TÀI LIỆU ĐẶC TẢ YÊU CẦU PHẦN MỀM

(Software Requirement Specification – SRS)

Hệ thống E-commerce Bán Quần Áo Phân Tán

Phiên bản 1.0.0

Nhóm sinh viên: Đinh Thị Ngọc Anh – 22010069

Nguyễn Thanh Phong - 22010251

Ngày tạo, ví dụ: 29/05/2025

Contents

[1. Phạm vi 1](#_Toc200204202)

[1.1 Mục đích 1](#_Toc200204203)

[1.2 Nhận diện 1](#_Toc200204204)

[1.3 Kiến trúc hệ thống 1](#_Toc200204205)

[2. Các yêu cầu chức năng 2](#_Toc200204206)

[2.1 Mô tả hệ thống 2](#_Toc200204207)

[2.2 Các chức năng của hệ thống 2](#_Toc200204208)

[2.2.1 Chức năng Người dùng (User Interface): 2](#_Toc200204209)

[2.2.2 Chức năng Quản trị viên (Admin Panel): 3](#_Toc200204210)

[2.3 Đặc tả use-case 3](#_Toc200204211)

[2.3.1 UC002: Đăng nhập vào hệ thống (User) 3](#_Toc200204212)

[2.3.2 UC006: Thêm sản phẩm vào giỏ hàng (User) 4](#_Toc200204213)

[2.3.3 UC010 & UC011: Tiến hành Đặt hàng và Tạo Đơn hàng (User) 5](#_Toc200204214)

[2.3.4 UC013: Xem lịch sử đơn hàng cá nhân (User) 6](#_Toc200204215)

[2.3.5 UC103 & UC104: Quản lý Người dùng - Xem danh sách và Cập nhật Vai trò (Admin) 7](#_Toc200204216)

[2.3.6 UC110 & UC111 & UC112: Quản lý Sản phẩm - Thêm, Sửa, Xóa (Admin) 8](#_Toc200204217)

[2.3.7 UC114: Quản lý Đơn hàng - Xem danh sách tất cả đơn hàng (Admin) 9](#_Toc200204218)

[2.3.8 UC115: Quản lý Đơn hàng - Xem chi tiết một đơn hàng (Admin) 10](#_Toc200204219)

[2.3.9 UC116 & UC117: Quản lý Đơn hàng - Cập nhật Trạng thái và Địa chỉ Giao hàng (Admin) 10](#_Toc200204220)

[3. Yêu cầu về hiệu năng 12](#_Toc200204221)

[3.1.1 Yêu cầu về Hiệu năng (Performance Requirements) 12](#_Toc200204222)

[3.1.2 Yêu cầu về Bảo mật (Security Requirements) 13](#_Toc200204223)

[3.1.3 Yêu cầu về Tính chịu lỗi và Khả năng phục hồi (Fault Tolerance & Recovery Requirements) 14](#_Toc200204224)

# Phạm vi

## Mục đích

Tài liệu này đặc tả các yêu cầu chức năng và phi chức năng cho Hệ thống E-commerce Bán Quần Áo Phân Tán (sau đây gọi là "Hệ thống"). Mục đích của tài liệu là cung cấp một cái nhìn rõ ràng và đầy đủ về các tính năng, hoạt động mong đợi của hệ thống, làm cơ sở cho việc thiết kế, phát triển và kiểm thử. Tài liệu này hướng đến các bên liên quan bao gồm đội ngũ phát triển, người kiểm thử và giảng viên hướng dẫn.

## Nhận diện

Hệ thống cần xây dựng và kiểm thử là một ứng dụng web E-commerce cho phép người dùng duyệt, tìm kiếm sản phẩm quần áo, quản lý giỏ hàng, đặt hàng và thực hiện thanh toán trực tuyến (mô phỏng qua VNPay sandbox). Hệ thống cũng bao gồm một giao diện quản trị (Admin Panel) cho phép quản lý người dùng, sản phẩm, danh mục và đơn hàng. Hệ thống được xây dựng dựa trên kiến trúc microservices.

## Kiến trúc hệ thống

* **Nền tảng phát triển:**
  + **Backend:** Kiến trúc Microservices sử dụng Node.js với framework NestJS, ngôn ngữ TypeScript.
  + **Frontend (User & Admin):** Ứng dụng Single Page Application (SPA) sử dụng React với Vite, ngôn ngữ TypeScript.
  + **API Gateway:** Nginx được sử dụng làm Reverse Proxy và API Gateway, định tuyến request đến các microservices backend.
* **Lưu trữ dữ liệu:**
  + **Cơ sở dữ liệu chính:** PostgreSQL (hai instance riêng biệt: một cho User & Product, một cho Order).
  + **Caching & Session (Giỏ hàng):** Redis.
* **Giao tiếp bất đồng bộ:**
  + **Message Queue:** RabbitMQ được sử dụng để xử lý các tác vụ bất đồng bộ như gửi thông báo, cập nhật trạng thái đơn hàng sau thanh toán.
* **Containerization & Orchestration (Môi trường Local):**
  + **Docker:** Các microservices và thành phần hạ tầng (PostgreSQL, Redis, RabbitMQ, Nginx, Loki Stack) được đóng gói thành các Docker container.
  + **Docker Compose:** Được sử dụng để định nghĩa, chạy và quản lý môi trường đa container trên máy local.
* **Logging:**
  + Hệ thống logging tập trung sử dụng Loki, Promtail và Grafana.

# Các yêu cầu chức năng

## Mô tả hệ thống

Hệ thống E-commerce Bán Quần Áo Phân Tán là một nền tảng trực tuyến cho phép khách hàng xem, tìm kiếm, chọn mua các sản phẩm thời trang (quần áo, phụ kiện). Người dùng có thể đăng ký tài khoản, quản lý giỏ hàng, thực hiện đặt hàng và thanh toán trực tuyến thông qua cổng thanh toán mô phỏng VNPay.

Hệ thống cũng cung cấp một giao diện quản trị riêng biệt cho phép quản trị viên (Admin) quản lý toàn bộ các khía cạnh của cửa hàng, bao gồm quản lý người dùng, sản phẩm, danh mục sản phẩm và theo dõi, xử lý các đơn hàng.

Kiến trúc microservices được áp dụng nhằm đảm bảo tính linh hoạt, khả năng mở rộng và bảo trì dễ dàng. Các thành phần chính như quản lý người dùng, sản phẩm, giỏ hàng, đơn hàng, thanh toán và thông báo được tách thành các dịch vụ độc lập, giao tiếp với nhau qua API RESTful và hàng đợi tin nhắn (Message Queue).

## Các chức năng của hệ thống

Hệ thống bao gồm hai nhóm chức năng chính: chức năng dành cho Người dùng (Khách hàng) và chức năng dành cho Quản trị viên (Admin).

### Chức năng Người dùng (User Interface):

* **UC001:** Đăng ký tài khoản người dùng mới.
* **UC002:** Đăng nhập vào hệ thống.
* **UC003:** Đăng xuất khỏi hệ thống.
* **UC004:** Xem danh sách sản phẩm (có phân trang, lọc theo danh mục, tìm kiếm theo tên).
* **UC005:** Xem chi tiết thông tin một sản phẩm.
* **UC006:** Thêm sản phẩm vào giỏ hàng.
* **UC007:** Xem giỏ hàng.
* **UC008:** Cập nhật số lượng sản phẩm trong giỏ hàng.
* **UC009:** Xóa sản phẩm khỏi giỏ hàng.
* **UC010:** Tiến hành đặt hàng (Checkout).
* **UC011:** Tạo đơn hàng và nhận URL thanh toán VNPay.
* **UC012:** Xử lý trang trả về (Return URL) sau khi thanh toán qua VNPay.
* **UC013:** Xem lịch sử đơn hàng cá nhân.
* **UC014:** Xem chi tiết một đơn hàng cá nhân.
* **UC015:** (Tùy chọn) Xem và cập nhật thông tin cá nhân (Profile).

### Chức năng Quản trị viên (Admin Panel):

* **UC101:** Đăng nhập với tài khoản Admin.
* **UC102:** Xem Dashboard tổng quan (thống kê cơ bản về doanh thu, đơn hàng, người dùng, sản phẩm).
* **UC103:** Quản lý Người dùng - Xem danh sách tất cả người dùng.
* **UC104:** Quản lý Người dùng - Cập nhật vai trò (role) cho người dùng.
* **UC105:** Quản lý Danh mục - Xem danh sách danh mục.
* **UC106:** Quản lý Danh mục - Thêm danh mục mới.
* **UC107:** Quản lý Danh mục - Sửa thông tin danh mục.
* **UC108:** Quản lý Danh mục - Xóa danh mục.
* **UC109:** Quản lý Sản phẩm - Xem danh sách tất cả sản phẩm.
* **UC110:** Quản lý Sản phẩm - Thêm sản phẩm mới (bao gồm thông tin, giá, tồn kho, danh mục, ảnh).
* **UC111:** Quản lý Sản phẩm - Sửa thông tin sản phẩm.
* **UC112:** Quản lý Sản phẩm - Xóa sản phẩm.
* **UC113:** Quản lý Sản phẩm - Cập nhật tồn kho cho sản phẩm.
* **UC114:** Quản lý Đơn hàng - Xem danh sách tất cả đơn hàng.
* **UC115:** Quản lý Đơn hàng - Xem chi tiết một đơn hàng.
* **UC116:** Quản lý Đơn hàng - Cập nhật trạng thái đơn hàng (ví dụ: Pending -> Processing -> Completed/Cancelled).
* **UC117:** Quản lý Đơn hàng - Cập nhật địa chỉ giao hàng cho đơn hàng.

## Đặc tả use-case

Đặc tả chi tiết cho từng use-case.

### UC002: Đăng nhập vào hệ thống (User)

* **a. Tên Use-case:** Đăng nhập vào hệ thống
* **b. Tác nhân (Actor):** Người dùng chưa đăng nhập (Guest User), Người dùng đã đăng ký (Registered User).
* **c. Mô tả ngắn gọn:** Use-case này cho phép người dùng đã có tài khoản truy cập vào các chức năng yêu cầu xác thực của hệ thống bằng cách cung cấp email và mật khẩu.
* **d. Tiền điều kiện:**
  + Người dùng đã có tài khoản được đăng ký thành công trên hệ thống.
  + Người dùng đang ở trang đăng nhập.
* **e. Luồng sự kiện chính (Basic Flow):**
* Người dùng nhập Email vào ô "Email".
* Người dùng nhập Mật khẩu vào ô "Mật khẩu".
* Người dùng nhấp vào nút "Đăng Nhập".
* Hệ thống (Frontend) gửi yêu cầu đăng nhập (chứa email và mật khẩu) đến API Gateway.
* API Gateway chuyển tiếp yêu cầu đến user-service.
* User-service xác thực thông tin email và mật khẩu với dữ liệu trong cơ sở dữ liệu.
* Nếu thông tin hợp lệ, user-service tạo một JSON Web Token (JWT) và trả về cho API Gateway.
* API Gateway trả JWT về cho Frontend.
* Frontend lưu trữ JWT (ví dụ: trong localStorage) và cập nhật trạng thái người dùng là đã đăng nhập.
* Hệ thống chuyển hướng người dùng đến trang chủ (hoặc trang trước đó họ muốn truy cập nếu có lưu).
* **f. Luồng sự kiện thay thế (Alternative Flows):**
  + **AE1: Thông tin đăng nhập không chính xác:**
    - Tại bước 7 của luồng chính, nếu user-service xác định email hoặc mật khẩu không đúng.
    - User-service trả về lỗi (ví dụ: 401 Unauthorized).
    - Frontend hiển thị thông báo lỗi "Email hoặc mật khẩu không chính xác." cho người dùng. Use-case kết thúc.
* **g. Hậu điều kiện:**
  + **Thành công:** Người dùng được xác thực thành công, chuyển sang trạng thái đã đăng nhập, và được chuyển hướng đến trang phù hợp. JWT được lưu trữ ở client.
  + **Thất bại:** Người dùng vẫn ở trang đăng nhập và nhận được thông báo lỗi.

### UC006: Thêm sản phẩm vào giỏ hàng (User)

* **a. Tên Use-case:** Thêm sản phẩm vào giỏ hàng
* **b. Tác nhân:** Người dùng đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép người dùng đã đăng nhập thêm một hoặc nhiều sản phẩm vào giỏ hàng ảo của họ để chuẩn bị cho việc đặt hàng.
* **d. Tiền điều kiện:**
  + Người dùng đã đăng nhập vào hệ thống.
  + Người dùng đang ở trang danh sách sản phẩm hoặc trang chi tiết sản phẩm.
  + Sản phẩm muốn thêm vẫn còn hàng (stockQuantity > 0).
* **e. Luồng sự kiện chính:**
  + - Người dùng chọn một sản phẩm và nhấp vào nút "Thêm vào giỏ" (hoặc chọn số lượng và nhấp nút).
    - Hệ thống (Frontend) gửi yêu cầu thêm sản phẩm (chứa productId và quantity) đến API Gateway.
    - API Gateway chuyển tiếp yêu cầu đến cart-service (kèm theo thông tin xác thực người dùng từ JWT).
    - Cart-service xác định giỏ hàng của người dùng dựa trên userId (lấy từ token).
    - Cart-service kiểm tra xem sản phẩm đã có trong giỏ hàng của người dùng chưa:
      * Nếu đã có: Tăng số lượng sản phẩm đó trong giỏ hàng.
      * Nếu chưa có: Thêm sản phẩm mới vào giỏ hàng với số lượng được chọn.
    - Cart-service lưu trữ thông tin giỏ hàng đã cập nhật vào Redis.
    - Cart-service trả về thông tin giỏ hàng đã cập nhật (hoặc chỉ thông báo thành công).
    - Frontend cập nhật giao diện giỏ hàng (ví dụ: số lượng sản phẩm trên icon giỏ hàng, hiển thị thông báo "Đã thêm sản phẩm vào giỏ").
* **f. Luồng sự kiện thay thế:**
  + **AE1: Sản phẩm không đủ tồn kho (Kiểm tra ở Frontend hoặc Backend):**
    - * Nếu người dùng cố gắng thêm số lượng vượt quá tồn kho hiện tại.
      * Frontend (nếu có kiểm tra trước) hoặc cart-service (nếu gọi product-service để kiểm tra lại) phát hiện.
      * Hiển thị thông báo lỗi "Sản phẩm không đủ số lượng tồn kho." Use-case kết thúc hoặc không cho phép thêm vượt quá.
  + **AE2: Người dùng chưa đăng nhập:**
    - * Nếu người dùng chưa đăng nhập mà nhấn nút "Thêm vào giỏ".
      * Hệ thống yêu cầu người dùng đăng nhập hoặc chuyển hướng đến trang đăng nhập.
  + Sản phẩm được thêm/cập nhật số lượng thành công trong giỏ hàng của người dùng trên Redis.
  + Giao diện người dùng phản ánh sự thay đổi trong giỏ hàng.

### UC010 & UC011: Tiến hành Đặt hàng và Tạo Đơn hàng (User)

* **a. Tên Use-case:** Tiến hành Đặt hàng và Tạo Đơn hàng
* **b. Tác nhân:** Người dùng đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép người dùng chuyển các sản phẩm từ giỏ hàng của họ thành một đơn hàng chính thức, cung cấp thông tin giao hàng, và nhận được hướng dẫn thanh toán.
* **d. Tiền điều kiện:**
  1. Người dùng đã đăng nhập vào hệ thống.
  2. Giỏ hàng của người dùng có ít nhất một sản phẩm.
  3. Người dùng đang ở trang Giỏ hàng hoặc trang Checkout.
* **e. Luồng sự kiện chính:**
  1. Người dùng xem lại các sản phẩm trong giỏ hàng và tổng tiền.
  2. Người dùng nhấp vào nút "Tiến hành đặt hàng" (hoặc tương tự).
  3. Hệ thống chuyển hướng người dùng đến trang Checkout (nếu chưa ở đó).
  4. Người dùng nhập (hoặc xác nhận) thông tin giao hàng (tên, địa chỉ, số điện thoại).
  5. Người dùng nhấp vào nút "Đặt hàng và Thanh toán" (hoặc tương tự).
  6. Hệ thống (Frontend) gửi yêu cầu tạo đơn hàng (chứa thông tin giao hàng và có thể là danh sách items từ giỏ hàng hiện tại) đến API Gateway.
  7. API Gateway chuyển tiếp yêu cầu đến order-service (kèm thông tin xác thực người dùng).
  8. **Order-service thực hiện các bước sau** 
     + a. Gọi cart-service để lấy thông tin chi tiết giỏ hàng hiện tại của người dùng (để đảm bảo tính nhất quán và lấy giá cuối cùng).
     + b. Với mỗi sản phẩm trong giỏ hàng, gọi product-service để **kiểm tra lại tồn kho**. Nếu bất kỳ sản phẩm nào không đủ, toàn bộ transaction sẽ thất bại và trả lỗi về cho người dùng.
     + c. Nếu tất cả sản phẩm đều đủ tồn kho, tạo một bản ghi Order mới trong database với trạng thái "Pending" (Chờ thanh toán) và lưu các OrderItem.
     + d. Gọi product-service để **giảm số lượng tồn kho** tương ứng cho các sản phẩm đã đặt. Nếu bước này thất bại, transaction cũng sẽ rollback.
     + e. (Sau khi transaction được commit thành công) Gọi cart-service để xóa giỏ hàng của người dùng.
  + Nếu transaction thành công:
    - Order-service trả về thông tin đơn hàng vừa tạo (bao gồm orderId và totalAmount) cho API Gateway.
    - Order-service **publish sự kiện order.created** lên RabbitMQ.
* API Gateway trả thông tin đơn hàng về cho Frontend.
* Hệ thống (Frontend) nhận orderId và totalAmount, sau đó gửi yêu cầu đến API Gateway để tạo URL thanh toán VNPay (chuyển đến payment-service).
* Payment-service tạo URL thanh toán VNPay và trả về cho Frontend.
* Frontend nhận URL thanh toán và **tự động điều hướng** người dùng sang cổng thanh toán VNPay.
* **f. Luồng sự kiện thay thế:**
  + - **AE1: Giỏ hàng trống:**
      * Tại bước 5, nếu người dùng nhấn "Đặt hàng" khi giỏ hàng trống.
      * Hệ thống hiển thị thông báo "Giỏ hàng của bạn đang trống." Use-case kết thúc.
    - **AE2: Sản phẩm không đủ tồn kho (phát hiện bởi order-service):**
      * Tại bước 8b, nếu order-service phát hiện một sản phẩm không đủ tồn kho.
      * Order-service rollback transaction và trả về lỗi 400 (Bad Request) với thông báo chi tiết.
      * Frontend hiển thị thông báo lỗi cho người dùng (ví dụ: "Sản phẩm [Tên sản phẩm] không đủ số lượng. Vui lòng cập nhật giỏ hàng."). Use-case kết thúc.
    - **AE3: Lỗi khi lưu đơn hàng hoặc giảm tồn kho:**
      * Tại bước 8c hoặc 8d, nếu có lỗi xảy ra.
      * Order-service rollback transaction và trả về lỗi 500 (Internal Server Error).
      * Frontend hiển thị thông báo lỗi chung "Đã có lỗi xảy ra trong quá trình tạo đơn hàng. Vui lòng thử lại." Use-case kết thúc.
    - **AE4: Lỗi khi tạo URL thanh toán:**
      * Tại bước 12, nếu payment-service không thể tạo URL.
      * Frontend hiển thị thông báo lỗi "Không thể tạo yêu cầu thanh toán. Vui lòng thử lại." Use-case kết thúc.
* **g. Hậu điều kiện:**
  + - **Thành công:** Đơn hàng được tạo trong hệ thống với trạng thái "Pending", tồn kho sản phẩm được cập nhật, giỏ hàng của người dùng được xóa, sự kiện order.created được publish, và người dùng được chuyển hướng đến cổng thanh toán.
    - **Thất bại:** Đơn hàng không được tạo, không có thay đổi nào trong hệ thống (do rollback transaction), người dùng nhận được thông báo lỗi.

### UC013: Xem lịch sử đơn hàng cá nhân (User)

* **a. Tên Use-case:** Xem lịch sử đơn hàng cá nhân
* **b. Tác nhân:** Người dùng đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép người dùng xem danh sách các đơn hàng họ đã đặt trước đó.
* **d. Tiền điều kiện:**
  + Người dùng đã đăng nhập vào hệ thống.
* **e. Luồng sự kiện chính:**
  + Người dùng điều hướng đến trang "Lịch sử đơn hàng" (hoặc "Đơn hàng của tôi").
  + Hệ thống (Frontend) gửi yêu cầu lấy danh sách đơn hàng (kèm token xác thực) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến order-service.
  + Order-service truy vấn cơ sở dữ liệu để lấy tất cả các đơn hàng thuộc về userId của người dùng hiện tại.
  + Order-service trả về danh sách đơn hàng (ví dụ: ID đơn hàng, ngày đặt, tổng tiền, trạng thái) cho API Gateway.
  + API Gateway trả dữ liệu về cho Frontend.
  + Frontend hiển thị danh sách đơn hàng cho người dùng, có thể sắp xếp theo ngày đặt (mới nhất trước). Mỗi đơn hàng có thể có link để xem chi tiết.
* **f. Luồng sự kiện thay thế:**
  + **AE1: Chưa có đơn hàng nào:**
    - Tại bước 5, nếu không tìm thấy đơn hàng nào cho người dùng.
    - Hệ thống hiển thị thông báo "Bạn chưa có đơn hàng nào."
* **g. Hậu điều kiện:**
  + Danh sách đơn hàng của người dùng được hiển thị.

### UC103 & UC104: Quản lý Người dùng - Xem danh sách và Cập nhật Vai trò (Admin)

* **a. Tên Use-case:** Quản lý Người dùng (Xem danh sách, Cập nhật Vai trò)
* **b. Tác nhân:** Quản trị viên (Admin) đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép Admin xem danh sách tất cả người dùng trong hệ thống và có khả năng thay đổi vai trò (ví dụ: từ User sang Admin hoặc ngược lại) của một người dùng cụ thể.
* **d. Tiền điều kiện:**
  + Admin đã đăng nhập vào hệ thống với tài khoản có quyền quản trị.
  + Admin đang ở trang quản lý người dùng.
* **e. Luồng sự kiện chính (Xem danh sách):**
  + Admin truy cập vào mục "Quản lý Người dùng".
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu lấy danh sách tất cả người dùng (kèm token xác thực Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến user-service (ví dụ: endpoint /admin/users/all).
  + User-service (sau khi xác thực Admin) truy vấn cơ sở dữ liệu để lấy danh sách tất cả người dùng với các thông tin cần thiết (ID, Email, Tên, Họ, Vai trò, Ngày tạo).
  + User-service trả về danh sách người dùng cho API Gateway.
  + API Gateway trả dữ liệu về cho Frontend Admin Panel.
  + Frontend Admin Panel hiển thị danh sách người dùng dưới dạng bảng, bao gồm các tùy chọn tìm kiếm, lọc và phân trang.
* **f. Luồng sự kiện chính (Cập nhật Vai trò):**
  + Admin chọn một người dùng từ danh sách.
  + Admin chọn vai trò mới cho người dùng đó (ví dụ: qua một dropdown hoặc nút "Thay đổi vai trò").
  + Admin xác nhận thay đổi.
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu cập nhật vai trò (chứa userId và newRole, kèm token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến user-service (ví dụ: endpoint PATCH /admin/users/:userId/role).
  + User-service (sau khi xác thực Admin) tìm người dùng theo userId và cập nhật trường role của họ trong cơ sở dữ liệu.
  + User-service trả về thông tin người dùng đã được cập nhật (hoặc thông báo thành công).
  + Frontend Admin Panel cập nhật lại giao diện (ví dụ: hiển thị vai trò mới trong bảng) và thông báo thành công cho Admin.
* **g. Luồng sự kiện thay thế:**
  + **AE1: Admin không có quyền:**
    1. Nếu một người dùng không phải Admin cố gắng truy cập chức năng này.
    2. Hệ thống từ chối truy cập và có thể chuyển hướng về trang không có quyền.
  + **AE2: Người dùng không tồn tại (khi cập nhật vai trò):**
    1. Tại bước f6, nếu userId không tìm thấy.
    2. User-service trả về lỗi 404.
    3. Frontend Admin Panel hiển thị thông báo lỗi "Người dùng không tồn tại."
  + **AE3: Lỗi cập nhật vai trò:**
    1. Tại bước f6, nếu có lỗi khi lưu vào database.
    2. User-service trả về lỗi 500.
    3. Frontend Admin Panel hiển thị thông báo lỗi "Không thể cập nhật vai trò người dùng."
* **h. Hậu điều kiện:**
  + **Xem danh sách:** Danh sách người dùng được hiển thị chính xác.
  + **Cập nhật vai trò thành công:** Vai trò của người dùng được thay đổi trong cơ sở dữ liệu và hiển thị trên giao diện.
  + **Cập nhật vai trò thất bại:** Vai trò người dùng không thay đổi, Admin nhận được thông báo lỗi.
* **i. Yêu cầu đặc biệt:**
  + Cần có cơ chế xác nhận cẩn thận trước khi thay đổi vai trò của người dùng, đặc biệt là khi gán quyền Admin.

### UC110 & UC111 & UC112: Quản lý Sản phẩm - Thêm, Sửa, Xóa (Admin)

* **a. Tên Use-case:** Quản lý Sản phẩm (Thêm, Sửa, Xóa)
* **b. Tác nhân:** Quản trị viên (Admin) đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép Admin thêm sản phẩm mới, chỉnh sửa thông tin sản phẩm hiện có, và xóa sản phẩm khỏi hệ thống.
* **d. Tiền điều kiện:**
  + Admin đã đăng nhập vào hệ thống với tài khoản có quyền quản trị.
  + Admin đang ở trang quản lý sản phẩm.
  + (Đối với Sửa/Xóa) Sản phẩm cần Sửa/Xóa phải tồn tại.
  + (Đối với Thêm/Sửa) Danh mục sản phẩm đã tồn tại (nếu sản phẩm được gán vào danh mục).
* **e. Luồng sự kiện chính (Thêm sản phẩm):**
  + Admin nhấp vào nút "Thêm Sản phẩm mới".
  + Hệ thống hiển thị form thêm sản phẩm, yêu cầu nhập: Tên, Mô tả, Giá, Số lượng tồn kho ban đầu, Hình ảnh (upload hoặc link), chọn Danh mục từ danh sách có sẵn.
  + Admin điền đầy đủ thông tin và nhấp "Lưu".
  + Hệ thống (Frontend Admin Panel) gửi dữ liệu sản phẩm mới (kèm token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến product-service (ví dụ: endpoint POST /admin/products hoặc POST /products có AdminGuard).
  + Product-service (sau khi xác thực Admin):
    - Kiểm tra tính hợp lệ của dữ liệu (validation).
    - Kiểm tra categoryId có hợp lệ không.
    - (Nếu có upload ảnh) Xử lý file ảnh, lưu trữ và lấy đường dẫn.
    - Tạo bản ghi sản phẩm mới trong cơ sở dữ liệu.
  + Product-service trả về thông tin sản phẩm vừa tạo (hoặc thông báo thành công).
  + Frontend Admin Panel hiển thị thông báo thành công và có thể chuyển hướng về trang danh sách sản phẩm.
* **f. Luồng sự kiện chính (Sửa sản phẩm):**
  + Admin chọn một sản phẩm từ danh sách và nhấp "Sửa".
  + Hệ thống hiển thị form sửa sản phẩm, điền sẵn thông tin hiện tại của sản phẩm đó.
  + Admin thay đổi các thông tin cần thiết và nhấp "Lưu".
  + Hệ thống (Frontend Admin Panel) gửi dữ liệu sản phẩm đã cập nhật (kèm token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến product-service (ví dụ: PATCH /admin/products/:id hoặc PATCH /products/:id có AdminGuard).
  + Product-service (sau khi xác thực Admin) tìm sản phẩm theo ID, cập nhật thông tin, xử lý ảnh (nếu có thay đổi), và lưu vào cơ sở dữ liệu.
  + Product-service trả về thông tin sản phẩm đã cập nhật.
  + Frontend Admin Panel hiển thị thông báo thành công và cập nhật lại danh sách hoặc chi tiết sản phẩm.
* **g. Luồng sự kiện chính (Xóa sản phẩm):**
  + Admin chọn một sản phẩm từ danh sách và nhấp "Xóa".
  + Hệ thống hiển thị hộp thoại xác nhận "Bạn có chắc chắn muốn xóa sản phẩm này?".
  + Admin xác nhận.
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu xóa sản phẩm (kèm token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến product-service (ví dụ: DELETE /admin/products/:id hoặc DELETE /products/:id có AdminGuard).
  + Product-service (sau khi xác thực Admin) tìm và xóa (hoặc đánh dấu đã xóa - soft delete) sản phẩm khỏi cơ sở dữ liệu.
  + Product-service trả về thông báo thành công.
  + Frontend Admin Panel hiển thị thông báo thành công và xóa sản phẩm khỏi danh sách.
* **h. Luồng sự kiện thay thế:**
  + **AE1: Dữ liệu không hợp lệ (khi Thêm/Sửa):**
    - Tại bước e3 hoặc f3, nếu Admin nhập dữ liệu không hợp lệ (ví dụ: giá âm, tên quá dài).
    - Validation ở Frontend hoặc Backend phát hiện.
    - Hiển thị thông báo lỗi chi tiết cho Admin trên form.
  + **AE2: Danh mục không tồn tại (khi Thêm/Sửa):**
    - Nếu Admin chọn một categoryId không tồn tại.
    - Product-service trả về lỗi.
    - Frontend hiển thị thông báo.
  + **AE3: Sản phẩm không tồn tại (khi Sửa/Xóa):**
    - Product-service trả về lỗi 404.
    - Frontend hiển thị thông báo.
  + **AE4: Lỗi upload ảnh (Nếu có):**
    - Hiển thị thông báo lỗi upload.
* **i. Hậu điều kiện:**
  + **Thêm thành công:** Sản phẩm mới được lưu vào DB, cache liên quan được xóa/cập nhật.
  + **Sửa thành công:** Thông tin sản phẩm được cập nhật trong DB, cache liên quan được xóa/cập nhật.
  + **Xóa thành công:** Sản phẩm được xóa (hoặc đánh dấu đã xóa) khỏi DB, cache liên quan được xóa.

### UC114: Quản lý Đơn hàng - Xem danh sách tất cả đơn hàng (Admin)

* **a. Tên Use-case:** Admin Xem danh sách tất cả đơn hàng
* **b. Tác nhân:** Quản trị viên (Admin) đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép Admin xem danh sách toàn bộ các đơn hàng đã được tạo trong hệ thống, bao gồm các thông tin tóm tắt và trạng thái của từng đơn hàng.
* **d. Tiền điều kiện:**
  + Admin đã đăng nhập vào hệ thống với tài khoản có quyền quản trị.
  + Admin đang ở trang quản lý đơn hàng.
* **e. Luồng sự kiện chính:**
  + Admin truy cập vào mục "Quản lý Đơn hàng".
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu lấy danh sách tất cả đơn hàng (kèm token xác thực Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến order-service (ví dụ: endpoint GET /admin/orders/all).
  + Order-service (sau khi xác thực Admin) truy vấn cơ sở dữ liệu để lấy danh sách tất cả đơn hàng, bao gồm các thông tin như: ID Đơn hàng, User ID (hoặc thông tin khách hàng nếu có join), Ngày đặt, Tổng tiền, Trạng thái hiện tại.
  + Order-service trả về danh sách đơn hàng cho API Gateway.
  + API Gateway trả dữ liệu về cho Frontend Admin Panel.
  + Frontend Admin Panel hiển thị danh sách đơn hàng dưới dạng bảng, có các cột thông tin chính. Hệ thống cung cấp các tùy chọn phân trang, tìm kiếm (theo ID đơn hàng, User ID/Email khách hàng) và lọc (theo trạng thái đơn hàng, theo khoảng thời gian đặt hàng).
* **f. Luồng sự kiện thay thế:**
  + **AE1: Không có đơn hàng nào:**
    - Nếu hệ thống chưa có đơn hàng nào.
    - Hệ thống hiển thị thông báo "Chưa có đơn hàng nào trong hệ thống."
* **g. Hậu điều kiện:**
  + Danh sách tất cả đơn hàng được hiển thị cho Admin.
* **h. Yêu cầu đặc biệt:**
  + Phân trang là bắt buộc nếu số lượng đơn hàng lớn để đảm bảo hiệu suất tải trang.
  + Khả năng lọc và tìm kiếm mạnh mẽ giúp Admin dễ dàng quản lý.

### UC115: Quản lý Đơn hàng - Xem chi tiết một đơn hàng (Admin)

* **a. Tên Use-case:** Admin Xem chi tiết một đơn hàng
* **b. Tác nhân:** Quản trị viên (Admin) đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép Admin xem toàn bộ thông tin chi tiết của một đơn hàng cụ thể, bao gồm thông tin khách hàng, sản phẩm đã đặt, lịch sử trạng thái, và thông tin thanh toán (nếu có).
* **d. Tiền điều kiện:**
  + Admin đã đăng nhập.
  + Đơn hàng cần xem phải tồn tại trong hệ thống.
  + Admin đang ở trang danh sách đơn hàng hoặc đã có ID của đơn hàng cần xem.
* **e. Luồng sự kiện chính:**
  + Admin nhấp vào một đơn hàng từ danh sách (hoặc nhập ID đơn hàng để tìm kiếm và xem).
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu lấy chi tiết đơn hàng (kèm orderId và token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến order-service (ví dụ: endpoint GET /admin/orders/:orderId).
  + Order-service (sau khi xác thực Admin) truy vấn cơ sở dữ liệu để lấy toàn bộ thông tin của đơn hàng được chỉ định, bao gồm:
    - Thông tin chung của đơn hàng (ID, User ID, tổng tiền, trạng thái, địa chỉ giao hàng, ngày tạo, ngày cập nhật).
    - Danh sách các OrderItem (sản phẩm, số lượng, giá tại thời điểm mua, tên sản phẩm).
    - (Tùy chọn) Thông tin người dùng liên quan (email, tên).
    - (Tùy chọn) Lịch sử thay đổi trạng thái của đơn hàng.
    - (Tùy chọn) Thông tin giao dịch thanh toán liên quan.
  + Order-service trả về dữ liệu chi tiết của đơn hàng.
  + Frontend Admin Panel hiển thị đầy đủ các thông tin này một cách rõ ràng và có tổ chức.
* **f. Luồng sự kiện thay thế:**
  + **AE1: Đơn hàng không tồn tại:**
    - Nếu orderId không hợp lệ hoặc không tìm thấy.
    - Order-service trả về lỗi 404.
    - Frontend hiển thị thông báo "Không tìm thấy đơn hàng."
* **g. Hậu điều kiện:**
  + Toàn bộ thông tin chi tiết của đơn hàng được hiển thị cho Admin.
* **h. Yêu cầu đặc biệt:**
  + Giao diện cần trực quan, dễ theo dõi các mục thông tin khác nhau của đơn hàng.

### UC116 & UC117: Quản lý Đơn hàng - Cập nhật Trạng thái và Địa chỉ Giao hàng (Admin)

* **a. Tên Use-case:** Admin Cập nhật Trạng thái và Địa chỉ Giao hàng của Đơn hàng
* **b. Tác nhân:** Quản trị viên (Admin) đã đăng nhập.
* **c. Mô tả ngắn gọn:** Cho phép Admin thay đổi trạng thái xử lý của một đơn hàng (ví dụ: từ "Pending" sang "Processing", "Shipping", "Completed", hoặc "Cancelled") và cập nhật địa chỉ giao hàng nếu cần thiết.
* **d. Tiền điều kiện:**
  + Admin đã đăng nhập.
  + Đơn hàng cần cập nhật phải tồn tại và đang ở trạng thái cho phép thay đổi.
  + Admin đang ở trang chi tiết đơn hàng (hoặc có giao diện cập nhật nhanh trên danh sách).
* **e. Luồng sự kiện chính (Cập nhật Trạng thái):**
  + Admin chọn trạng thái mới cho đơn hàng từ danh sách các trạng thái hợp lệ (ví dụ: dropdown).
  + (Tùy chọn) Admin nhập ghi chú cho việc thay đổi trạng thái.
  + Admin nhấp nút "Cập nhật Trạng thái" (hoặc "Lưu thay đổi").
  + Hệ thống (Frontend Admin Panel) gửi yêu cầu cập nhật (chứa orderId, newStatus, adminNotes (nếu có), và token Admin) đến API Gateway.
  + API Gateway chuyển tiếp yêu cầu đến order-service (ví dụ: endpoint PATCH /admin/orders/:orderId).
  + **Order-service thực hiện các bước sau bên trong một database transaction:**
    - a. Kiểm tra tính hợp lệ của việc chuyển đổi trạng thái (ví dụ: không thể chuyển từ "Completed" về "Pending").
    - b. Cập nhật trường status (và adminNotes nếu có) của đơn hàng trong database.
    - c. **Nếu trạng thái mới là "Cancelled" và trạng thái cũ không phải "Cancelled":** Gọi product-service để **hoàn lại tồn kho** cho các sản phẩm trong đơn hàng. Nếu việc hoàn kho thất bại, transaction phải rollback.
  + Nếu transaction thành công:
    - Order-service trả về thông tin đơn hàng đã cập nhật.
    - Order-service **publish sự kiện** (ví dụ: order.status.updated hoặc order.status.cancelled) lên RabbitMQ, chứa thông tin đơn hàng và trạng thái mới/cũ.
  + Frontend Admin Panel cập nhật giao diện, hiển thị trạng thái mới và thông báo thành công.
* **f. Luồng sự kiện chính (Cập nhật Địa chỉ Giao hàng):**
  + Admin chỉnh sửa thông tin địa chỉ giao hàng.
  + Admin nhấp nút "Lưu Địa chỉ".
  + Hệ thống gửi yêu cầu cập nhật (chứa orderId, newShippingAddress, token Admin) đến API Gateway.
  + API Gateway chuyển tiếp đến order-service.
  + Order-service cập nhật shippingAddress cho đơn hàng.
  + Order-service trả về thông tin đơn hàng đã cập nhật.
  + Frontend Admin Panel hiển thị địa chỉ mới và thông báo thành công.
* **g. Luồng sự kiện thay thế:**
  + **AE1: Chuyển đổi trạng thái không hợp lệ:**
    - Order-service phát hiện việc chuyển trạng thái không được phép.
    - Trả về lỗi 400 (Bad Request).
    - Frontend hiển thị thông báo.
  + **AE2: Lỗi hoàn kho khi hủy đơn:**
    - Order-service không thể hoàn kho (ví dụ: product-service lỗi).
    - Transaction được rollback, trạng thái đơn hàng không thay đổi.
    - Order-service trả về lỗi 500.
    - Frontend hiển thị thông báo.
  + **AE3: Đơn hàng không cho phép sửa địa chỉ (ví dụ: đã giao hàng):**
    - Order-service từ chối cập nhật địa chỉ.
    - Trả về lỗi 400.
    - Frontend hiển thị thông báo.
* **h. Hậu điều kiện:**
  + Thông tin trạng thái và/hoặc địa chỉ giao hàng của đơn hàng được cập nhật trong cơ sở dữ liệu.
  + Tồn kho được hoàn lại nếu đơn hàng bị hủy.
  + Sự kiện tương ứng được publish lên RabbitMQ.
  + Giao diện Admin được cập nhật.
* **i. Yêu cầu đặc biệt:**
  + Cần định nghĩa rõ ràng các quy tắc chuyển đổi trạng thái đơn hàng.
  + Logic hoàn kho phải đảm bảo tính chính xác.

# Yêu cầu về hiệu năng

Phần này mô tả các đặc tính chất lượng của hệ thống, không liên quan trực tiếp đến các hành động cụ thể mà người dùng thực hiện, nhưng ảnh hưởng đến trải nghiệm người dùng và sự ổn định của hệ thống.

### Yêu cầu về Hiệu năng (Performance Requirements)

* **NFR001: Thời gian phản hồi API Gateway:**
  + **Mô tả:** Thời gian phản hồi trung bình của API Gateway cho các request từ client (sau khi đã được xác thực và định tuyến) không nên vượt quá 500ms trong điều kiện tải bình thường.
  + **Đo lường:** Thời gian từ khi Gateway nhận request đến khi bắt đầu gửi response về client.
  + **Mức chấp nhận:** 95% request phải có thời gian phản hồi dưới 500ms.
* **NFR002: Thời gian phản hồi của các Microservice Backend:**
  + **Mô tả:** Thời gian xử lý nội bộ của mỗi microservice cho một request điển hình (không bao gồm thời gian mạng giữa các service) không nên vượt quá 300ms.
  + **Đo lường:** Thời gian từ khi service nhận request đến khi gửi response cho Gateway hoặc service gọi.
  + **Mức chấp nhận:** 95% request phải được xử lý dưới 300ms.
* **NFR003: Thời gian tải trang chính (Frontend):**
  + **Mô tả:** Thời gian tải các trang quan trọng của người dùng (Trang chủ, Trang danh sách sản phẩm, Trang chi tiết sản phẩm) không nên vượt quá 3 giây trên kết nối mạng thông thường.
  + **Đo lường:** Sử dụng công cụ Developer Tools của trình duyệt (chỉ số Load, DOMContentLoaded).
  + **Mức chấp nhận:** Trang tải hoàn chỉnh dưới 3 giây.
* **NFR004: Khả năng xử lý người dùng đồng thời (Concurrent Users - Mức cơ bản):**
  + **Mô tả:** Hệ thống (khi chạy local với tài nguyên giới hạn) có thể xử lý được một số lượng nhỏ người dùng đồng thời (ví dụ: 10-20 users) thực hiện các thao tác cơ bản (duyệt sản phẩm, thêm vào giỏ) mà không gặp lỗi nghiêm trọng hoặc suy giảm hiệu năng đáng kể.
  + **Đo lường:** Mô phỏng bằng công cụ kiểm thử tải cơ bản (Postman Collection Runner, Apache JMeter ở mức đơn giản) hoặc kiểm thử thủ công đồng thời.
  + **Mức chấp nhận:** Hệ thống không bị treo, thời gian phản hồi API không tăng quá 200% so với khi có 1 user.
* **NFR005: Hiệu quả Caching:**
  + **Mô tả:** Việc sử dụng Redis caching cho product-service phải giúp giảm đáng kể số lượng query đến database cho các request lấy sản phẩm lặp lại.
  + **Đo lường:** Theo dõi số lượng query database (qua log của TypeORM hoặc công cụ giám sát DB) khi có và không có cache cho cùng một kịch bản truy cập.
  + **Mức chấp nhận:** Giảm ít nhất 50% query DB cho các API được cache sau lần gọi đầu tiên.

### Yêu cầu về Bảo mật (Security Requirements)

* **NFR006: Xác thực Người dùng:**
  + **Mô tả:** Tất cả các truy cập đến tài nguyên yêu cầu đăng nhập (ví dụ: giỏ hàng, đặt hàng, trang cá nhân, API admin) phải được xác thực bằng JWT.
  + **Đo lường:** Kiểm tra các endpoint được bảo vệ có trả về lỗi 401 Unauthorized nếu không có token hoặc token không hợp lệ/hết hạn.
  + **Mức chấp nhận:** 100% các endpoint được bảo vệ phải thực hiện xác thực.
* **NFR007: Phân quyền Truy cập:**
  + **Mô tả:** Người dùng có vai trò "user" không được phép truy cập các API hoặc giao diện dành cho vai trò "admin".
  + **Đo lường:** Thử truy cập các API/trang Admin bằng tài khoản user thường.
  + **Mức chấp nhận:** 100% các truy cập trái phép bị từ chối (ví dụ: lỗi 403 Forbidden).
* **NFR008: Bảo vệ Mật khẩu:**
  + **Mô tả:** Mật khẩu người dùng phải được hash an toàn (sử dụng bcrypt) trước khi lưu vào cơ sở dữ liệu.
  + **Đo lường:** Kiểm tra dữ liệu cột password trong bảng users của database.
  + **Mức chấp nhận:** 100% mật khẩu được hash, không lưu trữ dưới dạng plain text.
* **NFR009: Bảo vệ chống Cross-Site Scripting (XSS) cơ bản:**
  + **Mô tả:** Dữ liệu do người dùng nhập (ví dụ: tên sản phẩm, mô tả, địa chỉ) khi hiển thị lại trên giao diện không được phép thực thi mã JavaScript độc hại.
  + **Đo lường:** Thử nhập các đoạn mã script đơn giản vào các trường input và xem chúng có bị escape (vô hiệu hóa) khi hiển thị hay không. (React thường tự động escape).
  + **Mức chấp nhận:** Không có lỗ hổng XSS cơ bản nào được phát hiện.
* **NFR010: Bảo vệ chống SQL Injection:**
  + **Mô tả:** Hệ thống không bị tấn công SQL Injection thông qua các input từ người dùng.
  + **Đo lường:** Do sử dụng TypeORM (ORM), nguy cơ này được giảm thiểu đáng kể nếu không viết query SQL thô. Kiểm tra không sử dụng query thô với dữ liệu người dùng không được sanitize.
  + **Mức chấp nhận:** Không có lỗ hổng SQL Injection nào được phát hiện.
* **NFR011: Xử lý Token an toàn:**
  + **Mô tả:** JWT được lưu trữ an toàn ở phía client (ví dụ: localStorage) và được truyền qua header Authorization với schema Bearer.
  + **Đo lường:** Kiểm tra cách token được lưu và gửi đi.
  + **Mức chấp nhận:** Token được xử lý theo chuẩn.

### Yêu cầu về Tính chịu lỗi và Khả năng phục hồi (Fault Tolerance & Recovery Requirements)

* **NFR012: Khởi động lại Service tự động (cơ bản):**
  + **Mô tả:** Nếu một microservice (trong Docker container) bị lỗi và dừng đột ngột (không phải lỗi khởi tạo nghiêm trọng), Docker Compose có thể tự động khởi động lại service đó.
  + **Đo lường:** Mô phỏng bằng cách dừng một container service và quan sát.
  + **Mức chấp nhận:** Service được khởi động lại (nếu không có lỗi liên tục trong code).
* **NFR013: API Gateway xử lý lỗi Backend:**
  + **Mô tả:** Nếu một microservice backend không khả dụng hoặc trả về lỗi, API Gateway phải trả về một mã lỗi HTTP phù hợp (ví dụ: 502 Bad Gateway, 503 Service Unavailable) cho client thay vì làm client bị treo.
  + **Đo lường:** Dừng một backend service và gọi API tương ứng qua Gateway.
  + **Mức chấp nhận:** Gateway trả về mã lỗi 50x.
* **NFR014: Health Checks:**
  + **Mô tả:** Mỗi microservice backend và API Gateway phải có endpoint /health để kiểm tra trạng thái hoạt động. docker-compose.yml sử dụng các healthcheck này.
  + **Đo lường:** Trạng thái của các container trong docker ps hiển thị (healthy).
  + **Mức chấp nhận:** Tất cả các service có healthcheck đều chuyển sang trạng thái healthy.
* **NFR015: Message Queue đảm bảo xử lý (RabbitMQ):**
  + **Mô tả:** Nếu notification-service (consumer) tạm thời không hoạt động, các message order.created được publish bởi order-service phải được lưu trữ trong RabbitMQ và được xử lý khi notification-service hoạt động trở lại (do queue và message được cấu hình durable/persistent).
  + **Đo lường:** Dừng notification-service, tạo đơn hàng, kiểm tra RabbitMQ UI thấy message trong queue, khởi động lại notification-service, thấy message được xử lý.
  + **Mức chấp nhận:** Message được xử lý thành công sau khi consumer hoạt động lại.
* **NFR016: Khôi phục Database từ Backup:**
  + **Mô tả:** Có khả năng khôi phục dữ liệu từ file backup SQL đã tạo.
  + **Đo lường:** Thực hiện quy trình backup và restore (thủ công) trên môi trường local.
  + **Mức chấp nhận:** Dữ liệu được khôi phục thành công.