thống hiển thị biểu mẫu với thông tin hiện tại.
- Khách hàng sửa đổi thông tin (ví dụ: địa chỉ, số điện thoại,...).

- Khách hàng nhấn nút 'Lưu'.
- Hệ thống kiểm tra dữ liệu hợp lệ và cập nhật vào cơ sở dữ liệu.
- Hệ thống hiển thị thông báo 'Cập nhật thành công'.

Ngoại lệ:

- Khách hàng để trống các thông tin

4. USE CASE 4: Xem chi tiết sản phẩm

Actor: Khách hàng

Mô tả: Hiển thị thông tin chi tiết về sản phẩm.

Điều kiện tiên quyết: Có danh sách khách hàng để khách hàng duyệt

Kết quả: Hiển thị thông tin sản phẩm.

Luồng chính:

- Khách hàng duyệt danh sách sản phẩm.
- 2. Khách hàng chọn một sản phẩm cụ thể.
- 3. Hệ thống hiển thị trang chi tiết sản phẩm.
- Hệ thống hiển thị đánh giá và nhận xét từ khách hàng khác.

Ngoại lệ:

- Chưa có đánh giá sản phẩm

5. USE CASE 5: Thêm sản phẩm vào giỏ

Actor: Khách hàng

Mô tả: Thêm một sản phẩm vào giỏ hàng.

Điều kiện tiên quyết: Sản phẩm còn hàng.

Kết quả: Sản phẩm được thêm vào giỏ.

Luồng chính:

- Khách hàng chọn một sản phẩm từ danh sách.
- Nhấn nút 'Thêm vào giỏ hàng'.
- 3. Hệ thống kiểm tra số lượng tồn kho.
- 4. Nếu còn hàng, thêm sản phẩm vào giỏ và thông báo thành công.
- Nếu hết hàng, hệ thống hiển thị thông báo 'Sản phẩm đã hết hàng'.

Ngoại lệ:

- Sản phẩm hết hàng -> không thể thêm vào giỏ.

6. USE CASE 6: Giảm hoặc xóa sản phẩm khỏi giỏ

Actor: Khách hàng

Mô tả: Cho phép chỉnh sửa giỏ hàng.

Điều kiện tiên quyết: Có sản phẩm trong giỏ.

Kết quả: Sản phẩm được cập nhật/xóa.

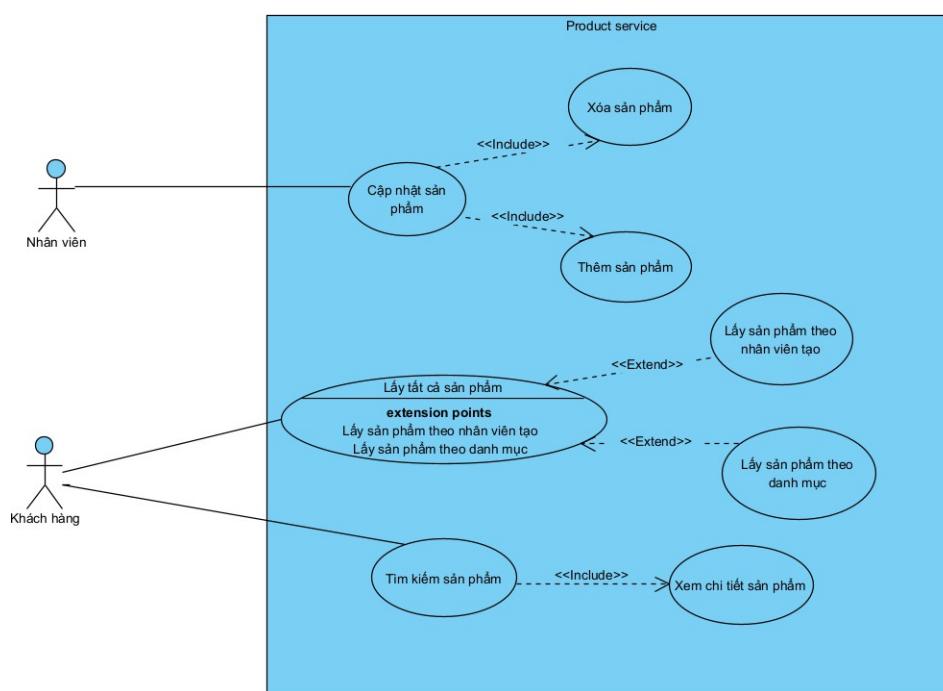
Luồng chính:

- Khách hàng vào trang 'Giỏ hàng'.

- 2. Hệ thống hiển thị danh sách sản phẩm đã thêm.
 - 3. Khách hàng chọn sản phẩm cần chỉnh sửa.
 - 4. Khách hàng chọn giảm số lượng hoặc nhấn 'Xóa'.
 - 5. Hệ thống cập nhật giỏ hàng tương ứng và hiển thị giỏ hàng mới.
7. Use Case 7: Đặt hàng
- Actor:** Khách hàng
- Mô tả:** Đặt mua sản phẩm đã chọn.
- Điều kiện tiên quyết:** Có sản phẩm trong giỏ.
- Kết quả:** Đơn hàng được ghi nhận.
- Luồng chính:**
- Khách hàng vào giỏ hàng và chọn 'Thanh toán'.
 - 2. Hệ thống yêu cầu xác nhận thông tin giao hàng.
 - 3. Khách hàng xác nhận thông tin và tiếp tục.
 - 4. Hệ thống tạo đơn hàng với các sản phẩm trong giỏ.
 - 5. Hệ thống hiển thị mã đơn hàng và thông báo 'Đặt hàng thành công'.
- Ngoại lệ:**
- Thông tin người nhận không hợp lệ -> yêu cầu sửa lại.

2.3.3 Biểu đồ Use Case Quản lý sản phẩm

- Quản Lý Sản Phẩm



- Đặc Tả Use Case

1. USE CASE: Cập nhật sản phẩm

- **Actor:** Nhân viên
- **Mô tả:** Nhân viên cập nhật thông tin của một sản phẩm đã có trong hệ thống.
- **Tiền điều kiện:** Nhân viên đã đăng nhập, sản phẩm cần cập nhật đã tồn tại.
- **Luồng chính:**
 - Nhân viên gửi yêu cầu cập nhật sản phẩm (bao gồm ID sản phẩm và thông tin mới).
 - Hệ thống xác thực quyền Nhân viên.
 - Gọi Microservice Quản lý Sản phẩm (Product Service) để cập nhật thông tin sản phẩm trong cơ sở dữ liệu.
 - Product Service trả về kết quả thành công hoặc lỗi (nếu sản phẩm không tồn tại).
 - Hệ thống thông báo kết quả cho Nhân viên.
- **Luồng phụ:** Nếu sản phẩm không tồn tại, trả về lỗi "Sản phẩm không tìm thấy".
- **Hậu điều kiện:** Thông tin sản phẩm được cập nhật trong hệ thống.
- **Tương tác Microservice:**
 - API gọi: PUT /products/{id}
 - Microservice: Product Service
 - Input: ID sản phẩm, thông tin cập nhật (tên, giá, mô tả,...)
 - Output: Thông báo thành công hoặc lỗi.

2. USE CASE: Thêm sản phẩm

- **Actor:** Nhân viên
- **Mô tả:** Nhân viên thêm một sản phẩm mới vào hệ thống.
- **Tiền điều kiện:** Nhân viên đã đăng nhập.
- **Luồng chính:**
 - Nhân viên gửi yêu cầu thêm sản phẩm với thông tin (tên, giá, danh mục, ...).
 - Hệ thống xác thực quyền Nhân viên.
 - Gọi Microservice Quản lý Sản phẩm (Product Service) để lưu sản phẩm mới vào cơ sở dữ liệu.
 - Product Service trả về kết quả thành công hoặc lỗi (nếu dữ liệu không hợp lệ).
 - Hệ thống thông báo kết quả cho Nhân viên.
- **Luồng phụ:** Nếu thông tin không hợp lệ (ví dụ: giá âm), trả về lỗi.
- **Hậu điều kiện:** Sản phẩm mới được thêm vào hệ thống.
- **Tương tác Microservice:**
 - API gọi: POST /products
 - Microservice: Product Service

- Input: Thông tin sản phẩm (tên, giá, ...)
- Output: ID sản phẩm mới hoặc lỗi.

3. USE CASE: Lấy tất cả sản phẩm

- **Actor:** Nhân viên
- **Mô tả:** Nhân viên lấy danh sách tất cả sản phẩm trong hệ thống.
- **Tiền điều kiện:** Nhân viên đã đăng nhập.
- **Luồng chính:**
 - Nhân viên gửi yêu cầu lấy tất cả sản phẩm.
 - Hệ thống xác thực quyền Nhân viên.
 - Gọi Microservice Quản lý Sản phẩm (Product Service) để lấy danh sách sản phẩm.
 - Product Service trả về danh sách sản phẩm.
 - Hệ thống hiển thị danh sách cho Nhân viên.
- **Luồng phụ:** Nếu không có sản phẩm nào, trả về danh sách rỗng.
- **Hậu điều kiện:** Nhân viên nhận được danh sách sản phẩm.
- **Tương tác Microservice:**
 - API gọi: GET /products
 - Microservice: Product Service
 - Input: Không có
 - Output: Danh sách sản phẩm.

4. USE CASE: Tìm kiếm sản phẩm

- **Actor:** Khách hàng
- **Mô tả:** Khách hàng tìm kiếm sản phẩm theo từ khóa (tên, danh mục, ...).
- **Tiền điều kiện:** Không có.
- **Luồng chính:**
 - Khách hàng nhập từ khóa tìm kiếm và gửi yêu cầu.
 - Hệ thống gọi Microservice Tìm kiếm (Search Service) để tìm kiếm sản phẩm theo từ khóa.
 - Search Service gọi Microservice Quản lý Sản phẩm (Product Service) để lấy dữ liệu sản phẩm khớp với từ khóa.
 - Hệ thống hiển thị kết quả tìm kiếm.
- **Luồng phụ:** Nếu không tìm thấy sản phẩm, trả về danh sách rỗng.
- **Hậu điều kiện:** Khách hàng nhận được danh sách sản phẩm khớp với từ khóa.
- **Tương tác Microservice:**
 - API gọi: GET /search?query={keyword}
 - Microservice: Search Service → Product Service
 - Input: Từ khóa tìm kiếm.
 - Output: Danh sách sản phẩm khớp.

5. USE CASE: Xem chi tiết sản phẩm

- **Actor:** Khách hàng
- **Mô tả:** Khách hàng xem thông tin chi tiết của một sản phẩm.
- **Tiền điều kiện:** Sản phẩm tồn tại.
- **Luồng chính:**
 - Khách hàng gửi yêu cầu xem chi tiết sản phẩm (theo ID).
 - Hệ thống gọi Microservice Quản lý Sản phẩm (Product Service) để lấy thông tin sản phẩm.
 - Product Service trả về thông tin chi tiết (tên, giá, mô tả, ...).
 - Hệ thống hiển thị thông tin cho Khách hàng.
- **Luồng phụ:** Nếu sản phẩm không tồn tại, trả về lỗi "Sản phẩm không tìm thấy".
- **Hậu điều kiện:** Khách hàng nhận được thông tin chi tiết sản phẩm.
- **Tương tác Microservice:**
 - API gọi: GET /products/{id}
 - Microservice: Product Service
 - Input: ID sản phẩm.
 - Output: Thông tin chi tiết sản phẩm hoặc lỗi.

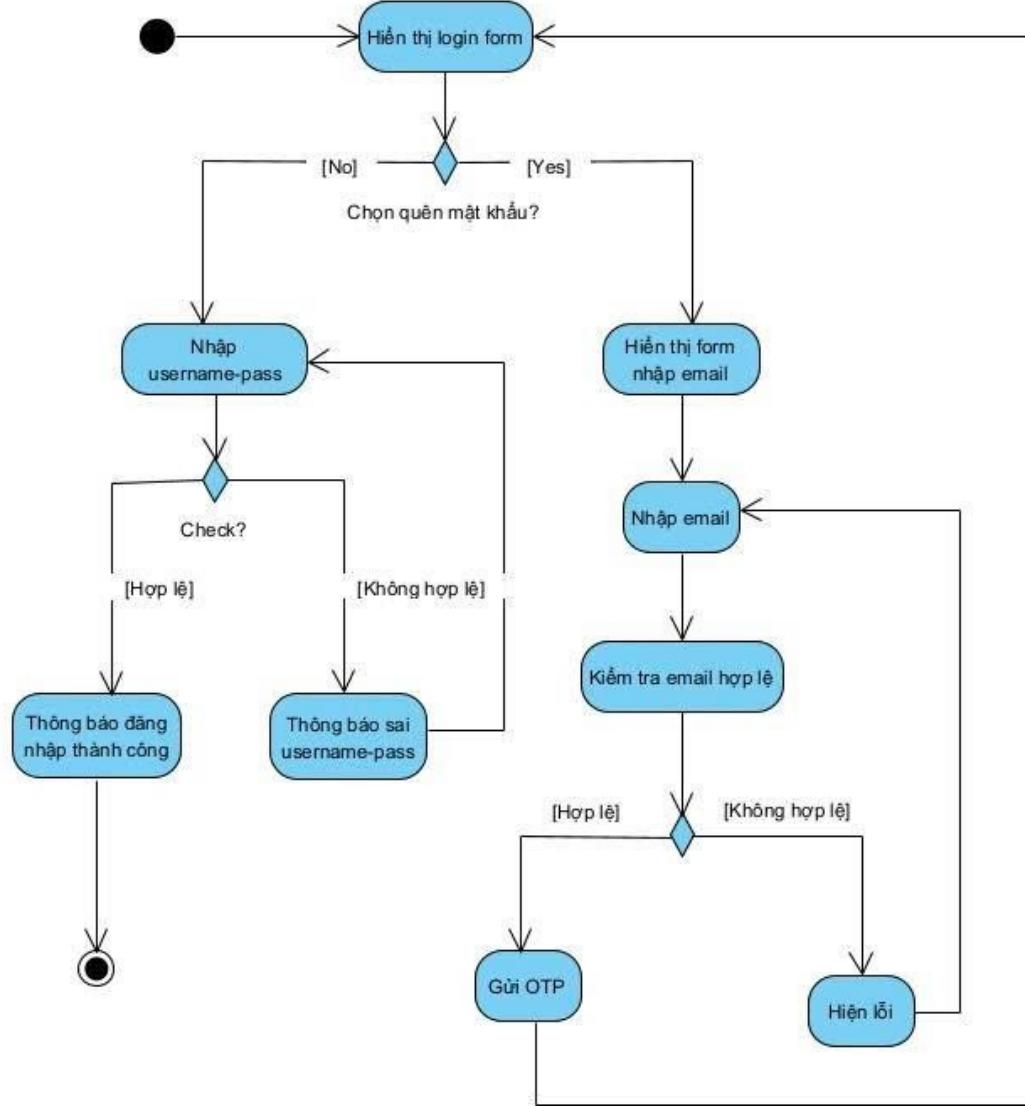
6. USE CASE: Lấy sản phẩm theo danh mục

- **Actor:** Khách hàng
- **Mô tả:** Khách hàng lấy danh sách sản phẩm theo danh mục cụ thể.
- **Tiền điều kiện:** Danh mục tồn tại.
- **Luồng chính:**
 - Khách hàng chọn danh mục và gửi yêu cầu.
 - Hệ thống gọi Microservice Quản lý Sản phẩm (Product Service) để lấy danh sách sản phẩm thuộc danh mục.
 - Product Service trả về danh sách sản phẩm.
 - Hệ thống hiển thị danh sách cho Khách hàng.
- **Luồng phụ:** Nếu danh mục không tồn tại hoặc không có sản phẩm, trả về danh sách rỗng.
- **Hậu điều kiện:** Khách hàng nhận được danh sách sản phẩm theo danh mục.
- **Tương tác Microservice:**
 - API gọi: GET /products?category={categoryId}
 - Microservice: Product Service
 - Input: ID danh mục.
 - Output: Danh sách sản phẩm thuộc danh mục.

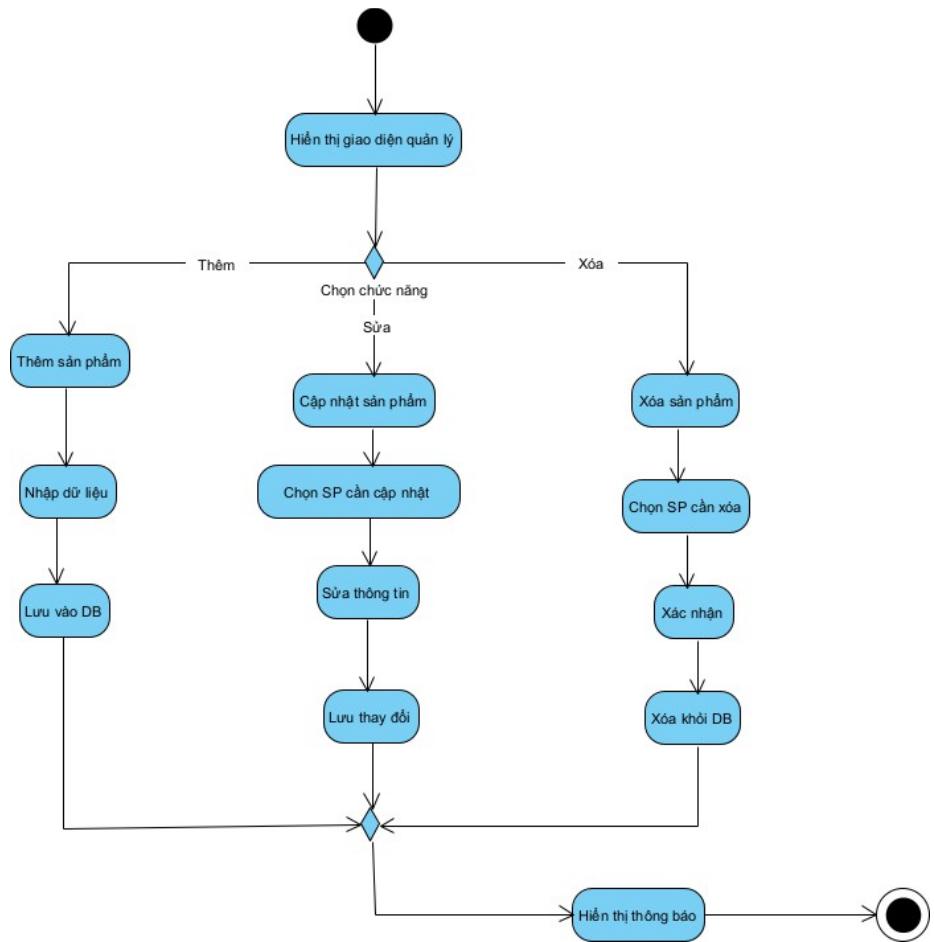
2.4 Biểu đồ hoạt động (Activity Diagram)

- **Quản Lý Nhân Viên**

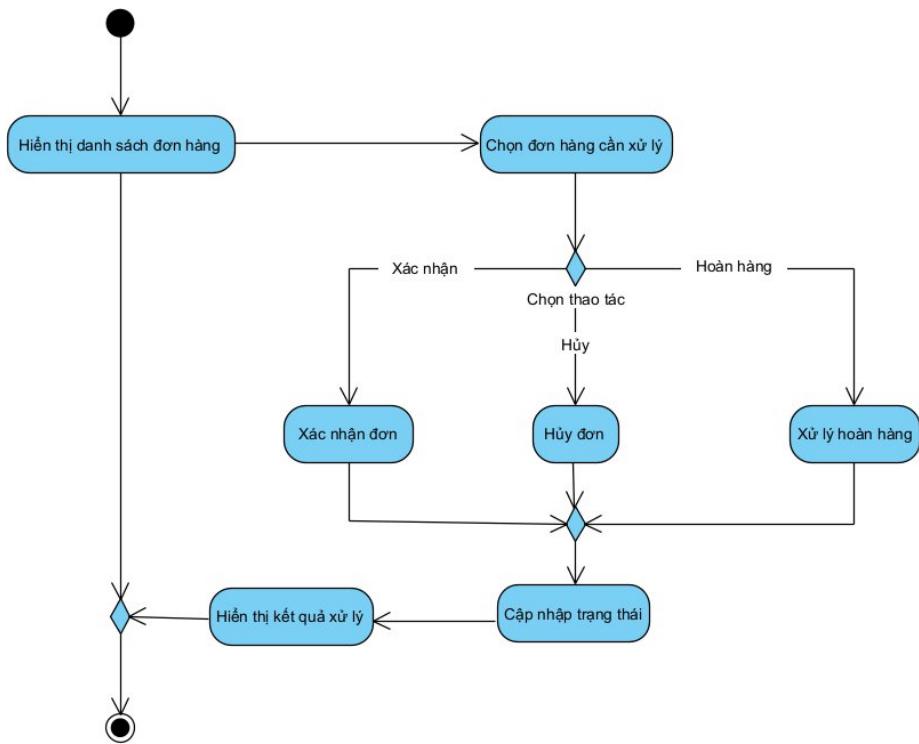
1. Đăng nhập



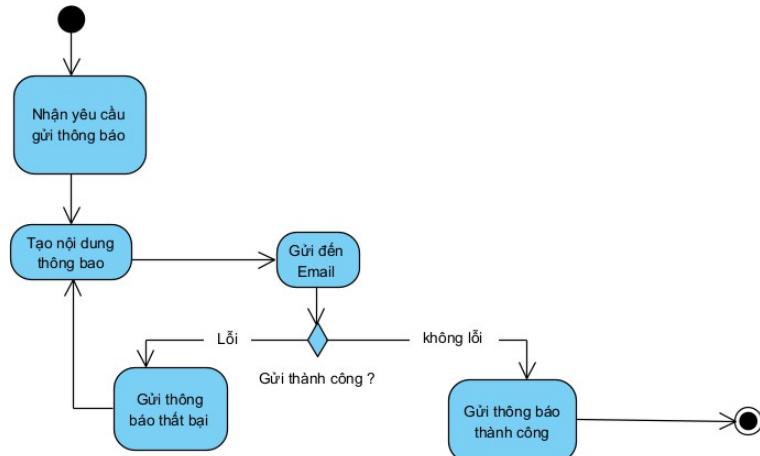
2. Quản lý sản phẩm



3. Quản lý đơn hàng

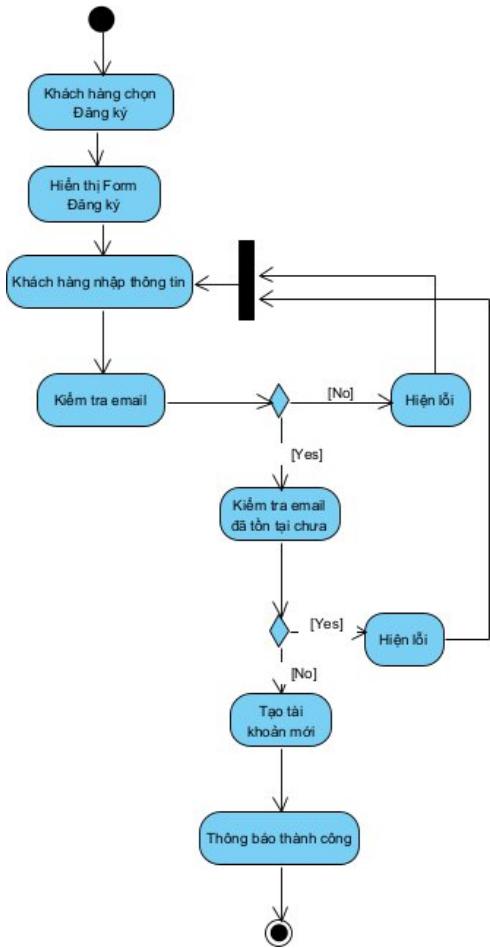


4. Gửi thông báo

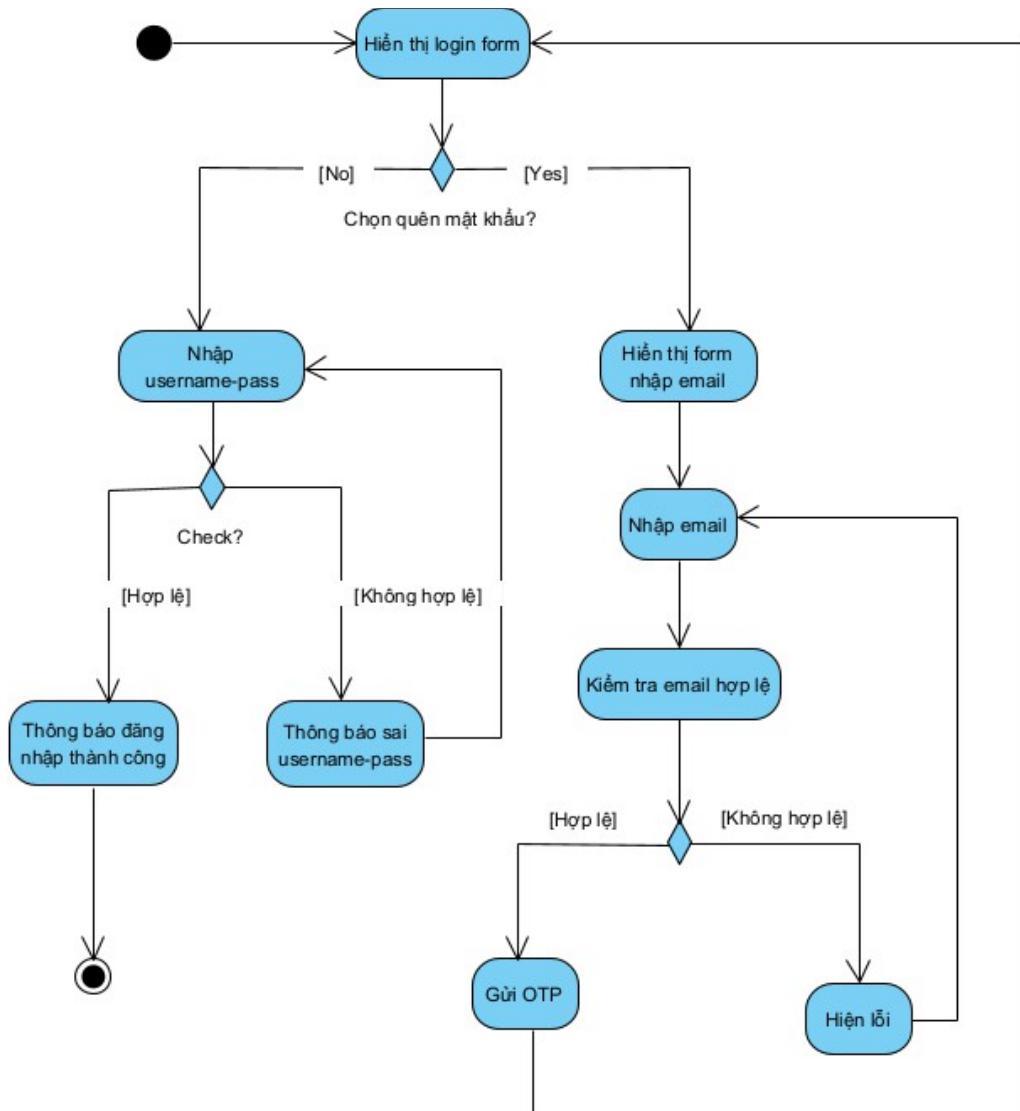


- Quản Lý Khách Hàng**

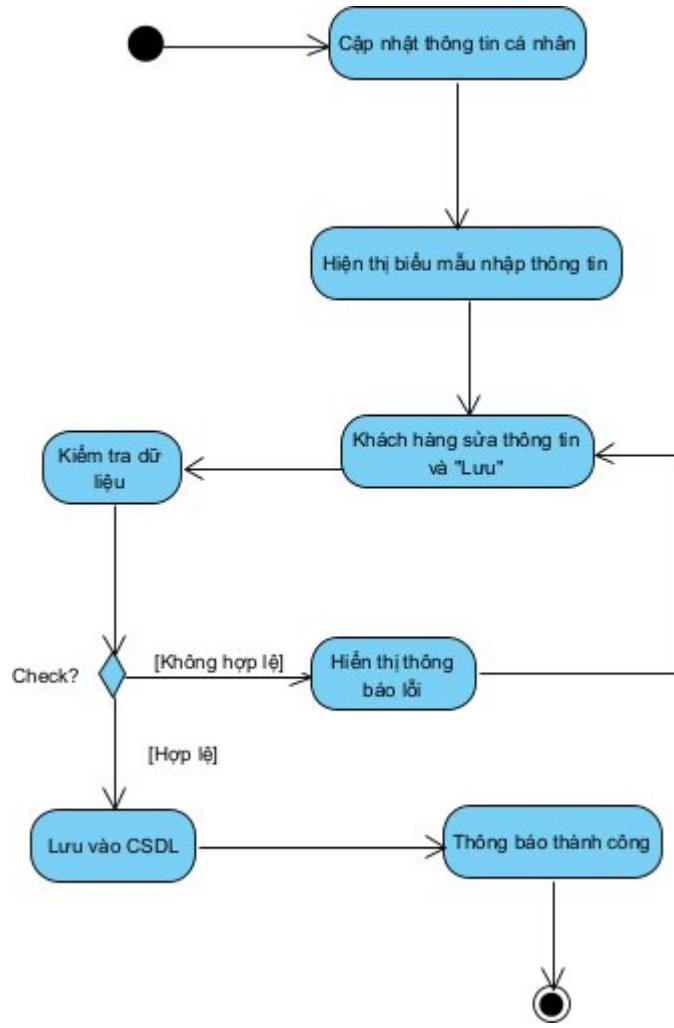
- Đăng ký



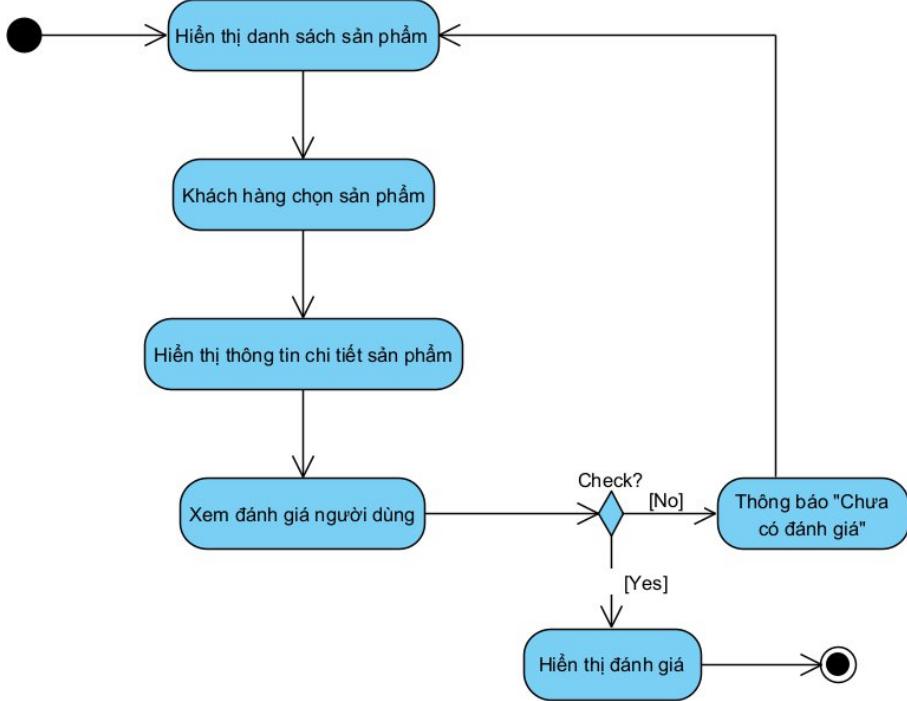
2. Đăng nhập



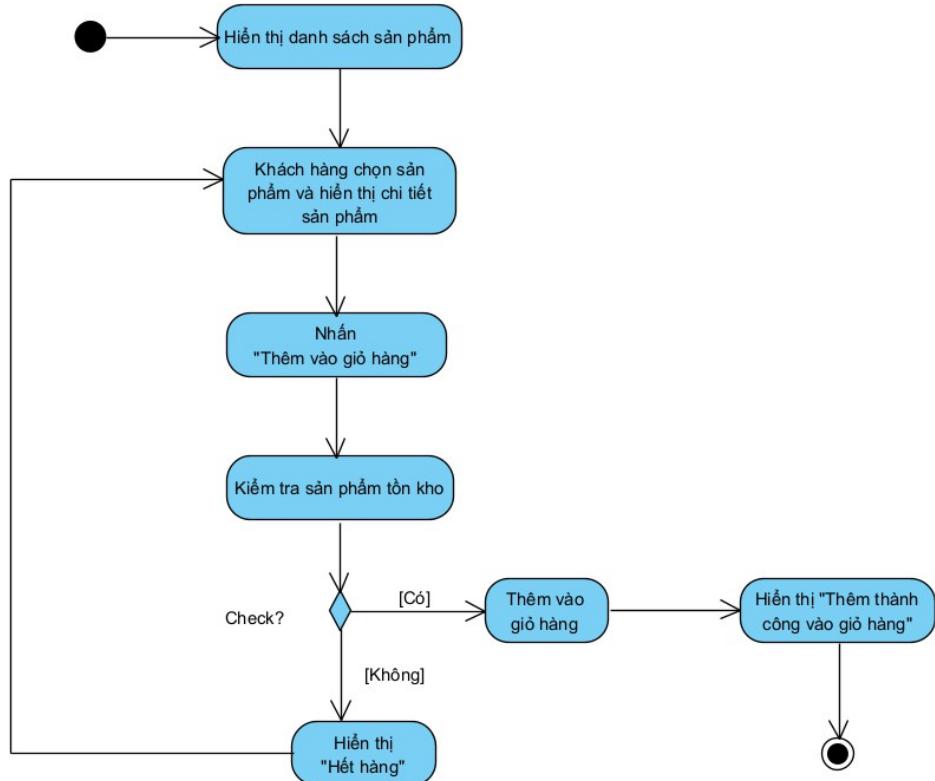
3. Cập nhật thông tin



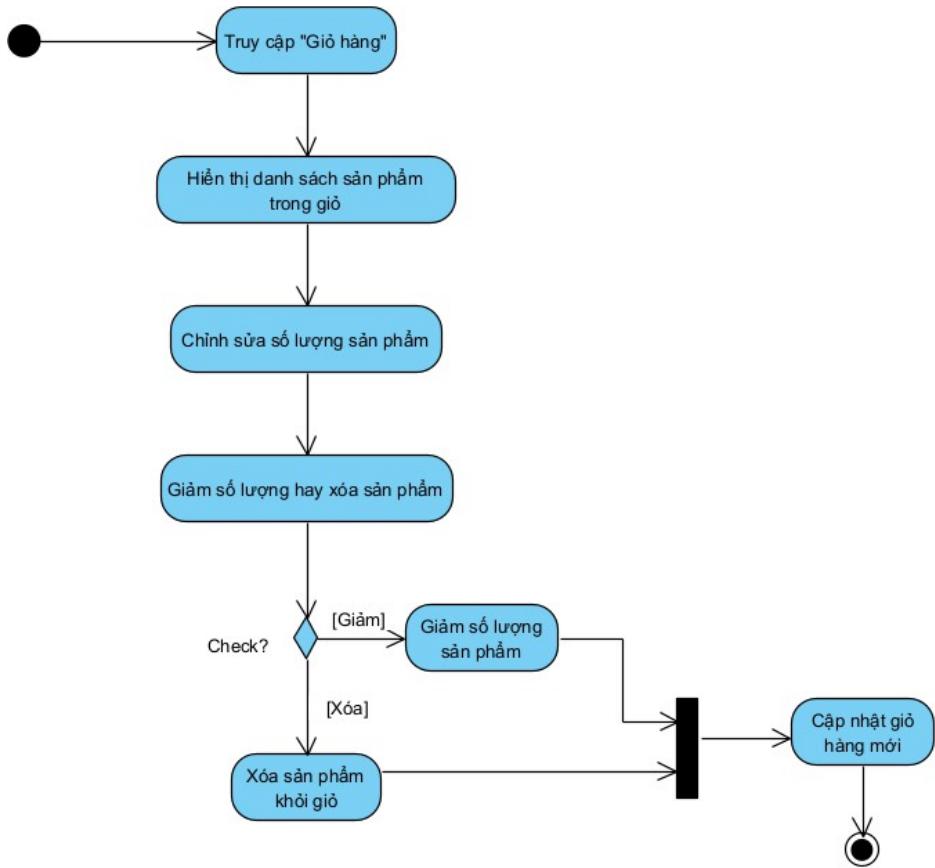
4. Xem chi tiết sản phẩm



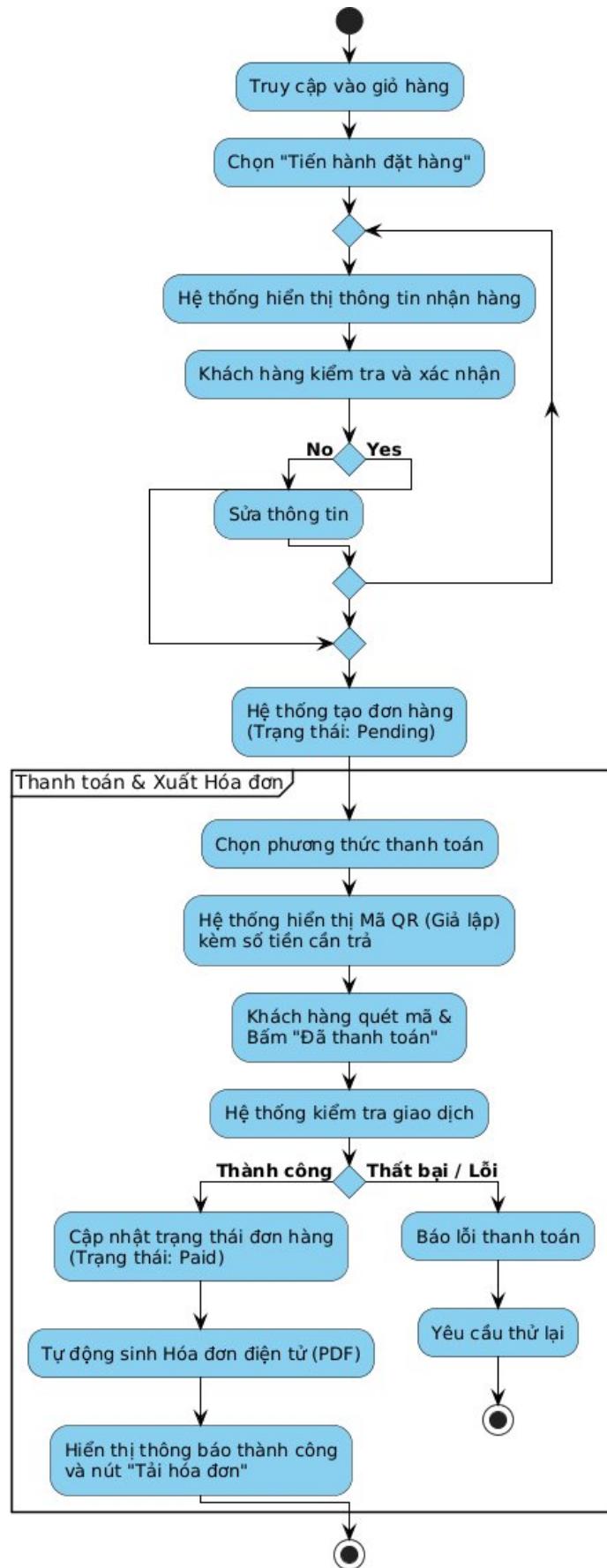
5. Thêm sản phẩm vào giỏ hàng



6. Giảm hoặc xóa sản phẩm khỏi giỏ

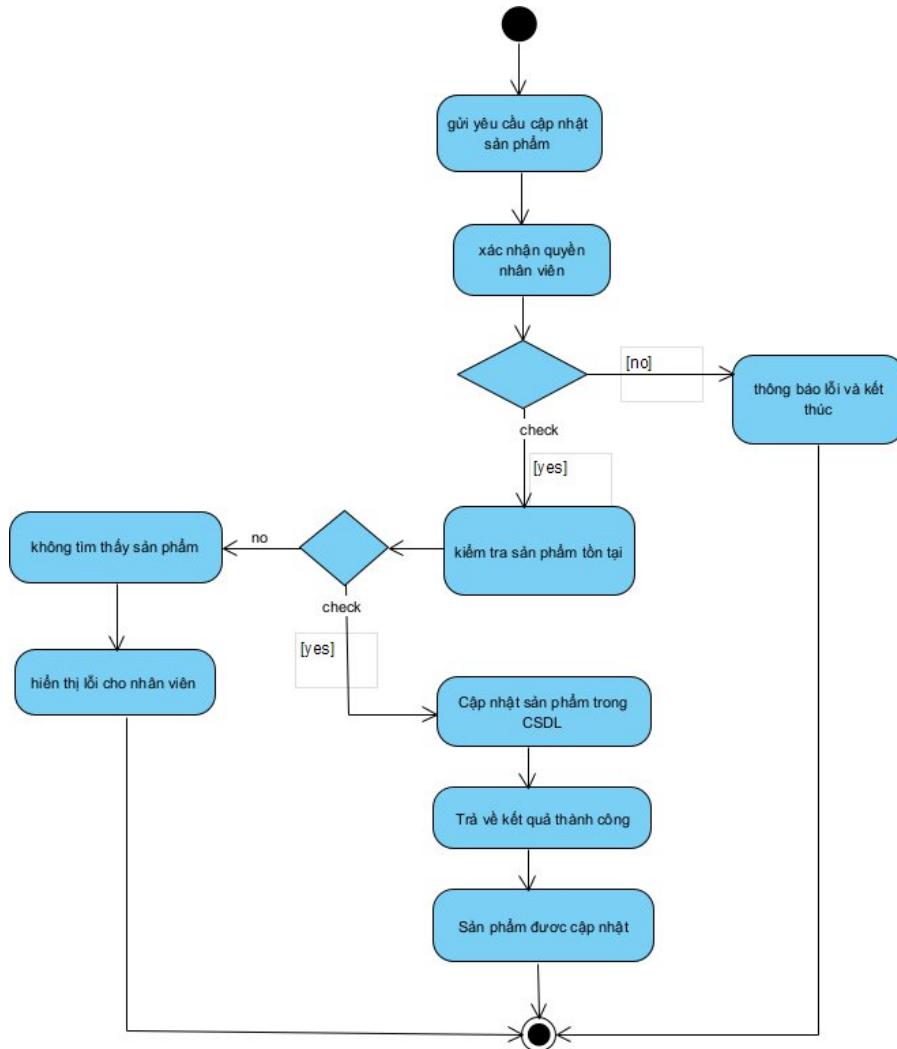


7. Đặt hàng

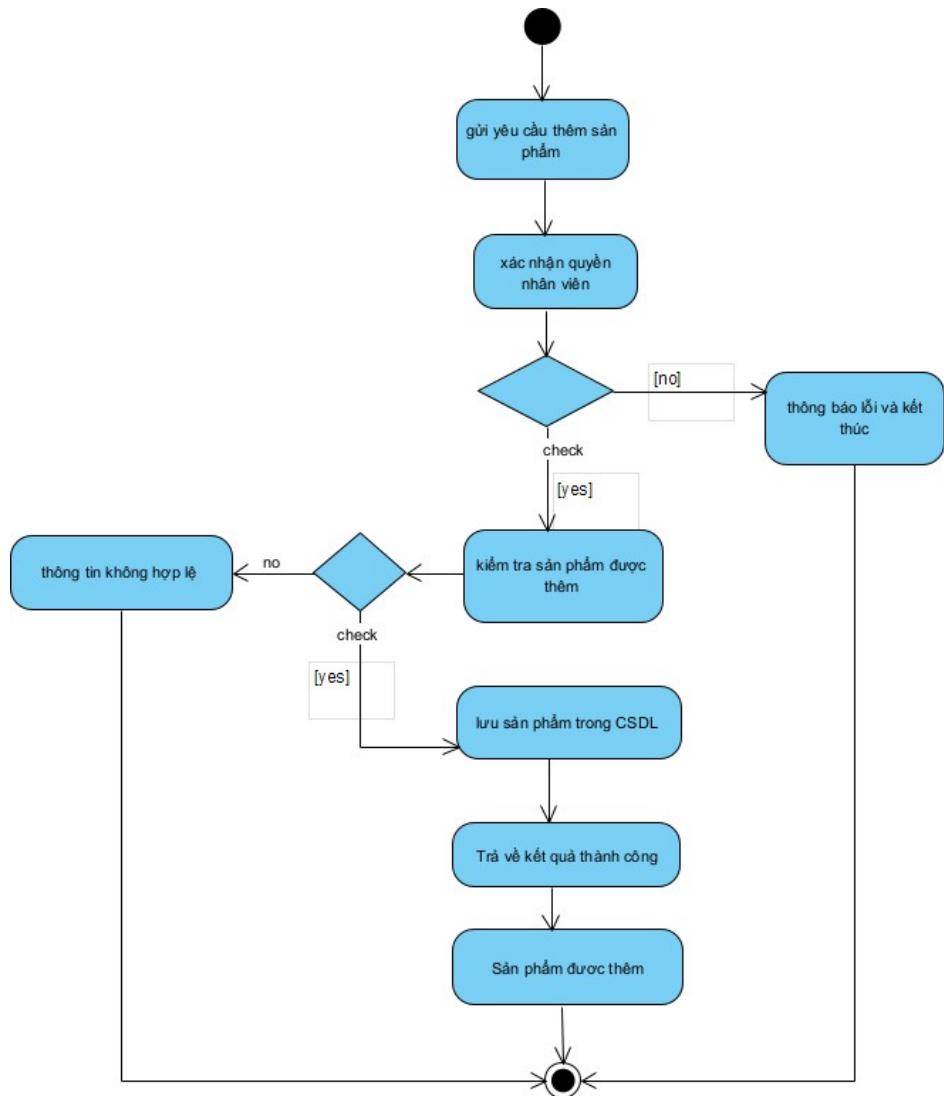


- Quản Lý Sản Phẩm

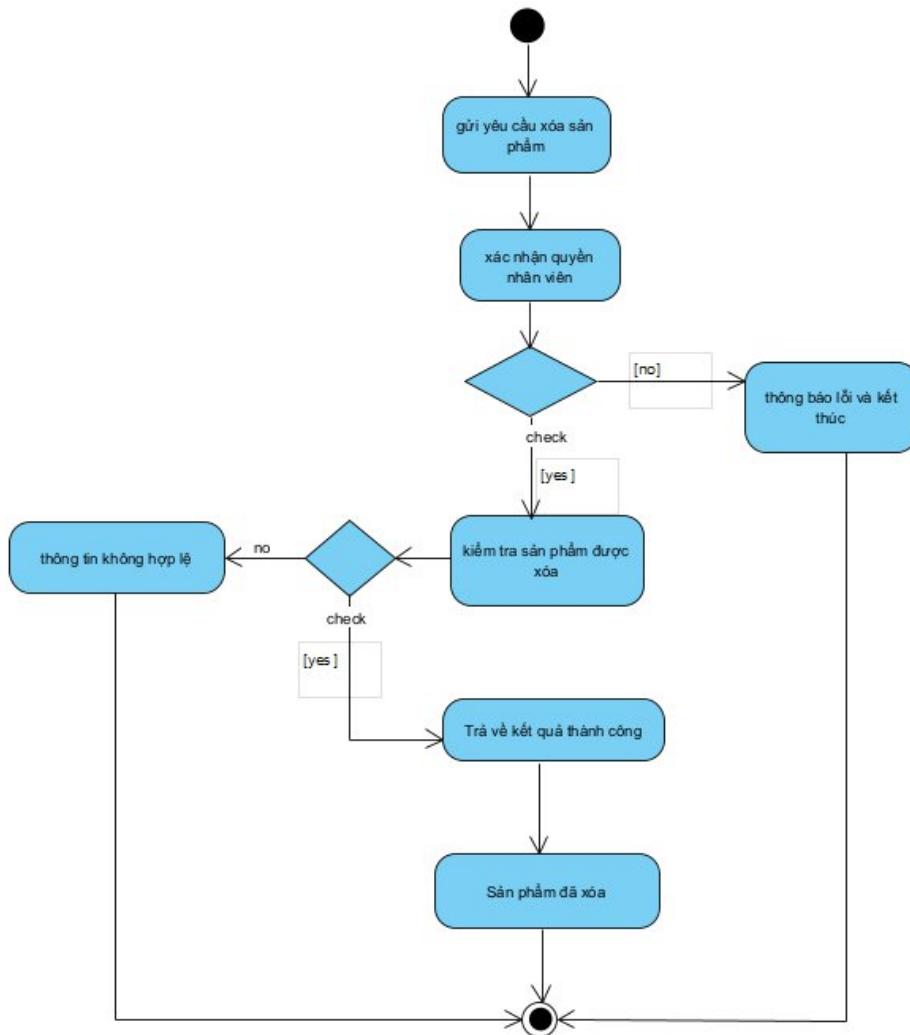
1. Cập nhật sản phẩm



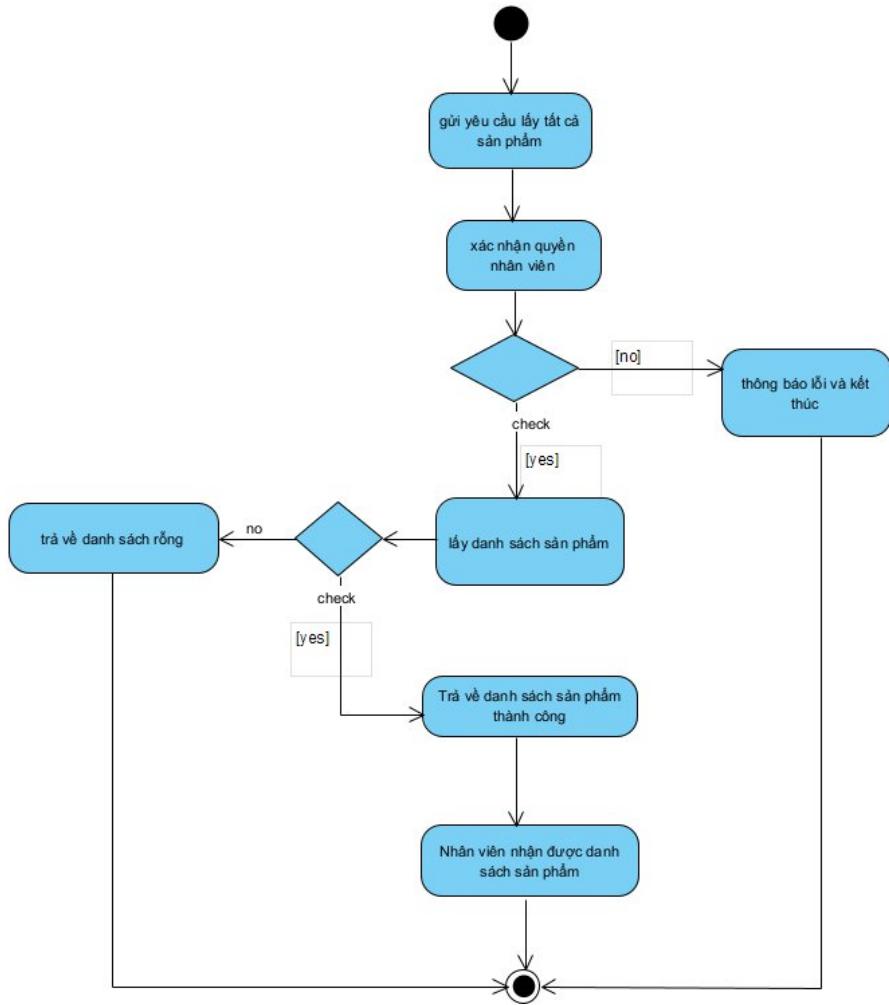
2. Thêm sản phẩm



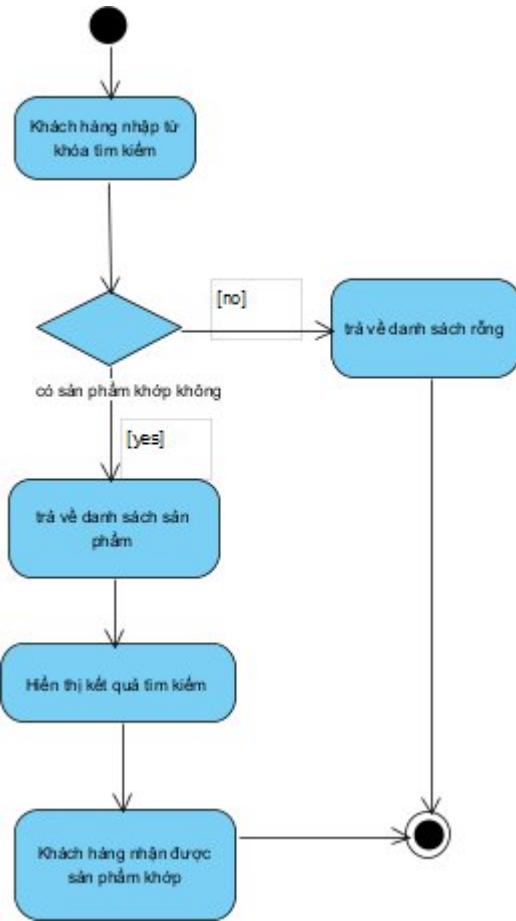
3. Xóa sản phẩm



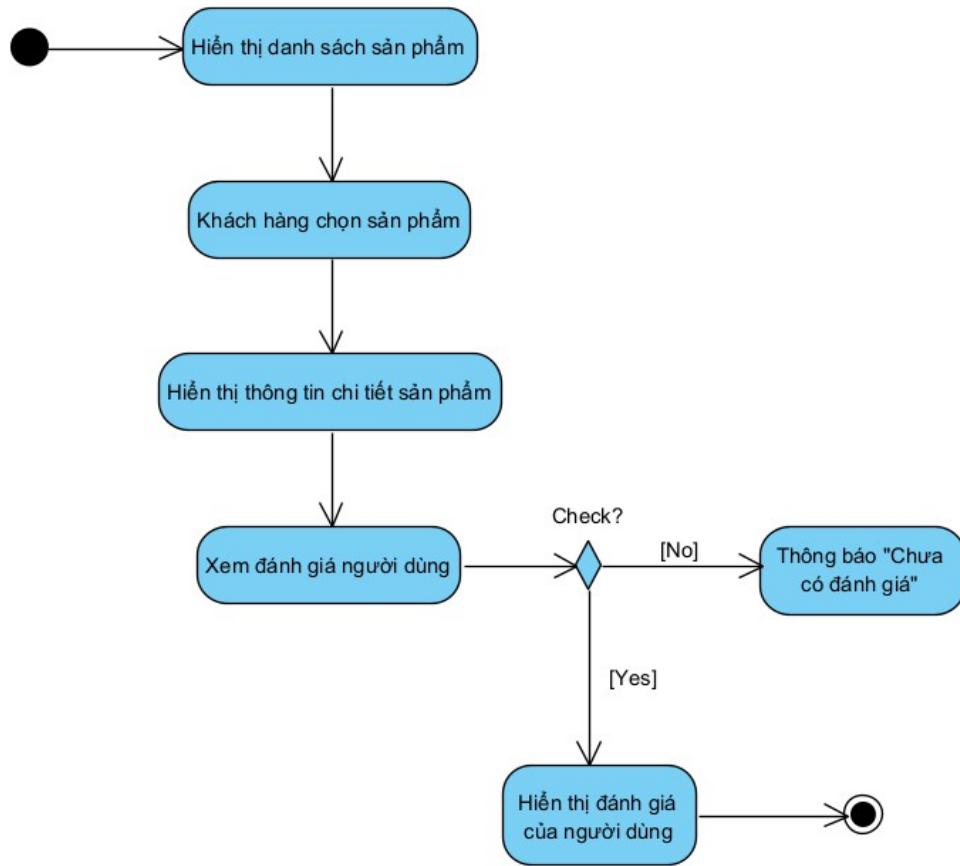
4. Lấy tất cả sản phẩm



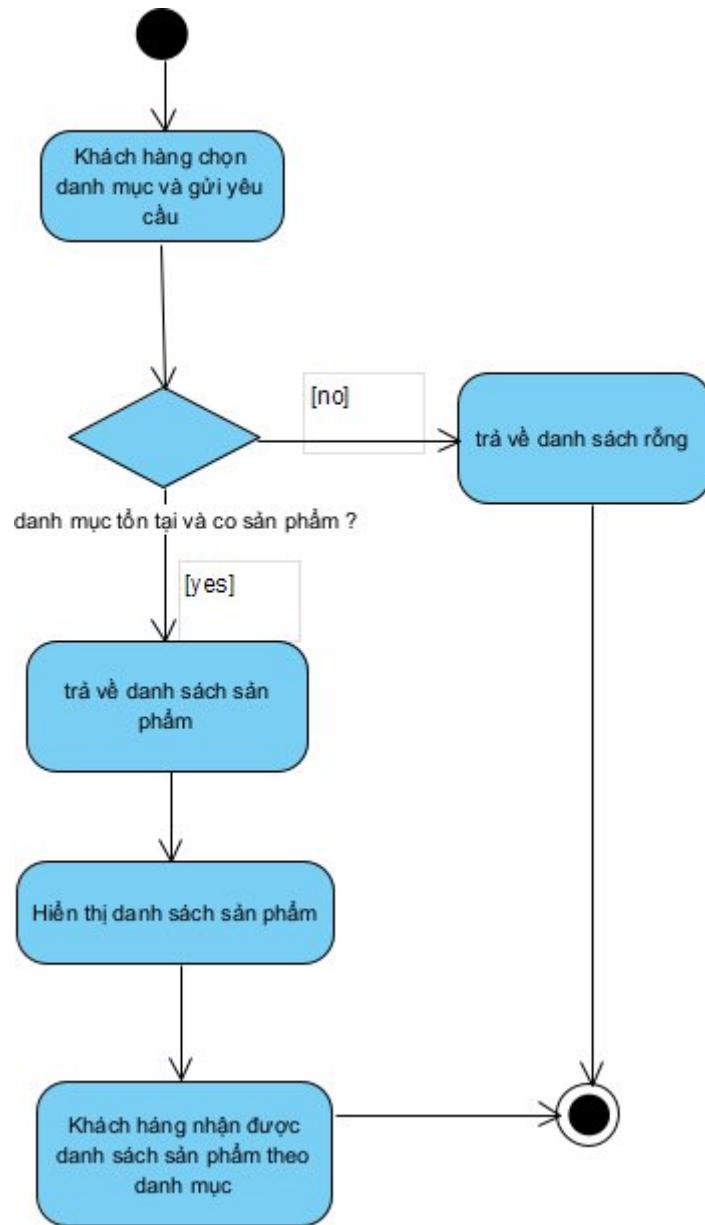
5. Tìm kiếm sản phẩm



6. Xem chi tiết sản phẩm



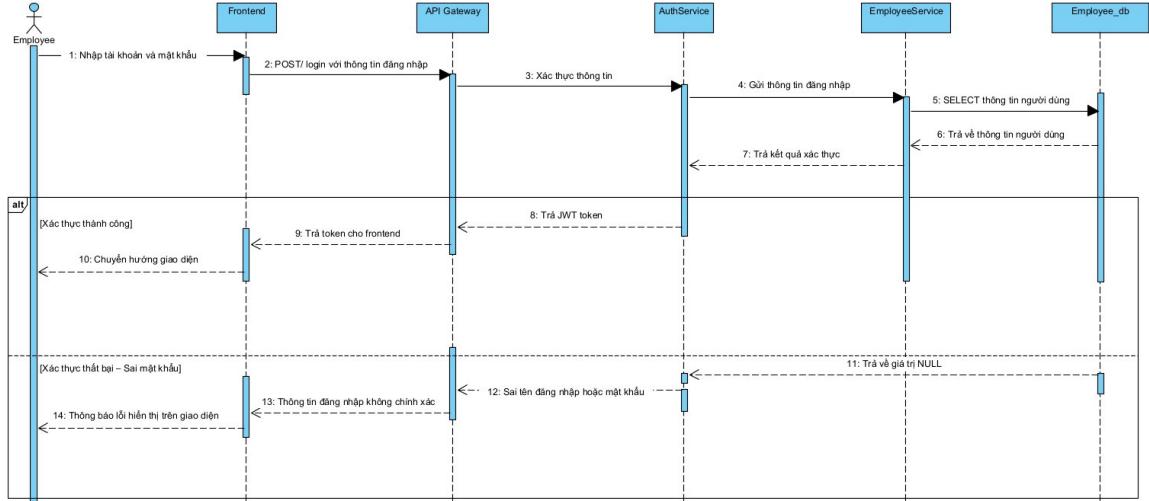
7. Lấy sản phẩm theo danh mục



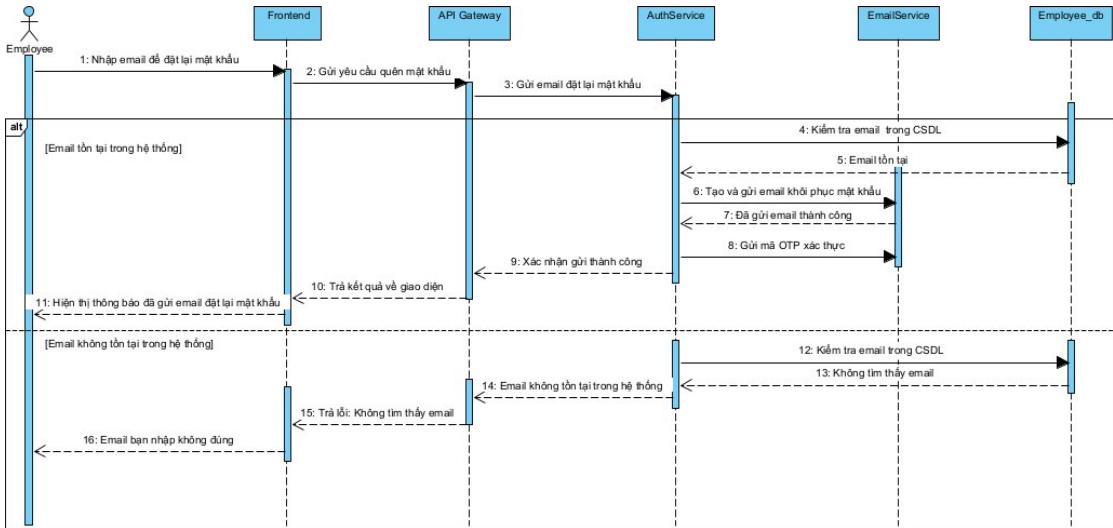
2.5 Biểu đồ tuần tự (Sequence Diagram)

- Quản Lý Nhân Viên

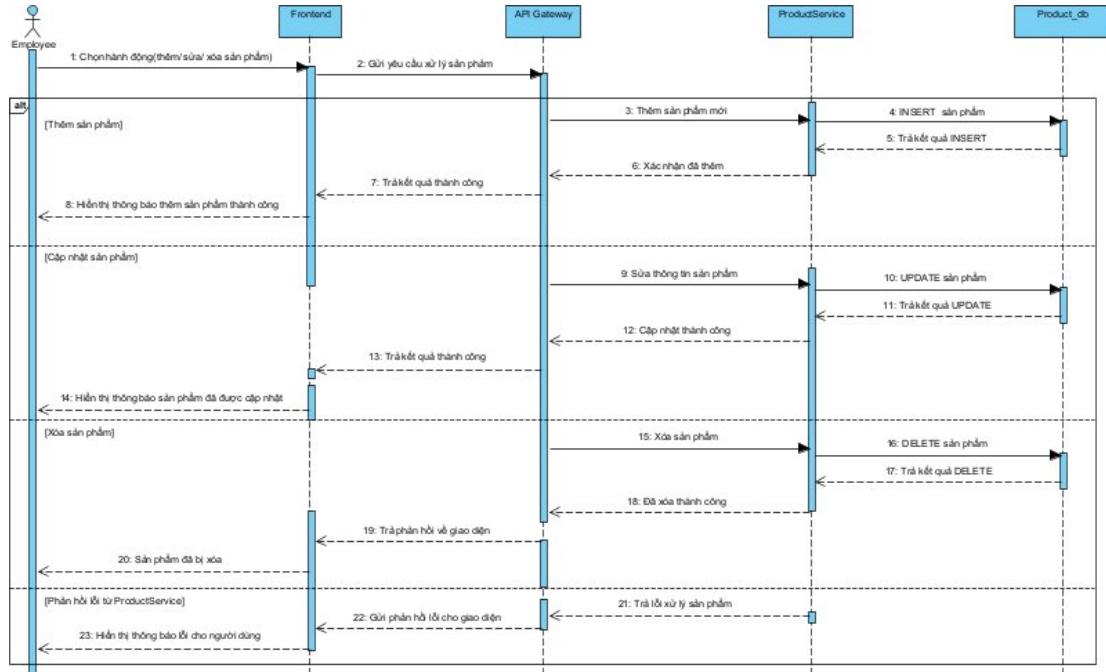
1. Đăng nhập



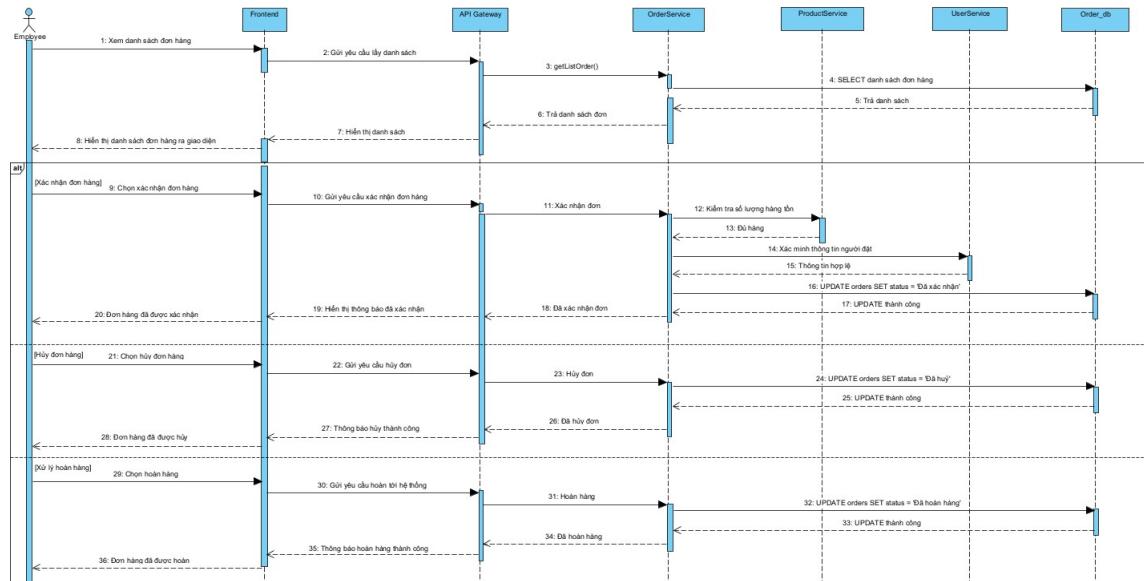
2. Quên mật khẩu



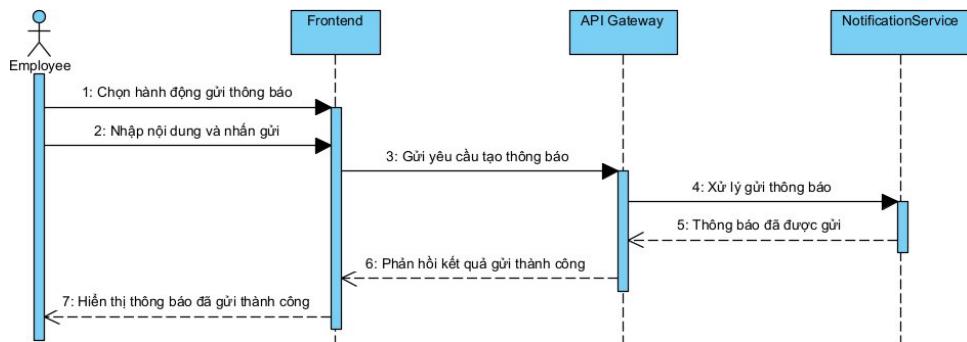
3. Quản lý sản phẩm



4. Quản lý đơn hàng

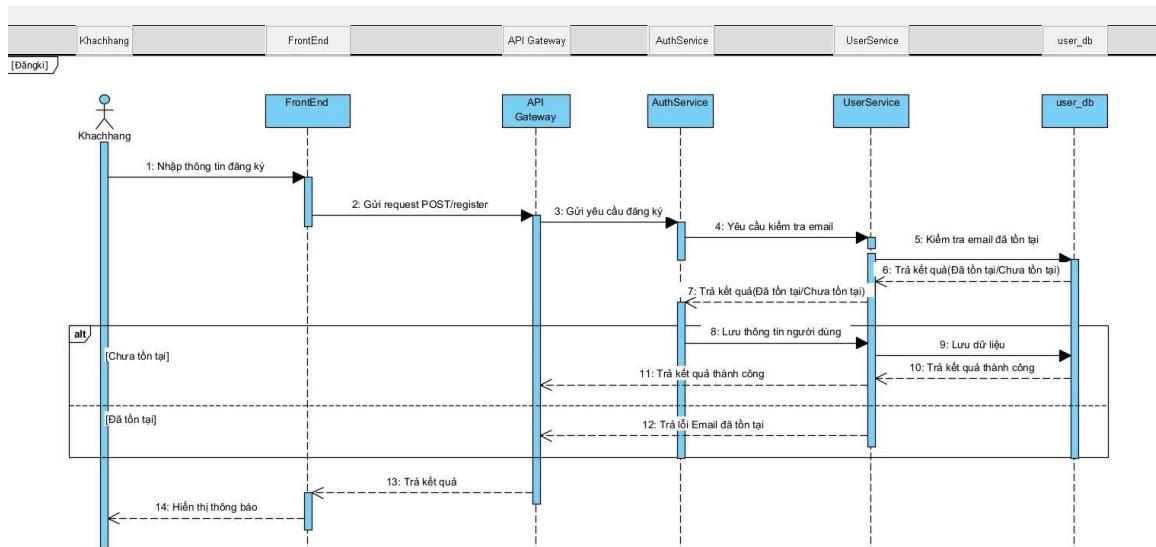


5. Gửi thông báo

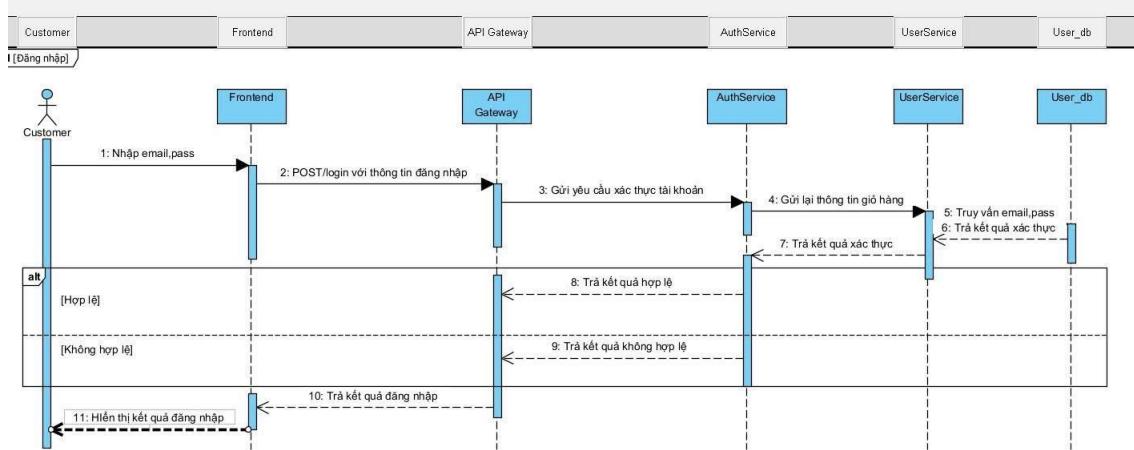


• Quản Lý Khách Hàng

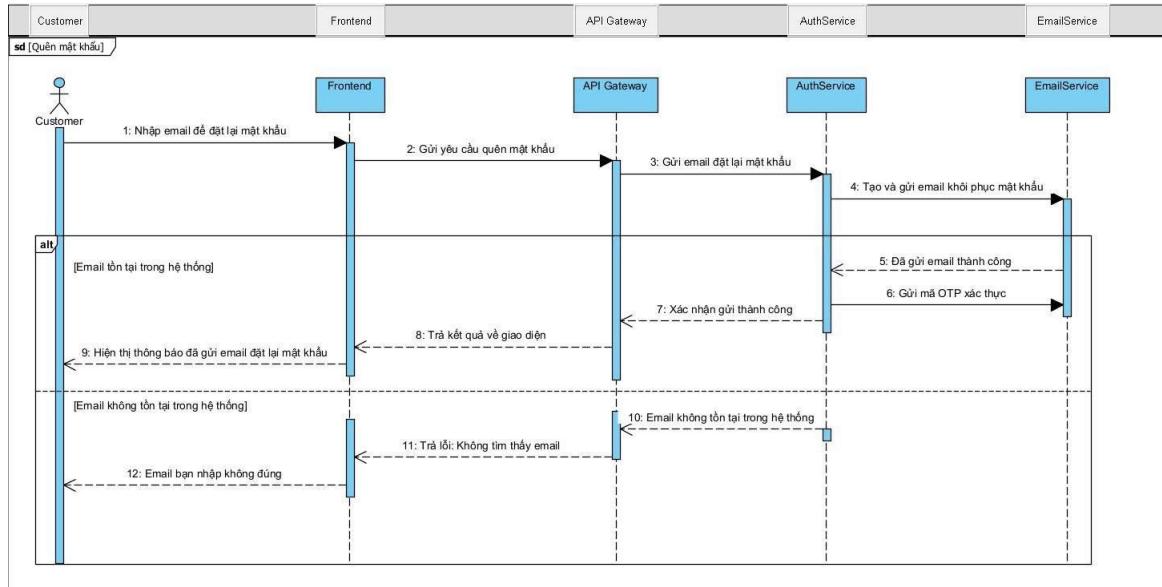
1. Đăng kí



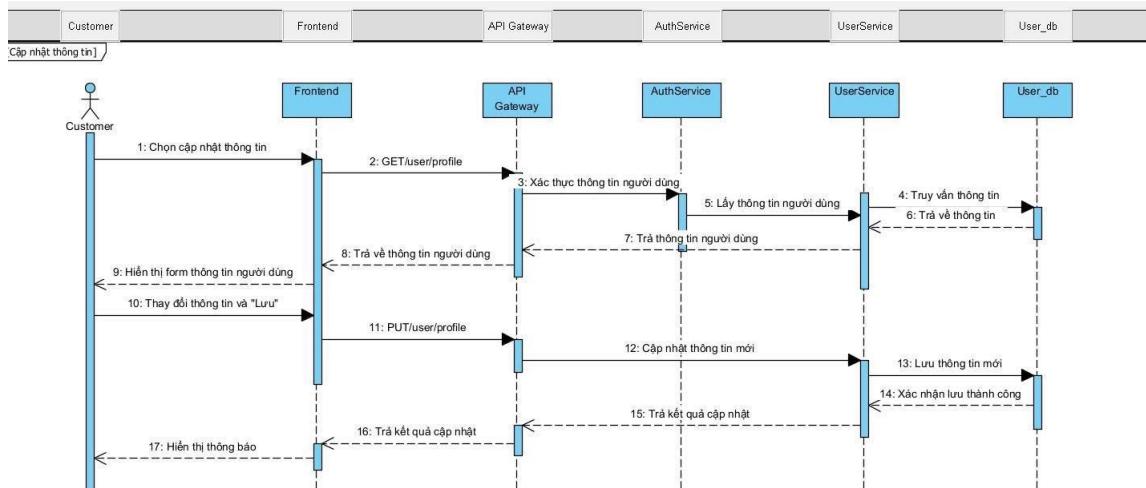
2. Đăng nhập



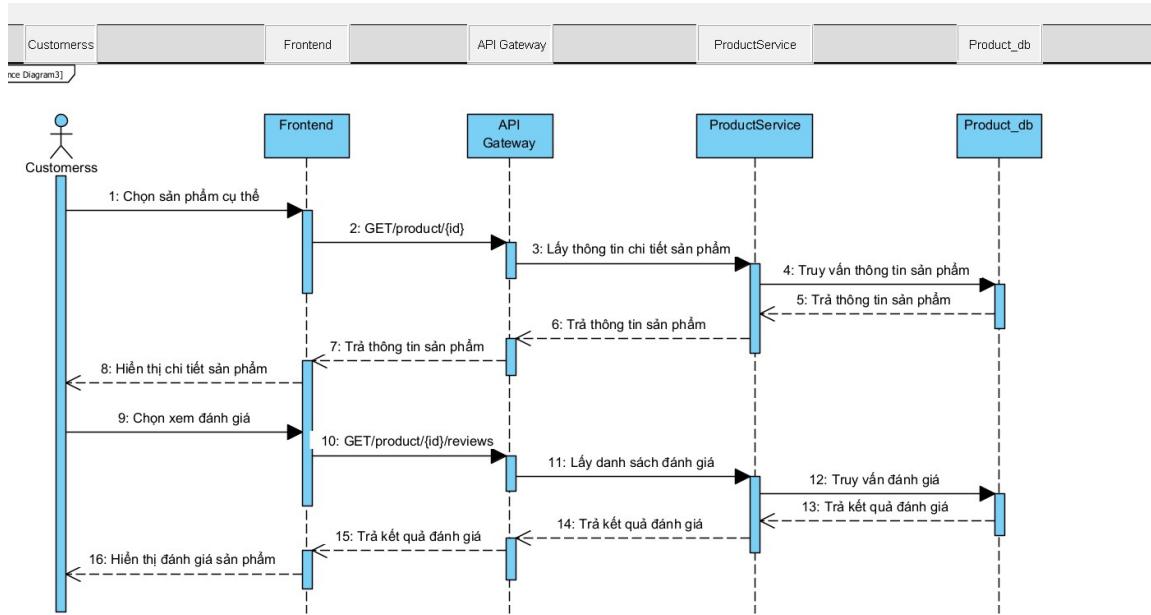
3. Quên mật khẩu



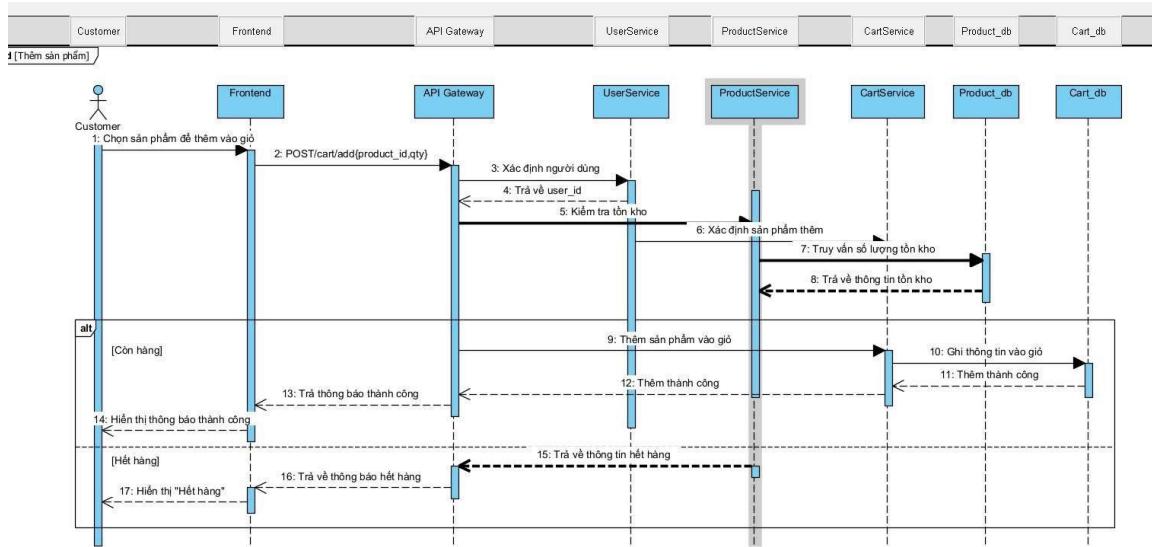
4. Cập nhật thông tin



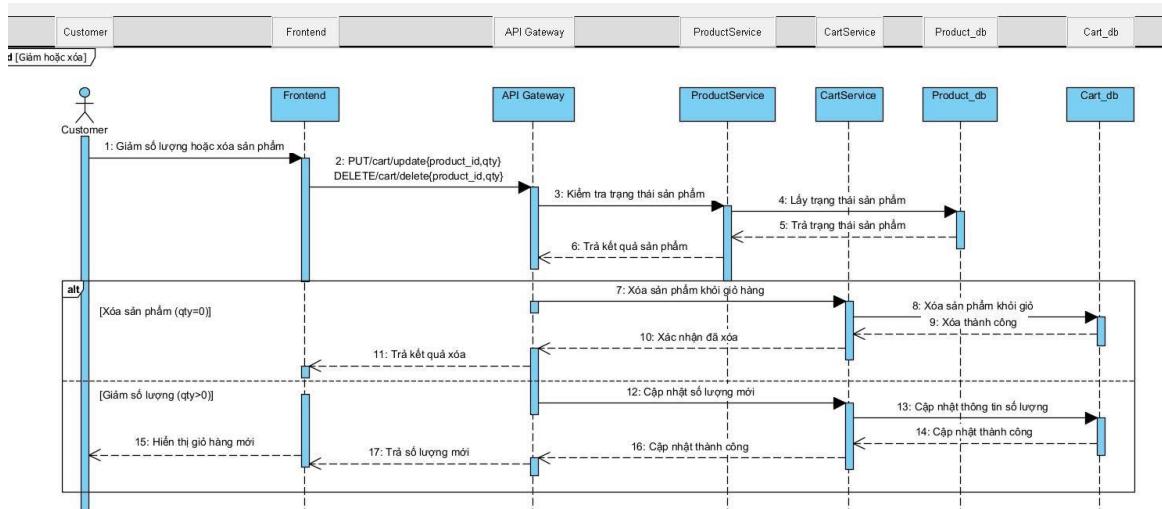
5. Xem chi tiết sản phẩm



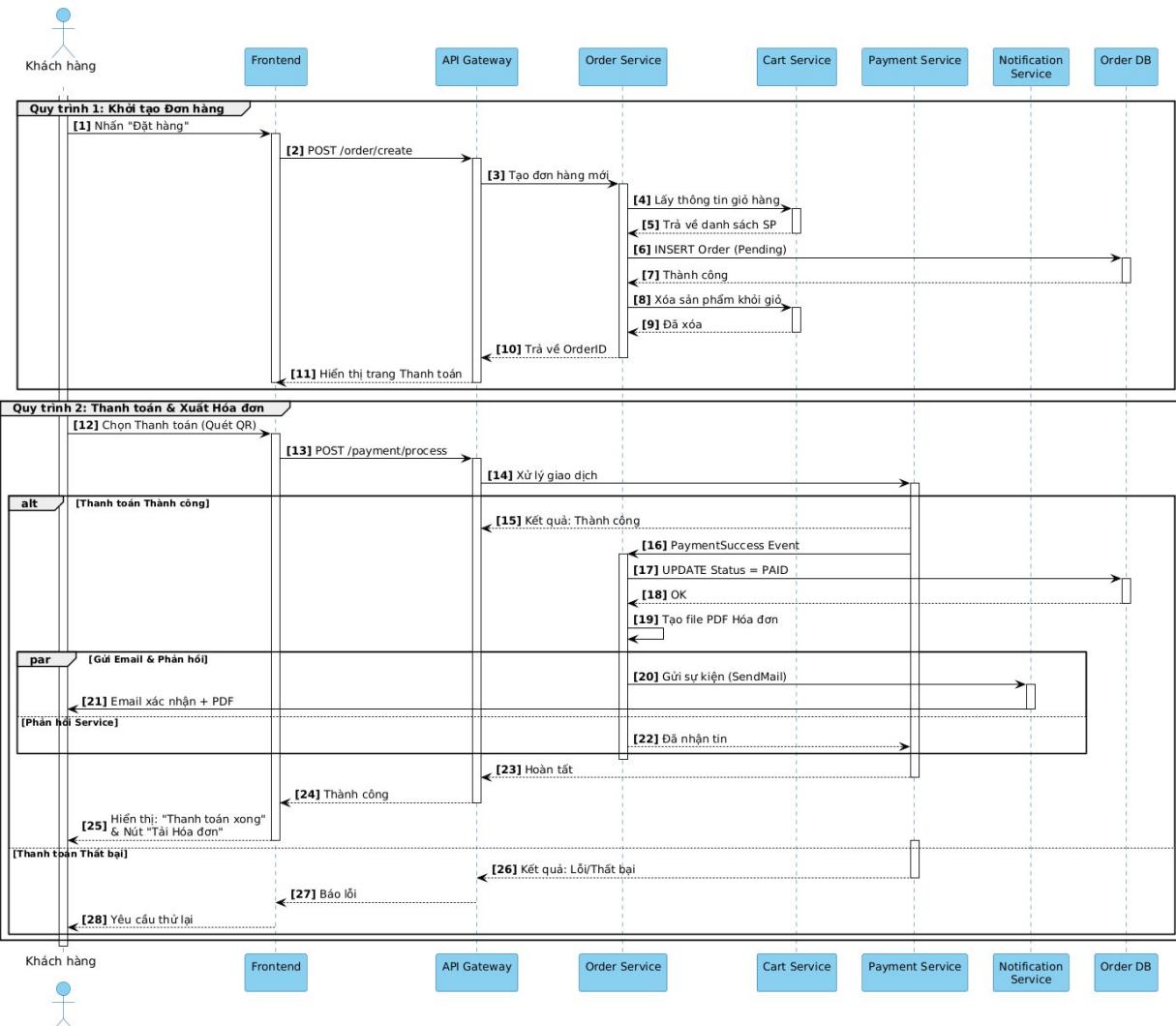
6. Thêm sản phẩm vào giỏ



7. Giảm hoặc xóa sản phẩm khỏi giỏ hàng

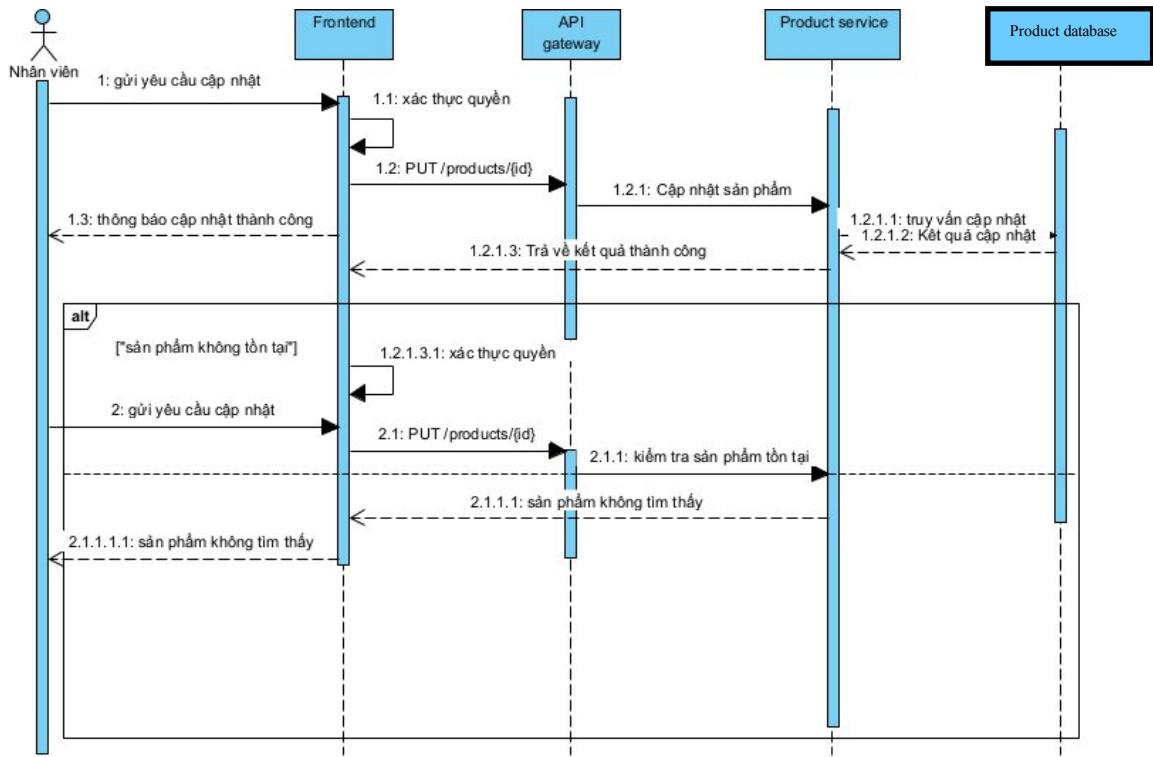


8. Đặt hàng

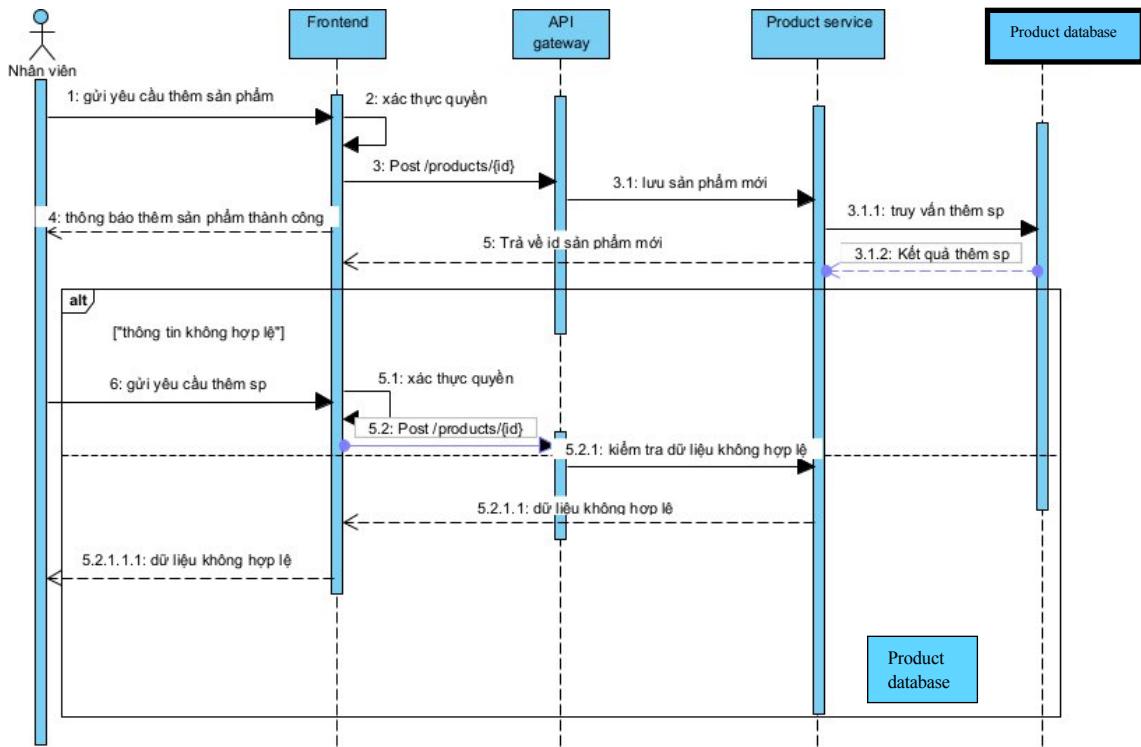


- Quản Lý Sản Phẩm

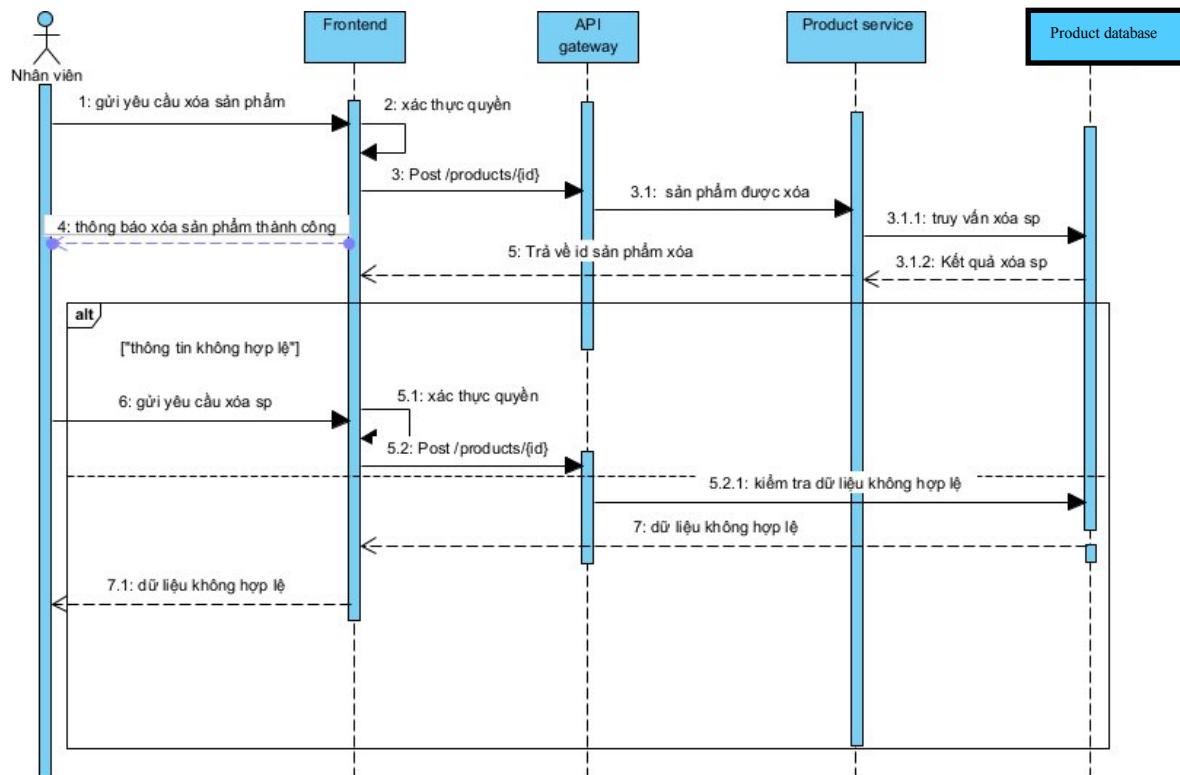
1. Cập nhật sản phẩm



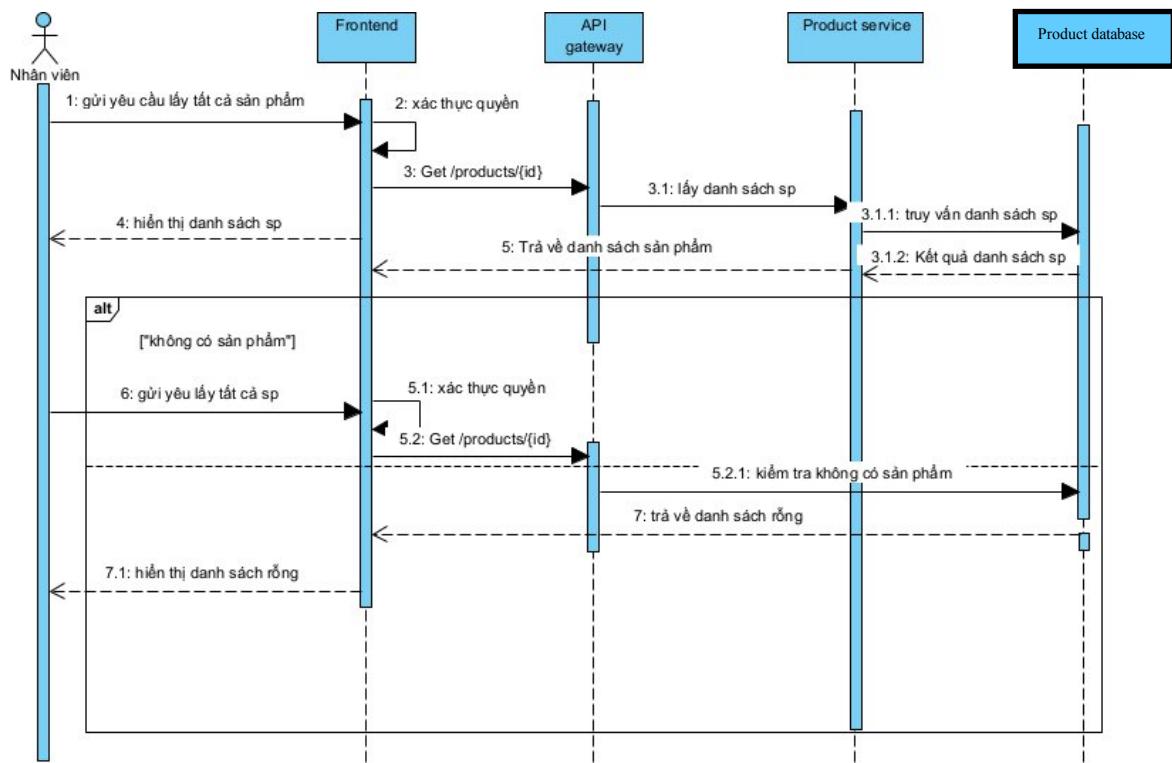
2. Thêm sản phẩm



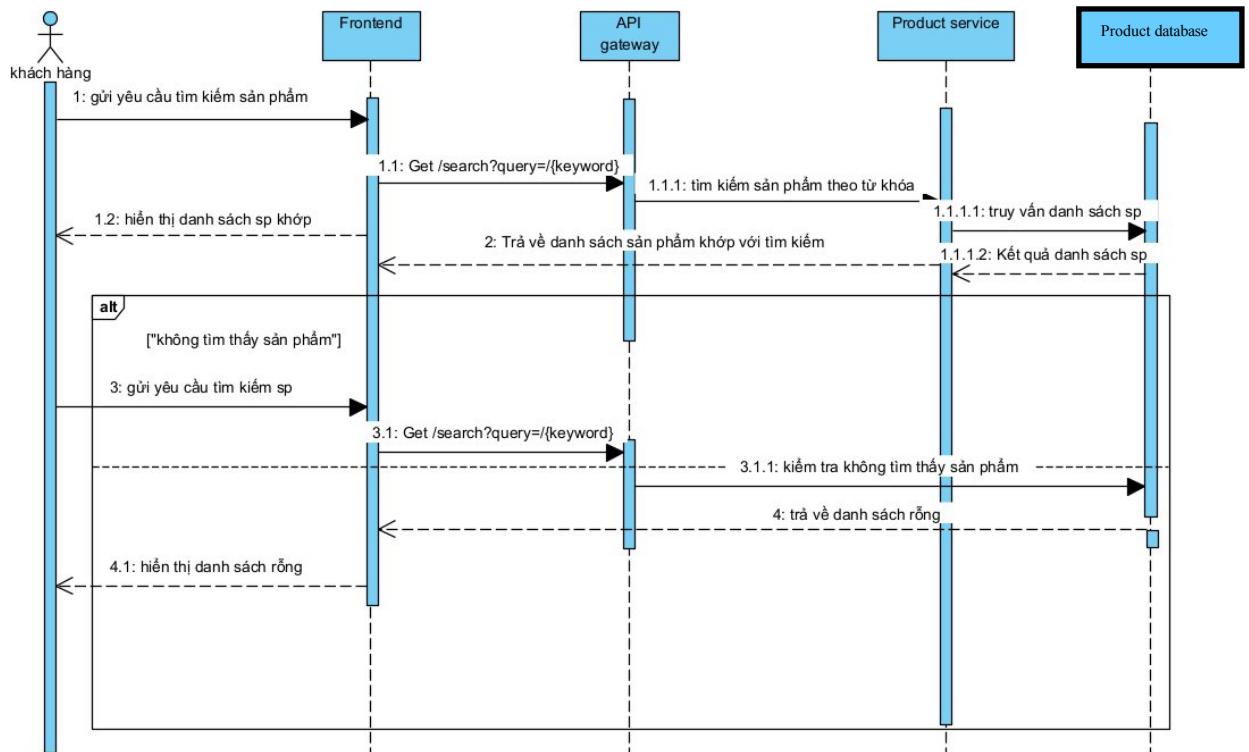
3. Xóa sản phẩm



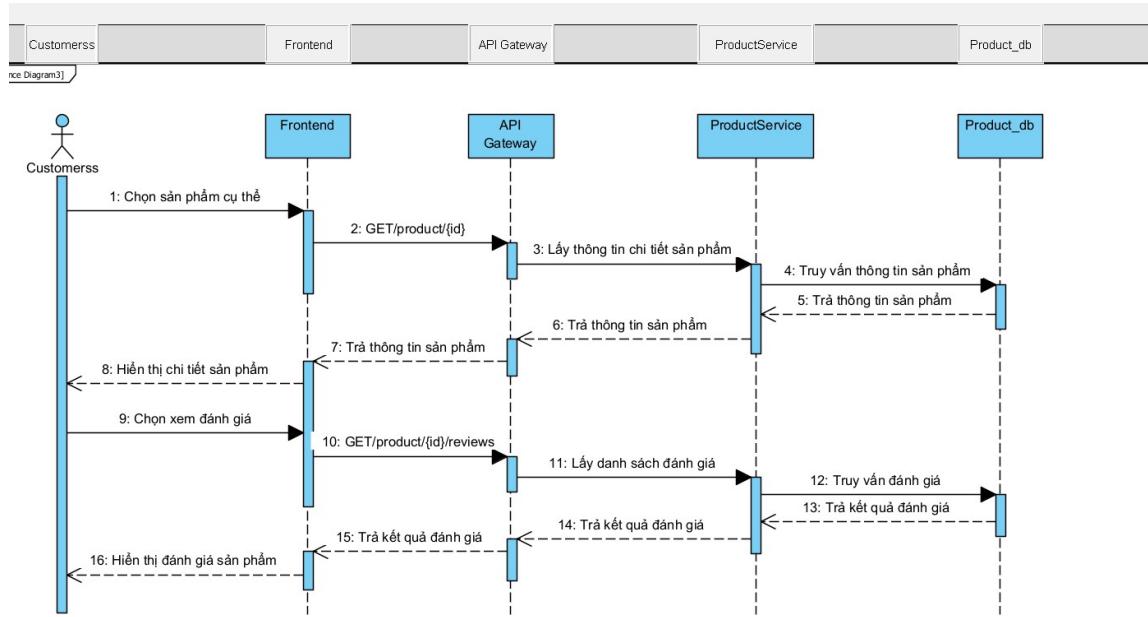
4. Lấy tất cả sản phẩm



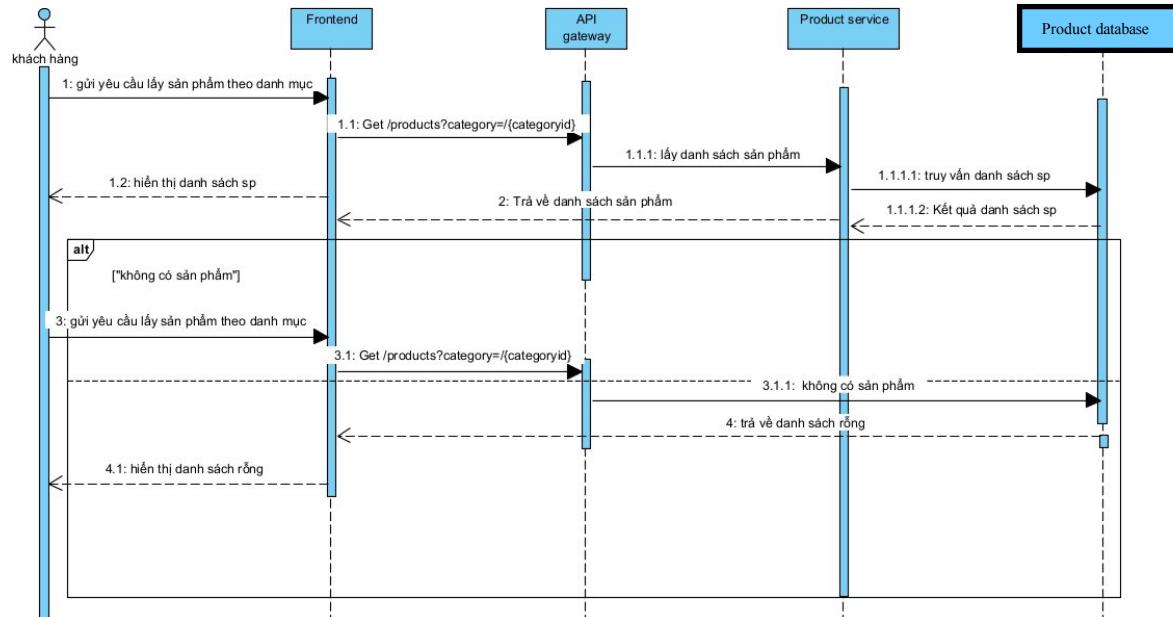
5. Tìm kiếm sản phẩm



6. Xem chi tiết sản phẩm



7. Lấy sản phẩm theo danh mục



2.6 Thiết kế cơ sở dữ liệu

2.6.1 Phần khách hàng

a) Mục tiêu sử dụng CSDL

- Hệ thống có thể:
 - Đăng ký / Đăng nhập khách hàng
 - Quản lý hồ sơ khách hàng (thông tin cá nhân, địa chỉ, liên hệ...)
 - Phân quyền theo vai trò (admin, user, staff...)
 - Ghi nhận các hoạt động như đơn hàng, bài viết, lịch sử truy cập, v.v.
- Khách hàng có thể được sử dụng trong các mối liên kết với nhiều thực thể khác nhau: Roles, Orders, Profiles, Permissions.

b) *Thông tin cần lưu trữ trong CSDL*

Trường	Kiểu dữ liệu SQL	Mô tả
UserId	Short	Mã định danh duy nhất của khách hàng
LastName	VARCHAR(50)	Họ của khách hàng
FirstName	VARCHAR(50)	Tên của khách hàng
Phone	VARCHAR(12)	Số điện thoại của khách hàng
Address	VARCHAR(256)	Địa chỉ giao hàng hoặc liên hệ của khách hàng
Email	VARCHAR(50)	Địa chỉ email (duy nhất, dùng để đăng nhập hoặc liên hệ)
Password	VARCHAR(255)	Mật khẩu đã mã hóa (dùng độ dài lớn để chứa hash như bcrypt)
Role	VARCHAR(20)	Vai trò của khách hàng (ví dụ: 'user', 'admin', 'shipper')
CreatedAt	DATETIME	Thời điểm tạo tài khoản
UpdatedAt	DATETIME	Thời điểm cập nhật gần nhất

c) *Tóm tắt yêu cầu nghiệp vụ*

Quản lý khách hàng: tạo, đọc, cập nhật, xóa (CRUD)

Mỗi khách hàng có thông tin đăng nhập, phân quyền, xác thực

Hệ thống dễ dàng mở rộng sang đơn hàng, thông báo, profile nâng cao...

Email không được trùng, UserID là duy nhất

Tối ưu hóa truy vấn qua chỉ mục (index) cho Email, UserID

2.6.2 Phần nhân viên

a) *Mục tiêu sử dụng CSDL*

Hệ thống có thể:

- Tạo mới nhân viên
- Cập nhật thông tin nhân viên
- Xoá nhân viên

- Tìm kiếm / tra cứu danh sách nhân viên
- Phân tích, báo cáo theo thông tin nhân viên như ngày tuyển dụng, số lượng theo địa chỉ.

Hệ thống này có thể là một microservice độc lập, phục vụ cho hệ thống lớn hơn (quản lý công ty, hệ thống nhân sự...).

b) Thông tin cần lưu trữ trong CSDL

Trường	Kiểu dữ liệu	Mô tả
EmployeeId	Short	Mã định danh duy nhất của nhân viên
LastName	String(50)	Họ của nhân viên
FirstName	String(50)	Tên của nhân viên
Phone	String(12)	Số điện thoại
Address	String(256)	Địa chỉ nơi ở hoặc nơi làm việc
Email	String(50)	Email liên hệ
HiredDate	String(30)	Ngày tuyển dụng

c) Tóm tắt yêu cầu nghiệp vụ

Cho phép thêm nhân viên mới vào hệ thống

Cập nhật nhân viên: Cập nhật thông tin như số điện thoại, địa chỉ, email

Xoá nhân viên: Xoá theo ID hoặc theo điều kiện

Tìm theo tên, họ, địa chỉ hoặc ngày tuyển

Lấy toàn bộ danh sách nhân viên hoặc phân trang

Thống kê số lượng nhân viên theo tháng/năm

2.6.3 Phần sản phẩm

a) Mục tiêu sử dụng CSDL

Hệ thống CSDL sẽ được sử dụng trong một ứng dụng để:

- Quản lý danh mục sản phẩm (Category)
- Quản lý sản phẩm (Product)
- Cung cấp API trả về dữ liệu phân trang (PaginationResponse) và phản hồi chuẩn hóa (responseObject)
- Hệ thống có thể phục vụ cho ứng dụng thương mại điện tử, hoặc hệ thống quản lý kho, hoặc backend quản lý sản phẩm của doanh nghiệp.

Các thao tác nghiệp vụ chính gồm:

- CRUD sản phẩm, danh mục
- Tìm kiếm, lọc, phân trang
- Quản lý nhân viên thao tác trên sản phẩm

b) Thông tin cần lưu trữ trong CSDL

❖ Bảng Product

Tên cột	Kiểu dữ liệu	Ràng buộc	Mô tả
---------	--------------	-----------	-------

id	BIGINT	PRIMARYKEY, AUTO_INCREMENT	Mã định danh duy nhất của sản phẩm (khóa chính)
name	VARCHAR(255)	NOT NULL	Tên sản phẩm
price	FLOAT hoặc DECIMAL(10,2)	NOT NULL	Giá sản phẩm
quantity	INT	DEFAULT 0	Số lượng còn trong kho
image	VARCHAR(500)	NULL	Đường dẫn hoặc URL ảnh sản phẩm
category_id	BIGINT	FOREIGN KEY → Category(id)	Mã danh mục sản phẩm thuộc về

❖ Bảng Category

Tên cột	Kiểu dữ liệu	Ràng buộc	Mô tả
id	BIGINT	PRIMARYKEY, AUTO_INCREMENT	Mã định danh duy nhất của danh mục (khóa chính)
name	VARCHAR(255)	NOT NULL, UNIQUE	Tên danh mục (ví dụ: Điện thoại, Thời trang)

(Product ← (ManyToOne) → Category)

- ❖ Các lớp hỗ trợ cho việc trả dữ liệu qua API (RESTful):
 - **PaginationResponse<T>** giúp trả danh sách có phân trang.
 - **responseObject<T>** giúp đóng gói kết quả API có status, message, data

c) *Tóm tắt yêu cầu nghiệp vụ*

- ❖ Quản lý sản phẩm (Product)

Thêm sản phẩm mới vào hệ thống với đầy đủ thông tin (tên, giá, số lượng, ảnh, danh mục).

Sửa thông tin sản phẩm đã có: tên, giá, ảnh, số lượng, danh mục.

Xoá một sản phẩm dựa trên id khi không còn kinh doanh hoặc nhập sai.

Xem đầy đủ thông tin một sản phẩm theo id.

Lấy danh sách toàn bộ sản phẩm hoặc phân trang (có hỗ trợ tìm kiếm/lọc).

Tìm sản phẩm theo tên, danh mục, giá...

Lọc các sản phẩm thuộc cùng một danh mục cụ thể.

- ❖ Quản lý danh mục (Category)

Tạo mới danh mục sản phẩm như: Điện thoại, Quần áo, Sách...

Chỉnh sửa tên danh mục khi cần.

Xoá danh mục khi không còn sử dụng (cần kiểm tra không còn sản phẩm liên quan).

Hiển thị danh sách tất cả danh mục hiện có.

Xem chi tiết một danh mục cụ thể.

2.7 Thiết kế mô hình hướng dịch vụ

Đề tài “Xây dựng hệ thống đặt đồ ăn online” được xây dựng theo mô hình kiến trúc microservice, với mục tiêu đảm bảo tính linh hoạt, mở rộng và khả năng bảo trì tốt trong quá trình phát triển và vận hành thực tế.

2.7.1 Định hướng thiết kế

Thay vì phát triển theo hướng nguyên khối (monolithic), hệ thống được chia thành nhiều dịch vụ nhỏ (microservice), mỗi dịch vụ đảm nhiệm một nghiệp vụ riêng biệt. Các dịch vụ hoạt động độc lập, có thể triển khai, bảo trì và mở rộng riêng mà không ảnh hưởng đến toàn hệ thống. Điều này không chỉ giúp giảm thiểu rủi ro khi có lỗi xảy ra mà còn giúp nhóm phát triển dễ dàng phân chia công việc và triển khai song song.

2.7.2. Các dịch vụ chính trong hệ thống

Hệ thống gồm các dịch vụ sau:

- **API Gateway:** Là điểm vào duy nhất của toàn bộ hệ thống. Có nhiệm vụ định tuyến các yêu cầu từ phía giao diện người dùng đến các dịch vụ tương ứng. Ngoài ra, Gateway cũng đảm nhiệm chức năng xác thực token JWT và bảo vệ các route nội bộ.
- **Authentication Service:** Chịu trách nhiệm xác thực người dùng, xử lý đăng nhập, đăng ký, và sinh mã thông báo (JWT). Đây là dịch vụ trọng yếu cho bảo mật toàn hệ thống.
- **User Service:** Quản lý thông tin người dùng, cho phép người dùng cập nhật hồ sơ cá nhân, truy xuất thông tin khi cần thiết.
- **Product Service:** Quản lý thông tin sản phẩm (món ăn), bao gồm thêm, sửa, xóa và hiển thị danh sách sản phẩm cho người dùng.
- **Cart Service:** Lưu trữ và xử lý các thao tác với giỏ hàng như thêm sản phẩm, cập nhật số lượng, xóa sản phẩm khỏi giỏ.
- **Order Service:** Xử lý việc đặt hàng, theo dõi trạng thái đơn hàng, và lưu trữ lịch sử giao dịch của người dùng.

- **Employee Service:** Dành cho vai trò quản trị viên (admin) trong hệ thống. Dịch vụ này cho phép quản lý danh sách nhân viên, phân quyền và kiểm soát quyền truy cập nội bộ.
- **Discovery Server (Eureka):** Là dịch vụ trung tâm giúp các microservice khác đăng ký và tự động phát hiện lẫn nhau, thay thế việc cấu hình địa chỉ thủ công, hỗ trợ hệ thống vận hành linh hoạt hơn.

2.7.3 Giao tiếp giữa các dịch vụ

- **Giao tiếp đồng bộ:** Các dịch vụ trong hệ thống sử dụng giao thức HTTP RESTful để gọi nhau thông qua các API được công bố. Tất cả các yêu cầu đi từ client đều được xử lý qua API Gateway.
- **Service Discovery (Eureka):** Thay vì gọi nhau qua IP tĩnh, các dịch vụ sẽ đăng ký với discovery-server và định danh nhau bằng tên dịch vụ. Việc này giúp hệ thống dễ mở rộng, triển khai nhiều instance.
- **Bảo mật giao tiếp:** Cơ chế bảo mật được xây dựng dựa trên Spring Security kết hợp với JWT. API Gateway đóng vai trò kiểm tra token trước khi chuyển tiếp yêu cầu đến các dịch vụ nội bộ.

2.7.4. Mối liên hệ giữa giao diện và các dịch vụ

Giao diện người dùng	Dịch vụ tương ứng
Đăng ký, đăng nhập	authentication-service
Hồ sơ cá nhân, thông tin người dùng	user-service
Danh sách sản phẩm, tìm kiếm món ăn	product-service
Giỏ hàng, thao tác mua hàng	cart-service
Đặt đơn hàng, lịch sử đơn hàng	order-service
Quản lý nhân viên (admin)	employee-service
Thanh toán	payment-service
Quản lý hóa đơn	invoice-service

2.7.5. Ưu điểm của mô hình đã triển khai

- **Tăng khả năng mở rộng:** Có thể triển khai độc lập từng service trên nhiều server khác nhau hoặc theo nhu cầu thực tế.
- **Dễ bảo trì:** Khi có lỗi xảy ra, chỉ cần kiểm tra và khắc phục ở service đó mà không ảnh hưởng đến toàn hệ thống.
- **Đảm bảo tính bảo mật và quản lý truy cập tập trung:** Nhờ cơ chế token-based authentication qua Gateway.
- **Tăng tính linh hoạt trong phát triển:** Các thành viên trong nhóm có thể làm việc đồng thời trên các service khác nhau mà không bị phụ thuộc lẫn nhau.

2.8 Giao diện mẫu

2.8.1 Giao diện phía khách hàng

- Trang chủ và Danh sách món ăn

The screenshot shows the homepage of the Tasty Treat website. At the top, there is a navigation bar with links for Home, Foods, Cart, Contact, Login, and Register. A shopping cart icon with a '0' is also present. Below the navigation bar, there is a promotional banner with the text "Easy order & fast delivery" and "Enjoy your favorite Pizza". A red button labeled "Menu >" is located below the banner. To the right, there is an illustration of a person wearing a red helmet and riding a red scooter, holding a pizza box. Below the banner, there is a search bar with placeholder text "I'm looking for..." and a dropdown menu for "Select Sorting Order". The main content area displays a grid of eight product cards, each showing a small image of a pizza, the product name (product1 through product8), the location (Kansas City, KS), the price (6 and 110 €), and a red "Add to Cart" button.

Product	Location	Price	Action
product1	Kansas City, KS	6 110 €	Add to Cart
product2	Kansas City, KS	6 110 €	Add to Cart
product3	Kansas City, KS	6 110 €	Add to Cart
product4	Kansas City, KS	6 110 €	Add to Cart
product5			
product6			
product7			
product8			

- Giao diện đăng ký



Home Foods Cart Contact Login Register

0

First Name

Last Name

Username

Email

Password

Phone

Address

- Giao diện đăng nhập



Home Foods Cart Contact Login Register

0



Username/Email

Password



Tasty Treat

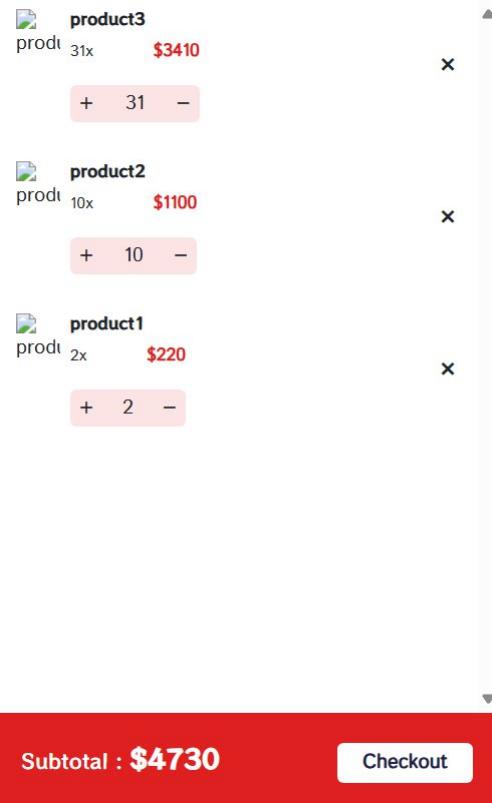
Best Pizzas in town, try it out!

Delivery Time

Friday - Tuesday 10:00am - 11:00pm

Wednesday - Thursday Off day

- Giao diện Cart



- Giao diện Order

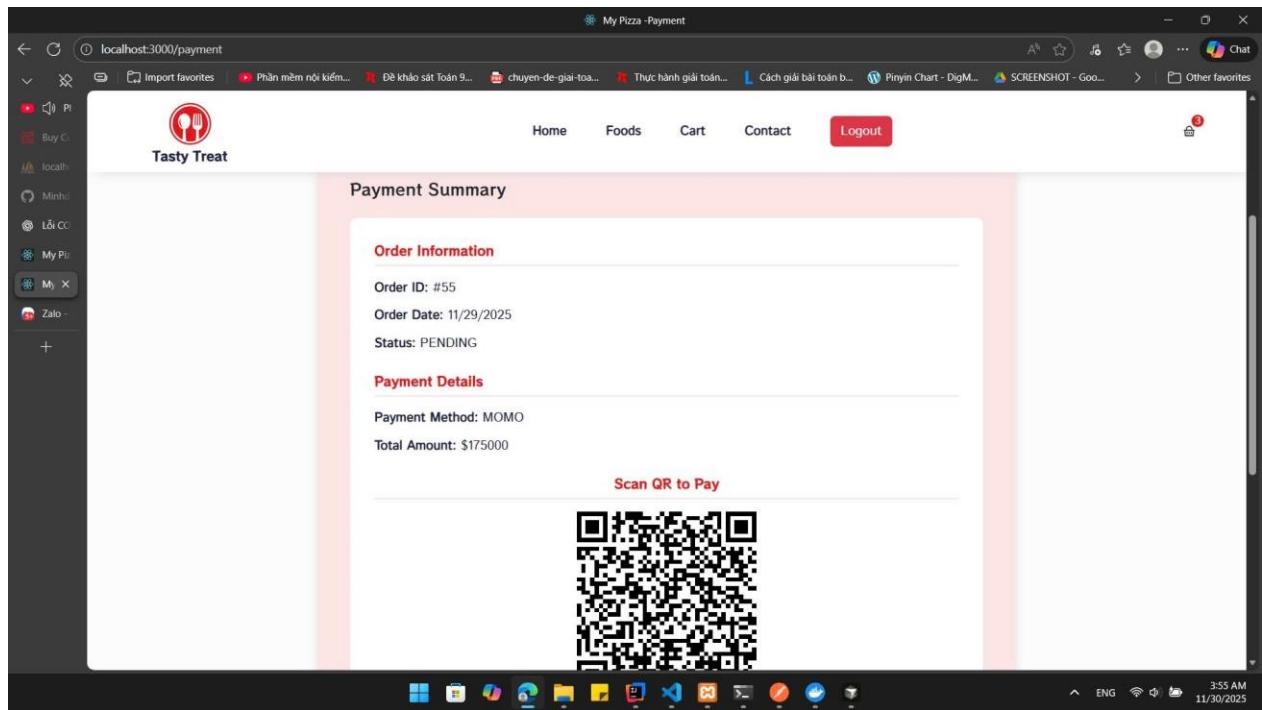
The screenshot shows a web browser displaying a checkout page for "Tasty Treat". The page includes the following sections:

- Order Summary** table:

Product	Price	Quantity	Total
Gỏi cuốn tôm thịt	\$45000	1	\$45000
Salad cá ngừ	\$60000	1	\$60000
Phở bò đặc biệt	\$70000	1	\$70000
- Payment Details** section:

Subtotal:	\$175000
Shipping:	\$0
Tax:	\$0
Total:	\$175000
- Payment Method** section:
 - VNPAY
 - MOMO** (selected)

- Giao diện Payment



- Giao diện xuất hóa đơn

My Pizza -Invoice Management

Tasty Treat

Invoice Management

Xem trước hóa đơn #6

Pizza Store
123 Food Street, District 1, Ho Chi Minh City
Tel: (028) 1234 5678 | Email: info@pizzastore.com

HÓA ĐƠN BÁN HÀNG

Mã hóa đơn:	#6	Mã đơn hàng:	#54	Thông tin khách hàng:
Ngày tạo:	20/10/13 29/11/2025	Mã KH:	#3	
Trạng thái:	CREATED	Tên KH:	N/A	
		Phương thức TT:	MOMO	

Chi tiết đơn hàng:

#	Sản phẩm	SL	Đơn giá	Thành tiền
5				

In hóa đơn | Tải PDF | Đóng

Delivery Time

2.8.2 Giao diện phía nhân viên

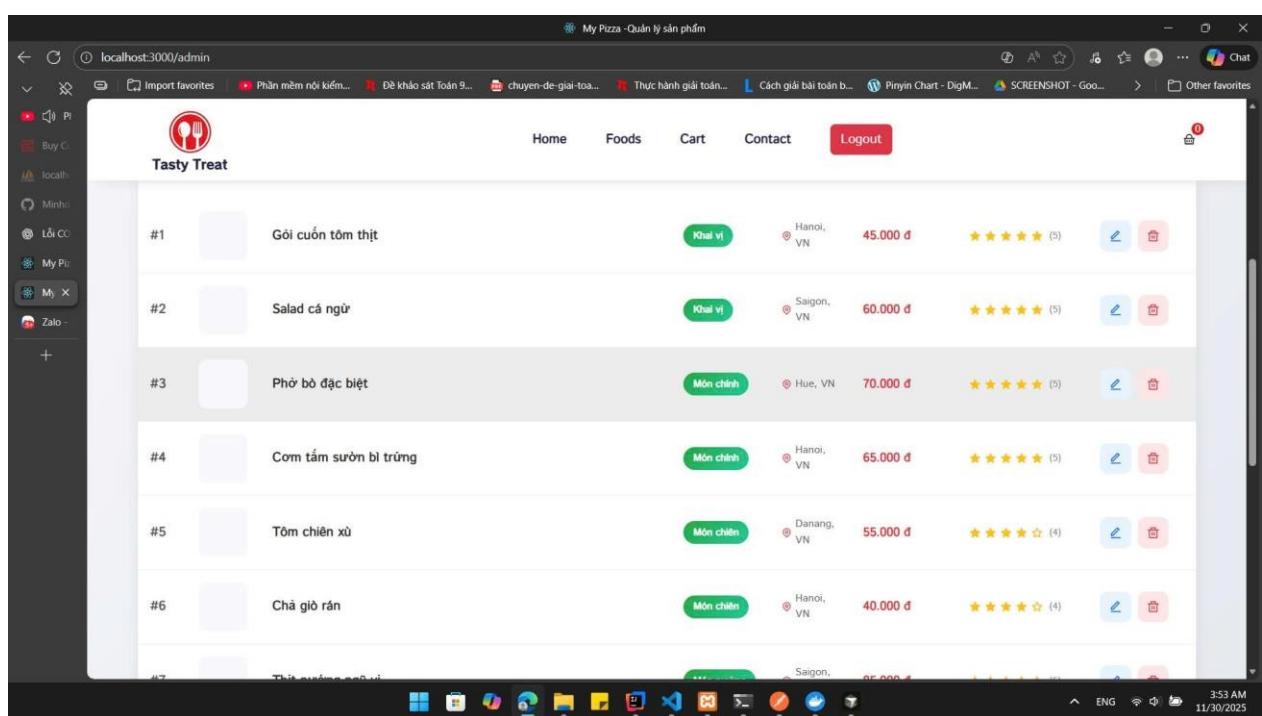
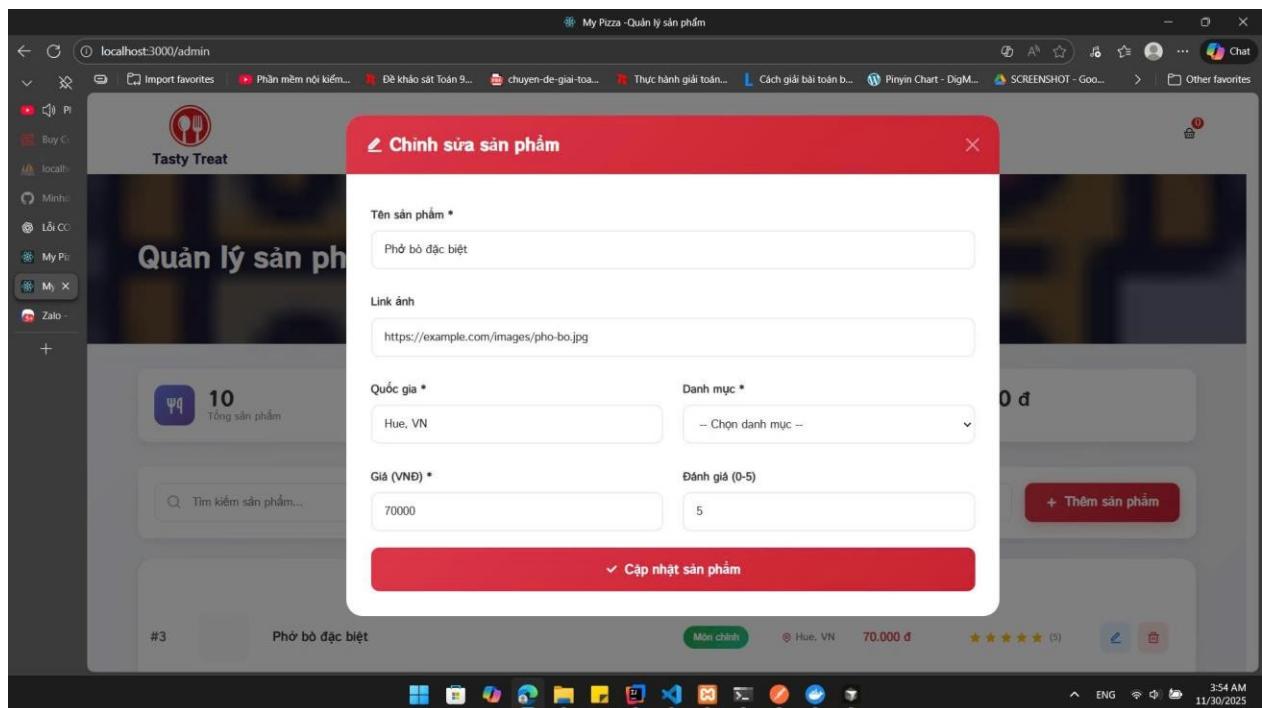
- Giao diện đăng nhập

The screenshot shows the login interface for the Tasty Treat website. At the top, there is a navigation bar with links for Home, Foods, Cart, Contact, Login (which is highlighted in red), and Register. To the right of the navigation bar is a small notification icon with a red '0'. Below the navigation bar is a large input form for logging in. It features a placeholder for a user profile picture, followed by two input fields labeled 'Username/Email' and 'Password', and a blue 'Login' button at the bottom.

The screenshot shows the homepage of the Tasty Treat website. It includes the logo, the restaurant name 'Tasty Treat', a brief description 'Best Pizzas in town, try it out!', and delivery time information: 'Friday - Tuesday 10:00am - 11:00pm' and 'Wednesday - Thursday Off day'.

- Giao diện quản lý sản phẩm

The screenshot shows the product management interface on a local server at localhost:3000/admin. The main page displays a summary with a total of 10 products. A modal window titled 'Thêm sản phẩm mới' (Add new product) is open, prompting the user to enter product details such as name, link ảnh (image link), Quốc gia (Country), Danh mục (Category), Giá (VNĐ) (Price), and Đánh giá (0-5) (Rating). A large red 'Thêm sản phẩm' (Add product) button is at the bottom of the modal. The background shows a blurred view of the product list and search bar.



- Giao diện xem hóa đơn

The screenshot shows the 'Invoice Management' section of the application. At the top, there is a search bar labeled 'Search by Invoice ID / Order ID...' and a 'Search' button. Below the search bar is a table with the following data:

Invoice ID	Order ID	Status	Total Amount	Date	Actions
6	54	CREATED	520 đ	29/11/2025	
5	46	CREATED	520.000 đ	12/11/2025	

2.8.3. Giao diện phía quản trị viên

- Giao diện xem báo cáo thống kê

The screenshot shows the 'Quản lý sản phẩm' (Product Management) section. At the top, there are three summary boxes: 'Tổng sản phẩm' (10), 'Danh mục' (5), and 'Tổng giá trị' (865.000 đ). Below these are buttons for searching, filtering, and adding products. A detailed product view for 'Gỏi cuốn tôm thịt' is shown, including its price (45.000 đ), location (Hanoi, VN), rating (5 stars), and edit/delete buttons.

- Giao diện quản lý nhân viên

<input type="checkbox"/>	ID	First Name	Last Name	Email	Phone	Address	Hired Date	Delete
<input type="checkbox"/>	1	Nguyễn	An	an.nguyen@example.com	0909123456	123 Đường Lê Lợi, Q1	2022-01-15	
<input type="checkbox"/>	2	Trần	Bình	binh.tran@example.com	0909234567	456 Nguyễn Huệ, Q3	2023-04-10	
<input type="checkbox"/>	3	Lê	Cường	cuong.le@example.com	0909345678	789 Hai Bà Trưng, Q5	2024-03-20	

Rows per page: 100 < > 1-3 of 3

CHƯƠNG 3. CÔNG CỤ, CÔNG NGHỆ VÀ TRIỂN KHAI HỆ THỐNG

3.1 Công cụ và công nghệ sử dụng

Để hiện thực hóa kiến trúc Microservices và đảm bảo hiệu năng cho hệ thống bán đồ ăn nhanh, nhóm đã lựa chọn bộ công nghệ (Tech Stack) phổ biến và mạnh mẽ nhất hiện nay.

3.1.1 Ngôn ngữ lập trình và Framework (Backend)

Hệ thống phía Server được xây dựng hoàn toàn bằng ngôn ngữ **Java**, sử dụng hệ sinh thái **Spring**.

- **Java Development Kit (JDK) 17:** Phiên bản hỗ trợ dài hạn (LTS), mang lại hiệu suất cao và nhiều tính năng mới cho lập trình hướng đối tượng.
- **Spring Boot 3.2.5:** Framework cốt lõi giúp khởi tạo các Microservice nhanh chóng, giảm thiểu cấu hình XML phức tạp, tích hợp sẵn máy chủ Tomcat.
- **Spring Cloud:** Bộ công cụ không thể thiếu để vận hành kiến trúc phân tán:
 - *Spring Cloud Netflix Eureka:* Đóng vai trò Service Discovery (Máy chủ định danh), giúp các dịch vụ (Order, Product, Payment...) tự động tìm thấy nhau.
 - *Spring Cloud Gateway:* API Gateway đóng vai trò cửa ngõ duy nhất, định tuyến request từ Frontend đến đúng Service xử lý.
 - *Spring Cloud Config:* Quản lý toàn bộ file cấu hình (`application.yml`) của các dịch vụ tại một nơi tập trung (Git).
 - *OpenFeign:* Thư viện giúp việc gọi API giữa các service (ví dụ: Order gọi Payment) trở nên đơn giản như gọi hàm Java thông thường.

3.1.2 Giao diện người dùng (Frontend)

Phía Client được xây dựng theo mô hình **Single Page Application (SPA)** để tối ưu trải nghiệm người dùng.

- **ReactJS:** Thư viện JavaScript của Facebook, giúp xây dựng giao diện dựa trên các Component tái sử dụng (Header, Footer, ProductCard...).
- **Axios:** Thư viện HTTP Client dùng để gọi API từ React sang Spring Boot.
- **Ant Design / Bootstrap:** Framework CSS giúp thiết kế giao diện nhanh, đẹp và tương thích (Responsive) với nhiều kích thước màn hình.

3.1.3 Hệ quản trị Cơ sở dữ liệu (Database)

MySQL: Hệ quản trị CSDL quan hệ (RDBMS) mã nguồn mở.

- Áp dụng mô hình **Database per Service**: Mỗi dịch vụ (Customer, Product, Order) sẽ kết nối đến một Schema (Database) riêng biệt trong MySQL để đảm bảo tính độc lập dữ liệu (Loose Coupling).
- Sử dụng **Spring Data JPA (Hibernate)** để thao tác với CSDL thông qua các Java Entity mà không cần viết quá nhiều câu lệnh SQL thủ công.

3.1.4. Các công cụ hỗ trợ phát triển (DevTools)

- Docker: Đóng gói từng Service và Database vào các Container riêng biệt, giúp việc triển khai đồng bộ trên mọi môi trường máy tính.
- Postman: Công cụ dùng để test từng API riêng lẻ (Login, Create Order...) trước khi ghép nối với giao diện.
- Git & GitHub: Quản lý mã nguồn và phối hợp làm việc nhóm.
- Maven: Công cụ quản lý thư viện và build dự án Java.
- 3.2. Triển khai và Cài đặt
- 3.2.1. Yêu cầu môi trường
- Để chạy được hệ thống demo bằng Docker, máy tính cần cài đặt:
 - Docker Desktop (khuyến nghị)
 - Docker Compose (tích hợp sẵn trong Docker Desktop)
 - (Tùy chọn) Java JDK 17+ nếu muốn chạy service thủ công ngoài Docker
 - (Tùy chọn) Node.js 16+ & NPM nếu muốn chạy frontend để phát triển
- Lưu ý: Khi sử dụng Docker, bạn không cần cài Java, MySQL hay các service phụ trợ trên máy.

3.2 Các bước cài đặt hệ thống

Bước 1: Khởi chạy Backend (Các Microservices) bằng Docker

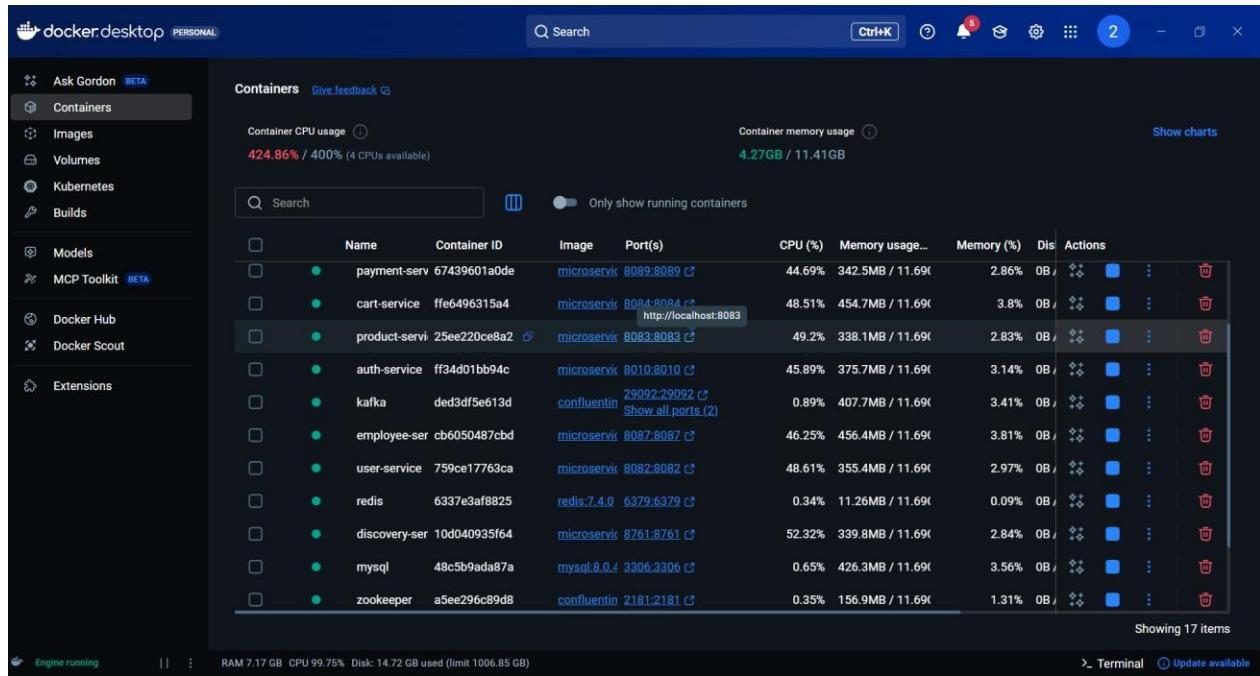
Hệ thống backend được container hóa hoàn toàn.

Chỉ cần chạy một lệnh duy nhất để lên tất cả service:

Docker compose -f docker-compose.yml up -d --build

Các service sẽ lần lượt khởi động và chạy theo thứ tự;

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. On the left sidebar, there are links for Ask Gordon (BETA), Containers (selected), Images, Volumes, Kubernetes, Builds, Models, MCP Toolkit (BETA), Docker Hub, Docker Scout, and Extensions. The main pane displays a table of running containers. The table columns are: Name, Container ID, Image, Port(s), CPU (%), Memory usage..., Memory (%), Dis, Actions. There are 17 items listed, each with a green dot icon indicating it is running. The first item is 'microservice-jav...'. The table shows resource usage for each container, such as CPU usage (e.g., 428.01%, 43.54%), memory usage (e.g., 3.2GB / 151.97GB, 257MB / 11.69GB), and memory percentage (e.g., 27.38%, 2.15%). The 'Actions' column contains icons for restarting, stopping, and deleting each container. At the bottom of the table, it says 'Showing 17 items'. The status bar at the bottom of the window shows 'Engine running', 'RAM 6.13 GB CPU 100.00% Disk: 14.72 GB used (limit 1006.85 GB)', and '2' in the top right corner.



Bước 2: Khởi chạy Frontend

Mở Terminal tại thư mục frontend-react.

Chạy lệnh: npm install (cài thư viện) -> npm start.

Hệ thống sẽ mở tại địa chỉ http://localhost:3000.

3.3. Mô tả quy trình Demo (Kịch bản kiểm thử)

Dưới đây là kịch bản demo luồng nghiệp vụ chính "**Đặt hàng - Thanh toán - Xuất hóa đơn**" mà nhóm sẽ trình bày.

3.3.1 Kịch bản 1: Khách hàng mua hàng thành công

Đăng nhập: Khách hàng truy cập web, đăng nhập bằng tài khoản customer@gmail.com.

Chọn món: Tại trang chủ, khách hàng chọn "Pizza Hải Sản" (Size L), thêm vào giỏ hàng.

Đặt hàng: Vào giỏ hàng, bấm "Tiến hành thanh toán". Hệ thống hiển thị form xác nhận địa chỉ.

Thanh toán:

Chọn phương thức "Ví điện tử (Giả lập)".

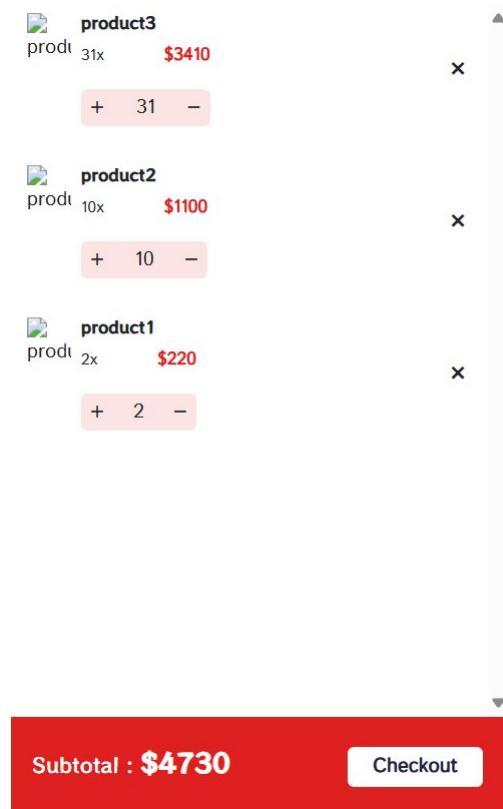
Hệ thống hiện mã QR. Khách hàng bấm "Đã thanh toán".

Kết quả:

Hệ thống thông báo "Thanh toán thành công".

Nút "**Tải hóa đơn**" xuất hiện. Khách hàng bấm tải file PDF về máy.

- **Ảnh chụp màn hình demo**



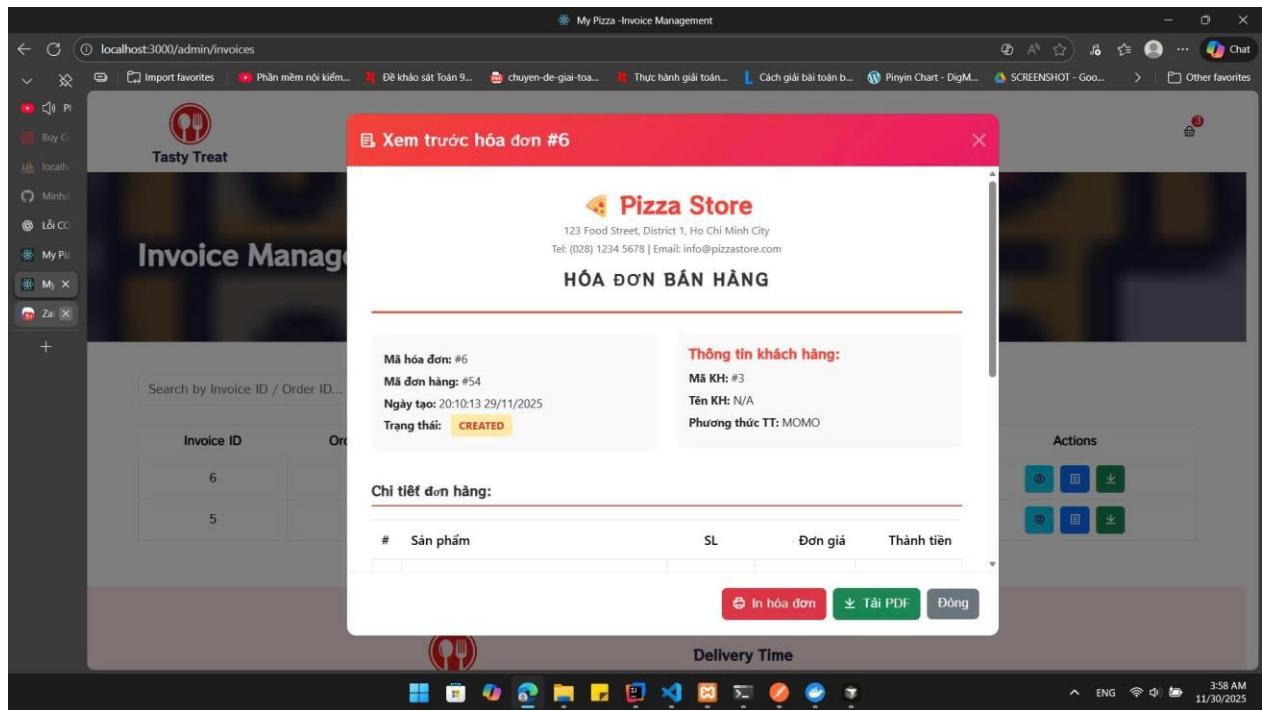
Sản phẩm trong giỏ hàng

The screenshot shows a web browser window for 'My Pizza - Checkout' at 'localhost:3000/checkout'. The page displays an 'Order Summary' table with three items: Gỏi cuốn tôm thịt (\$45000), Salad cá ngừ (\$60000), and Phở bò đặc biệt (\$70000). To the right, a 'Payment Details' section shows a subtotal of \$175000, shipping of \$0, and tax of \$0, with a total of \$175000. Below the summary is a 'Payment Method' section with options for VNPAY and MOMO, where MOMO is selected. The browser's address bar shows the URL 'localhost:3000/checkout'. The taskbar at the bottom includes icons for various Windows applications like File Explorer, Edge, and Control Panel.

Tiến hành thanh toán

The screenshot shows a web browser window for 'My Pizza - Payment' at 'localhost:3000/payment'. The page displays a 'Payment Summary' section with 'Order Information' showing Order ID #55, Order Date 11/29/2025, and Status PENDING. It also shows 'Payment Details' with a Payment Method of MOMO and a Total Amount of \$175000. Below this is a 'Scan QR to Pay' section featuring a large QR code. The browser's address bar shows the URL 'localhost:3000/payment'. The taskbar at the bottom includes icons for various Windows applications like File Explorer, Edge, and Control Panel.

Thanh toán



3.3.2 Kịch bản 2: Quản trị viên quản lý

Đăng nhập Admin: Truy cập trang /admin, đăng nhập bằng tài khoản quản lý.

Quản lý đơn: Vào menu "Đơn hàng", thấy đơn hàng vừa đặt của khách đang ở trạng thái PAID (Đã thanh toán).

Xử lý: Admin bấm chuyển trạng thái sang SHIPPED (Đang giao).

Kiểm tra bên Khách: Khách hàng F5 lại trang Lịch sử đơn hàng, thấy trạng thái đã đổi thành "Đang giao hàng".

- **Ảnh chụp màn hình demo**

The screenshot shows the Tasty Treat Admin Order Management interface. At the top, there are four summary boxes: 'Tổng đơn hàng' (10), 'Chờ xử lý' (8), 'Hoàn thành' (0), and 'Tổng doanh thu' (2,095,000đ). Below this is a search bar and a filter dropdown. A large button at the bottom right says '10 đơn hàng'. The main table lists 10 orders, each with columns for order ID, customer name, quantity, total price, date, status, and actions.

MÃ ĐH	KHÁCH HÀNG	SL SP	TỔNG TIỀN	NGÀY ĐẶT	TRẠNG THÁI	THAO TÁC
#	Kh #1	2	520.000 đ	N/A	Chờ xử lý	

This screenshot is identical to the one above, showing the same order statistics and the same table of 10 pending orders.

3.3.3. Kiểm thử hộp đen

Function Name	Đăng Ký		
Test Result	Chrome	Firefox	
Total	29	29	

Passed	27	27	
Failed	2	2	
Not Run	0	0	
NA	0	0	

ID	Summary	Steps	Expected Output	Test Results	
				Chrome	Firefox
I. Check UI/UX của màn hình "Đăng Ký"					
Dk_001	Check Giao diện mặc định của màn hình "Đăng Ký"	1. Di chuyển đến màn hình "Đăng Ký" 2. Check giao diện mặc định của màn hình "Đăng Ký"	2. UI mặc định của màn hình "Đăng Ký" được hiển thị đúng như ảnh dưới đây:	Passed	Passed
DK_02	Check bố cục,căn chỉnh các field, font,size,color ... (icon,logo,placeholder... nếu có)	1. Check bố cục,căn chỉnh các field, font,size,color .. của form "Đăng Ký"	1. - Căn chỉnh các field hợp lý, thẳng hàng - Bố cục,font,size,color hiển thị giống như trong ảnh đính kèm ở trên	Passed	Passed
DK_03	Check xử lý của hệ thống khi nháy "Tab" từ bàn phím	1. Focus chuột vào text box đầu tiên của form "Đăng Ký" 2. Nhấn phím "tab" từ bàn phím	2. Con trỏ chuột sẽ dịch chuyển qua các textbox từ trái sang phải, từ trên xuống dưới	Passed	Passed
DK_04	Check xử lý của hệ thống khi nháy	1. Focus chuột vào text box cuối cùng của form "Đăng Ký"	2. Con trỏ chuột sẽ dịch chuyển	Passed	Passed

	"Shift+ Tab' từ bàn phím	2. Nhấn phím "shift+tab" từ bán phím	qua các textbox từ phải sang trái, từ dưới lên trên		
II. Check validation của các field trên màn hình "Đăng Ký"					
2. Email					
DK_05	Để trống field "Email"	1. Để trống field "Email" 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị error message dưới field "Email" với nội dung : "Please input your email ". - Highlight và focus vào field bị lỗi. - Hiển thị thông báo trên màn hình "Please check that the form is filled out correctly".	Passed	Passed
DK_06	Check độ dài tối đa (Điều kiện : Email chưa tồn tại trong DB)	1. Nhập email độ dài = 255 ký tự vào field "Email". 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị error message dưới field "Email" với nội dung : "Please enter a valid email address!". - Highlight và focus	Passed	Passed

			vào field bị lỗi		
DK_07		1. Nhập email hợp lệ chưa tồn tại trong DB với độ dài =191 ký tự vào field "Email" 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị thông báo "Đăng ký thành công"	Passed	Passed
DK_08	Nhập sai định dạng email	1.Nhập email thiếu tên miền vào field "Email" vidu : abc@sss			
DK_09		2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị error message dưới field "Email" với nội dung : "Please enter a valid email address! "	Passed	Passed
DK_10		1. Nhập email thiếu tên miền sau dấu chấm vào field "Email"	- Highlight và focus vào field bị lỗi	Passed	Passed

		<p>"</p> <p>vidu : abc@abc.</p> <p>2. Nhập giá trị hợp lệ vào tất cả các trường còn lại.</p> <p>3. Nhấn button [Sign up]</p>			
DK_0 11		<p>1. Nhập email chứa ký tự đặc biệt (loại trừ @, _ , .) vào field "Email "</p> <p>vidu : abc\$\$\$\$^@abc.com</p> <p>2. Nhập giá trị hợp lệ vào tất cả các trường còn lại.</p> <p>3. Nhấn button [Sign up]</p>	<p>2. Hiển thị thông báo "Đăng ký thành công"</p>	Passed	Passed
DK_0 12	Nhập html java script và field "Email "	<p>1.Nhập html java script và field " Email" Ví dụ : <script> alert ('Hello') </script></p> <p>2. Nhập giá trị hợp lệ vào tất cả các trường còn lại.</p> <p>3. Nhấn button [Sign up]</p>	<p>3. Hiển thị error message dưới field "Email " với nội dung : "Please enter a valid email address! " - Highlight và focus vào field bị lỗi - Hiển thị thông báo "Please check that"</p>	Passed	Passed

			the form is filled out correctly".		
DK_0 13	Nhập Email đã tồn tại trong DB	1. Nhập "Email" đã tồn tại trong DB 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị thông báo trên màn hình với nội dung : "Email Address is already used, please change the email address ".	Passed	Passed
DK_0 14	<i>Nhập Email toàn là chữ hoa (Điều kiện : Email chưa tồn tại trong DB)</i>	1. Nhập "Email" toàn là chữ hoa 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Đăng ký thành công => Hệ thống tự động chuyển đổi và lưu email là chữ thường vào trong DB	Passed	Passed
DK_0 15	Nhập space vào trước/ sau địa chỉ Email (Điều kiện : Email chưa tồn tại trong DB)	1. Nhập space vào trước và sau email hợp lệ trên field "Email" với độ dài hợp lệ Ví dụ : " ABCD@GMAIL.COM " 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị error message dưới field "Email" với nội dung : "Please enter a valid email address! "- Highlight và focus vào field bị lỗi	Passed	Passed
3. Mật					

khẩu						
DK_0 16	Để trống field "Password"	1. Tiến hành để trống field "Password" 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Hiển thị error message dưới field "Password" với nội dung : "Please enter password " - Highlight và focus vào field bị lỗi. - Hiển thị thông báo "Please check that the form is filled out correctly"	Passed	Passed	
DK_0 17	Check hiển thị ở field "Password"	1. Nhập data vào field "Password" 2. Quan sát hiển thị data ở field đó	2. Mật khẩu vừa nhập có định dạng : ***	Passed	Passed	
DK_0 18	Check độ dài tối thiểu	1. Nhập chữ/số vào field "Password" với độ dài = 1 ký tự 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	2. Hiển thị thông báo "Đăng ký thành công"	Passed	Passed	
DK_0 19	Check độ dài tối đa	1. Trên form "Đăng Ký" tiến hành nhập chữ/số vào field với độ dài= 50 ký tự 2. Nhập giá trị	3. Màn hình hiển thị thông báo : "Đăng ký thành công"	Passed	Passed	

		hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Đăng Ký]			
DK_0 20		1.Nhập chữ/số vào field với độ dài= 51 ký tự 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Đăng Ký]	3. Hiển thị error message dưới field "Mật khẩu " với nội dung : "Mật khẩu phải có độ dài không được vượt quá 50 ký tự "	Passed	Passed
DK_0 21	Nhập chữ/số/ký tự đặc biệt (có thể bao gồm cả space, chữ hoa...) vào field "Password" với độ dài hợp lệ	1. Nhập chữ/số/space/ký tự đặc biệt vào field "Mật Khẩu" có độ dài trong khoảng từ 8 đến 50 ký tự 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Đăng Ký]	1. Mỗi ký tự/space đều được hiển thị dưới dạng mã hóa là * trên field "Password". 3. Màn hình hiển thị thông báo : "Đăng ký thành công" >> Data trường password được lưu vào trong DB như khi nhập	Passed	Passed
DK_0 22	Nhập Mật khẩu nhưng	1.Nhập Mật khẩu nhưng	3. Màn hình hiển	Passed	Passed

	không có chứa kí tự đặc biệt	không có chứa số 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	thị thông báo : "Đăng ký thành công"		
DK_0 23	Nhập Mật khẩu nhưng không có chứa số	1.Nhập Mật khẩu nhưng không có chứa số 2. Nhập giá trị hợp lệ vào tất cả các trường còn lại. 3. Nhấn button [Sign up]	3. Màn hình hiển thị thông báo : "Đăng ký thành công"	Passed	Passed
III. Function Đăng ký					
DK_0 24	Đăng ký thành công" khi click chuột vào button [Sign up]	1. Nhập valid data vào tất cả các field. 2. Click chuột và button [Sign up]	2. Hiển thị pop-up thông báo : "Đăng ký thành công" 3. Hệ thống sẽ tự động chuyển tới trang chính của hệ thống	Passed	Passed
DK_0 25	Đăng ký thành công" khi nhấn Enter từ bàn phím	1. Tại form "Đăng ký" , tiến hành nhập valid data vào tất cả các field. 2. Nhấn Enter từ bàn phím	2. Hiển thị pop-up thông báo : "Đăng ký thành công" 3. Hệ	Passed	Passed

			thông sẽ tự động chuyển tới trang chính của hệ thống		
DK_0 26	Đăng ký không thành công trong trường hợp đang nhấn button [Sign up] thì bị mất kết nối với server	1.Nhập valid data vào tất cả các field. 2. Click chuột và button [Sign up] nhưng bị mất kết nối với server	2. Quá trình Đăng ký không thành công, chỉ hiển thị pop-up "loading".	Failed	Failed
DK_0 27	Đăng ký không thành công khi bỏ trống tất cả các field bắt buộc phải nhập	1. Bỏ trống tất cả các field bắt buộc phải nhập 2. Click chuột và button [Sign up]	2. Hiển thị error message dưới field mà bạn vừa nhập bỏ trống : " Please input your email", "Please enter password". => Đăng ký không thành công	Passed	Passed
IV. Impact/ Other Cases					
DK_0 28	Check đăng nhập thành công bằng tài khoản vừa mới tạo thành công	1. Đăng ký 1 account thành công 2. Mở trình duyệt khác rồi di chuyển đến màn login , dùng tài khoản vừa đăng ký để login	2. Login thành công	Passed	Passed
Function Name	Đăng nhập				

Test Result	Chrome	Firefox			
Total	0	0			
Passed	0	0			
Failed	0	0			
Not Run	0	0			
NA	0	0			
1. Kiểm tra giao diện màn hình 'Đăng nhập'					
DL_001	Kiểm tra giao diện mặc định của màn hình 'Đăng nhập'	1. Điều hướng đến màn hình 'Đăng nhập' 2. Kiểm tra giao diện mặc định	Giao diện mặc định hiển thị đúng với các trường: Email, Mật khẩu, nút 'Đăng nhập', link 'Quên mật khẩu'	Passed	Passed
DL_002	Kiểm tra bố cục các trường trên form đăng nhập	1. Kiểm tra bố cục các trường trên màn hình 'Đăng nhập'	Các trường được sắp xếp hợp lý	Passed	Passed
DL_003	Kiểm tra điều hướng khi đã đăng nhập	1. Đăng nhập thành công 2. Truy cập lại trang đăng nhập	Hệ thống tự động chuyển hướng đến trang tài khoản	Passed	Passed
2. Kiểm tra validation cho form đăng nhập					
DL_004	Trường 'Email' để trống	1. Để trống trường 'Email' 2. Nhập mật khẩu hợp lệ 3. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Vui lòng nhập email của bạn'	Passed	Passed
DL_005	Trường 'Mật khẩu' để trống	1. Nhập email hợp lệ 2. Để trống trường 'Mật khẩu' 3. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Vui lòng nhập mật khẩu'	Passed	Passed
DL_006	Email không đúng định dạng	1. Nhập email không hợp lệ (ví dụ: 'invalid-email') 2. Nhập mật khẩu hợp	Hiển thị thông báo lỗi: 'Vui lòng nhập địa chỉ email hợp'	Passed	Passed

		lệ 3. Nhấn nút 'Đăng nhập'	lệ!'		
DL_007	Đăng nhập với email không tồn tại	1. Nhập email không tồn tại trong hệ thống 2. Nhập mật khẩu bất kỳ 3. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Email hoặc Mật khẩu không đúng!'	Passed	Passed
DL_008	Đăng nhập với mật khẩu sai	1. Nhập email hợp lệ 2. Nhập mật khẩu sai 3. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Email hoặc Mật khẩu không đúng!'	Passed	Passed
3. Chức năng đăng nhập					
DL_009	Đăng nhập thành công với tài khoản hợp lệ	1. Nhập email và mật khẩu hợp lệ 2. Nhấn nút 'Đăng nhập'	Hiển thị thông báo 'Đăng nhập thành công!' và chuyển hướng đến trang tài khoản	Passed	Passed
DL_010	Đăng nhập bằng phím 'Enter'	1. Nhập email và mật khẩu hợp lệ 2. Nhấn phím 'Enter'	Hiển thị thông báo 'Đăng nhập thành công!' và chuyển hướng đến trang tài khoản	Passed	Passed
DL_011	Đăng nhập với tài khoản chưa được phê duyệt (pending)	1. Nhập email và mật khẩu của tài khoản có status 'pending' 2. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Khách hàng chưa được phê duyệt'	Passed	Passed
DL_012	Đăng nhập với tài khoản bị từ chối (rejected)	1. Nhập email và mật khẩu của tài khoản có status 'rejected' 2. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Khách hàng đã bị từ chối'	Passed	Passed
DL_013	Đăng nhập với tài khoản không hoạt động (inactive)	1. Nhập email và mật khẩu của tài khoản có active = 0 2. Nhấn nút 'Đăng nhập'	Hiển thị thông báo lỗi: 'Khách hàng không hoạt động!'	Passed	Passed
DL_014	Kiểm tra merge giỏ hàng guest sau khi đăng nhập	1. Thêm sản phẩm vào giỏ hàng khi chưa đăng nhập 2. Đăng nhập thành công	Sản phẩm từ giỏ hàng guest được merge vào giỏ hàng của tài khoản đã đăng	Passed	Passed

		3. Kiểm tra giỏ hàng	nhập		
DL_015	Đăng nhập với đăng nhập bên thứ ba (social login)	1. Nhấn nút đăng nhập bằng tài khoản social (nếu có) 2. Xác thực qua tài khoản social	Đăng nhập thành công và chuyển hướng đến trang tài khoản	Passed	Passed

ID	Summary	Steps	Expected Output	Test Results	
				Chrome	Firefox
I. Kiểm tra Luồng Thanh toán Thành công (Positive Flow)					
TT-001	Thanh toán thành công bằng Tiền mặt (COD)	1. Thêm sản phẩm vào giỏ hàng. 2. Tiến hành Checkout, nhập đầy đủ thông tin giao hàng hợp lệ. 3. Chọn phương thức thanh toán "Tiền mặt khi nhận hàng (COD)". 4. Nhấn nút [Hoàn tất Đơn hàng].	4. Hiển thị thông báo "Đặt hàng thành công". Đơn hàng được tạo với trạng thái 'PENDING' (chờ xử lý) hoặc 'AWAITING PAYMENT'.	Passed	Passed
TT-002	Thanh toán thành công bằng Cổng thanh toán (ví dụ: VNPay/Momo)	1. Thêm sản phẩm vào giỏ hàng. 2. Tiến hành Checkout, chọn phương thức Thanh toán qua cổng (Online Payment). 3. Hoàn tất giao dịch tại cổng thanh toán (dùng tài khoản Test thành công).	3. Hệ thống redirect về trang xác nhận đơn hàng. 4. Đơn hàng được tạo với trạng thái 'PAID' (Đã thanh toán) và tồn kho được trừ vĩnh viễn (Hard Deduct).	Passed	Passed
TT-003	Kiểm tra việc trừ tồn kho sau khi thanh toán thành công	1. Kiểm tra tồn kho ban đầu của SP X. 2. Thực hiện Thanh toán thành công (TT_002) mua N sản phẩm X. 3. Kiểm tra lại tồn kho của SP X.	3. Tồn kho mới = Tồn kho ban đầu - N.	Passed	Passed
II. Kiểm tra Xử lý Lỗi Thanh toán (Negative Flow)					
TT-004	Thanh toán	1. Tiến hành	2. Hệ thống	Passed	Passed

	thất bại (tại cổng thanh toán)	Thanh toán qua cổng (Online Payment). 2. Tại giao diện cổng thanh toán, chọn hủy giao dịch hoặc dùng tài khoản Test thất bại.	redirect về trang thanh toán với thông báo: "Giao dịch thất bại". 3. Đơn hàng được tạo với trạng thái 'FAILED' hoặc không được tạo. 4. Tồn kho tạm khóa (Soft Lock) phải được mở khóa.		
TT_005	Thanh toán khi sản phẩm hết hàng (Race Condition)	1. Tồn kho SP X = 1. 2. User A bắt đầu thanh toán SP X. 3. User B hoàn tất thanh toán SP X (trước A).	3. Đơn hàng của User B thành công. Đơn hàng của User A bị từ chối ở bước xác nhận cuối cùng với thông báo: "Sản phẩm X không đủ số lượng trong kho".	Passed	Passed
TT-006	Để trống trường bắt buộc (ví dụ: Địa chỉ)	1. Bỏ trống trường Địa chỉ giao hàng. 2. Nhấn nút [Hoàn tất Đơn hàng].	2. Hiển thị thông báo lỗi dưới trường bị thiếu: "Vui lòng nhập địa chỉ giao hàng".	Passed	Passed
III. Kiểm tra Xuất Hóa đơn (Invoice Generation)					
IV-001	Tạo Hóa đơn và lưu trữ dữ liệu thành công	1. Hoàn tất một đơn hàng (TT_002). 2. Truy cập vào Module Quản lý Đơn hàng/Kế toán.	2. Hệ thống tạo một bản ghi Hóa đơn (Invoice) tương ứng với Đơn hàng vừa tạo. 3. Hóa đơn có các trường dữ liệu bắt buộc (Mã Hóa đơn, Ngày xuất, Tổng tiền, VAT).	Passed	Passed

IV-002	Xuất và Gửi Hóa đơn (PDF>Email)	<p>1. Hoàn tất một đơn hàng (TT_002). 2. Kiểm tra hộp thư Email của khách hàng.</p>	<p>2. Khách hàng nhận được Email xác nhận đơn hàng. 3. Email đính kèm file Hóa đơn/Biên lai ở định dạng PDF.</p>	Passed	Passed
IV-003	Hóa đơn hiển thị đúng chi tiết đơn hàng	<p>1. Mở Hóa đơn PDF/HTML đã tạo (IV_002). 2. So sánh chi tiết trong Hóa đơn với chi tiết Đơn hàng (giá, số lượng, thuế).</p>	<p>2. Thông tin chính xác: Tên sản phẩm, số lượng, đơn giá, tổng tiền, thuế VAT (nếu có), chi phí vận chuyển.</p>	Passed	Passed
IV_004	Kiểm tra trạng thái Hóa đơn khi đơn hàng bị hủy	<p>1. Thực hiện một đơn hàng thanh toán COD (TT_001). 2. Trên hệ thống Admin, hủy đơn hàng đó.</p>	<p>2. Bản ghi Hóa đơn (nếu đã tạo) phải được đánh dấu là 'CANCELED' hoặc không thể xuất bản.</p>	Passed	Passed

CHƯƠNG 4. KẾT QUẢ VÀ ĐÁNH GIÁ

4.1 Tóm tắt kết quả và lợi ích đạt được

Sau quá trình nghiên cứu, phân tích và triển khai, nhóm đã xây dựng thành công "**Hệ thống bán đồ ăn trực tuyến**" dựa trên kiến trúc **Microservices**. Đề tài đã giải quyết được các mục tiêu ban đầu đề ra cả về mặt nghiệp vụ lẫn kỹ thuật.

4.1.1 Về mặt kỹ thuật

Hệ thống đã hoàn thành các chức năng chính theo kế hoạch đề ra:

- Các microservice hoạt động độc lập và giao tiếp được thông qua HTTP hoặc messaging.
- Cơ sở dữ liệu MySQL kết nối ổn định thông qua Spring Data JPA.
- Hệ thống đăng ký - phát hiện dịch vụ (Eureka) hoạt động chính xác, các service tự động đăng ký và có thể truy cập lẫn nhau.
- Spring Security và JWT được tích hợp để đảm bảo quá trình xác thực và phân quyền an toàn.
- Config Server giúp quản lý cấu hình tập trung, hỗ trợ tái khởi động linh hoạt.

4.1.2 Về mặt thực tiễn

- **Số hóa quy trình bán hàng:** Thay thế hoàn toàn quy trình ghi chép thủ công bằng hệ thống phần mềm, giúp giảm thiểu sai sót và tiết kiệm thời gian cho nhân viên.
- **Minh bạch hóa giao dịch:** Việc tích hợp xuất hóa đơn điện tử và gửi email xác nhận giúp khách hàng an tâm mua sắm, đồng thời hỗ trợ chủ cửa hàng quản lý doanh thu chính xác.
- **Khả năng mở rộng:** Hệ thống sẵn sàng cho việc nâng cấp, mở rộng thêm các tính năng mới (như App Mobile, giao hàng) mà không ảnh hưởng đến vận hành hiện tại.

4.2 Đánh giá kết quả thực hiện

- Hệ thống đạt yêu cầu cả về chức năng lẫn kiến trúc (microservice).
- Tốc độ phản hồi của hệ thống nhanh nhờ sử dụng Spring Boot, kết hợp với HikariCP, tối ưu hóa kết nối DB.
- Khả năng mở rộng và bảo trì tốt: các dịch vụ tách biệt, dễ triển khai độc lập hoặc nâng cấp từng phần.
- Bảo mật được đảm bảo nhờ vào cơ chế xác thực hiện đại (JWT).
- Có thể triển khai thực tế hoặc phát triển thêm các chức năng mới dễ dàng.

4.3 Hạn chế và hướng phát triển

Hạn chế:

- Giao diện hiện tại còn đơn giản, chưa có chức năng nâng cao như biểu đồ, thống kê, thông báo theo thời gian thực.
- Chưa triển khai đầy đủ hệ thống CI/CD hay giám sát log tập trung (ELK stack).
- Một số tính năng còn chạy trên localhost, chưa triển khai cloud thực tế.

Hướng phát triển:

- Nâng cấp giao diện khách hàng bằng các thư viện UI nâng cao (MUI, Ant Design).
- Tích hợp hệ thống CI/CD như Jenkins hoặc GitHub Actions để tự động build và deploy.
- Đưa ứng dụng lên các nền tảng cloud như Docker + Kubernetes, hoặc AWS/GCP.
- Bổ sung module thống kê, báo cáo và phân tích dữ liệu trực quan.
- Tích hợp AI nhằm đưa ra phân tích sản phẩm, hành vi tiêu dùng của khách hàng cho admin. Tích hợp chatbox AI cho khách hàng có thể nhận tư vấn và so sánh sản phẩm nhanh và chính xác nhất.

TÀI LIỆU THAM KHẢO

- [1] Ian Sommerville, *Software Engineering* (Kỹ nghệ phần mềm), 10th Edition, Pearson, 2015.
- [2] Eric Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Addison-Wesley Professional, 2003.
- [3] Spring IO, *Spring Boot Reference Documentation*. [Online]. Available: <https://docs.spring.io/spring-boot/docs/current/reference/html/>.
- [4] Spring Cloud, *Spring Cloud Netflix Eureka & Gateway Documentation*. [Online]. Available: <https://spring.io/projects/spring-cloud>.
- [5] React, *ReactJS Official Documentation*. [Online]. Available: <https://react.dev/>
- [8] Docker, *Docker Documentation - Get Started*. [Online]. Available: <https://docs.docker.com/>.
- [6] GitHub, *GitHub Docs - Collaborating with Pull Requests*. [Online]. Available: <https://docs.github.com/>.
- [7] Atlassian, *Agile Coach - The Ultimate Guide to Agile*. [Online]. Available: <https://www.atlassian.com/agile>.
- [8] Postman, *API Documentation & Testing Guide*. [Online]. Available: <https://learning.postman.com/docs/>.