# Mục lục

[Mục lục 1](#_Toc212436817)

[Phạm vi 3](#_Toc212436818)

[1.1 Mục đích: 3](#_Toc212436819)

[1.2 Nhận diện: 3](#_Toc212436820)

[1.3 Kiến trúc hệ thống: 3](#_Toc212436821)

[Các yêu cầu chức năng 4](#_Toc212436822)

[2.1 Mô tả hệ thống – Các tác nhân: 4](#_Toc212436823)

[2.2 Các chức năng của hệ thống: 5](#_Toc212436824)

[2.3 Biểu đồ use-case tổng quan: 6](#_Toc212436825)

[2.4 Đặc tả use-case: 7](#_Toc212436826)

[2.4.1 Đăng nhập vào hệ thống 7](#_Toc212436827)

[2.4.2 Đăng kí vào hệ thống 11](#_Toc212436828)

[2.4.3 Xem, tìm kiếm, lọc danh sách sản phẩm 15](#_Toc212436829)

[2.4.4 Quản lí tài khoản 19](#_Toc212436830)

[2.4.5 Quản lí giỏ hàng 22](#_Toc212436831)

[2.4.6 Đặt hàng 25](#_Toc212436832)

[2.4.7 Thanh toán qua VNPay 27](#_Toc212436833)

[2.4.8 Theo dõi đơn hàng 31](#_Toc212436834)

[2.4.9 Gửi đánh giá sản phẩm 33](#_Toc212436835)

[2.4.10 Thiết kế bánh tùy chỉnh 36](#_Toc212436836)

[2.4.11 Quản lí Đơn hàng (Admin) 40](#_Toc212436837)

[2.4.12 Quản lí Sản phẩm (Admin) 44](#_Toc212436838)

[2.4.12 Quản lí Danh mục Sản phẩm (Admin) 47](#_Toc212436839)

[2.4.13 Quản lí Người dùng (Admin) 50](#_Toc212436840)

[2.4.14 Quản lí Đánh giá (Admin) 54](#_Toc212436841)

[2.4.15 Thống kê Báo cáo (Admin) 57](#_Toc212436842)

[2.5 Sơ đồ Sequence: 61](#_Toc212436843)

[Các yêu cầu phi chức năng 74](#_Toc212436844)

[3.1 Hiệu năng (Performance): 74](#_Toc212436845)

[3.1.1 Tối ưu hóa thời gian tải trang 74](#_Toc212436846)

[3.1.2 Hiệu năng cảm nhận và tương tác 74](#_Toc212436847)

[3.2 Tính khả dụng (Usability): 75](#_Toc212436848)

[3.2.1 Giao diện trực quan và nhất quán 75](#_Toc212436849)

[3.2.2 Tính dễ đọc 75](#_Toc212436850)

[3.2.3 Điều hướng và truy cập thôn tin rõ ràng 75](#_Toc212436851)

[3.3 Độ tin cậy và khả năng chịu lỗi (Reliability vs Fault Tolerance): 75](#_Toc212436852)

[3.3.1 Ưu tiên luồng nghiệp vụ cốt lõi 76](#_Toc212436853)

[3.3.2 Sự bảo vệ toàn vẹn của trạng thái 76](#_Toc212436854)

[3.3.3 Khử nhiễu qua giao tiếp bất đồng bộ 76](#_Toc212436855)

[3.3.4 Xử lí lỗi từ các dịch vụ bên ngoài 76](#_Toc212436856)

[3.4 Bảo mật (Security): 77](#_Toc212436857)

[3.4.1 Xác thực và phân quyền 77](#_Toc212436858)

[3.4.2 Bảo vệ dữ liệu 77](#_Toc212436859)

[3.4.3 Phòng chống các lỗ hổng web phổ biến 78](#_Toc212436860)

[3.5 Pháp luật và tuân thủ (Legal vs Compliance): 78](#_Toc212436861)

[3.5.1 Bảo vệ quyền sở hữu trí tuệ 78](#_Toc212436862)

[3.5.2 Tuân thủ tiêu chuẩn công thanh toán 78](#_Toc212436863)

[3.6 Khả năng bảo trì (Maintainability): 79](#_Toc212436864)

[3.6.1 Kiến trúc phân tách rõ ràng 79](#_Toc212436865)

[3.6.2 Mã nguồn dễ đọc, tính module hóa cao 79](#_Toc212436866)

[3.6.3 Công nghệ hiện đại, cộng đồng mạnh 80](#_Toc212436867)

# Phạm vi

# 1.1 Mục đích:

Tài liệu này dùng để đặc tả các yêu cầu và kiến trúc của hệ thống website bán hàng trực tuyến "Cake Shop". Mục đích chính là cung cấp một cái nhìn tổng quan, chi tiết về hệ thống để phục vụ cho việc phát triển, kiểm thử và bảo trì.

# 1.2 Nhận diện:

Hệ thống cần kiểm thử là trang web bán bánh ngọt trực tuyến "Cake Shop". Đây là một ứng dụng web cho phép khách hàng xem, chọn, tùy chỉnh và đặt mua các sản phẩm bánh ngọt. Hệ thống cũng bao gồm các chức năng quản trị cho phép người quản lý cập nhật sản phẩm, xem đơn hàng và quản lý người dùng.

# 1.3 Kiến trúc hệ thống:

Hệ thống được xây dựng theo kiến trúc Microservices, bao gồm các thành phần chính sau:

* **Frontend (Client-side):**
  + **Nền tảng:** Được xây dựng bằng thư viện **React** và ngôn ngữ **TypeScript**.
  + **Build tool:** Sử dụng **Vite** để tối ưu hóa quá trình phát triển và build ứng dụng.
  + **Môi trường chạy:** Chạy trên trình duyệt web của người dùng (client-side rendering).
* **Backend (Server-side):**
  + **Kiến trúc:** Gồm nhiều microservices độc lập (như User Service, Product Service, Order Service, v.v.).
  + **Nền tảng:** Mỗi microservice được xây dựng bằng framework **NestJS** (một framework Node.js).
  + **API Gateway:** Sử dụng **Nginx** làm reverse proxy và API Gateway để điều hướng các yêu cầu từ frontend đến các microservice tương ứng.
  + **Database:** Hệ thống sử dụng cơ sở dữ liệu quan hệ (SQL) để lưu trữ dữ liệu, có thể là PostgreSQL hoặc MySQL.
* **Triển khai (Deployment):**
  + Toàn bộ hệ thống (frontend, backend services, database) được đóng gói và quản lý bằng **Docker** và **Docker Compose**. Điều này giúp đảm bảo tính nhất quán giữa các môi trường phát triển và triển khai.

# Các yêu cầu chức năng

# 2.1 Mô tả hệ thống – Các tác nhân:

Hệ thống "Cake Shop" là một nền tảng thương mại điện tử chuyên cung cấp các sản phẩm bánh ngọt. Ra đời từ năm 2015, Cake Shop tập trung vào việc tạo ra những chiếc bánh được làm thủ công, không chỉ ngon miệng mà còn là những tác phẩm nghệ thuật tinh tế.

Ứng dụng được thiết kế để mang lại trải nghiệm mua sắm trực tuyến mượt mà và an toàn cho khách hàng, từ việc duyệt sản phẩm, tùy chỉnh bánh theo ý muốn, đến việc đặt hàng và thanh toán. Phía sau giao diện người dùng thân thiện là một hệ thống quản trị mạnh mẽ, cho phép quản trị viên kiểm soát hoàn toàn các khía cạnh của cửa hàng, bao gồm sản phẩm, đơn hàng, khách hàng và xem các báo cáo thống kê chi tiết.

**Các tác nhân (Actors)**

Hệ thống có ba tác nhân chính, mỗi tác nhân có những vai trò và quyền hạn riêng biệt:

1. **Khách vãng lai (Guest):**
   * **Mô tả:** Là bất kỳ ai truy cập vào website mà không cần đăng nhập.
   * **Hành động:**
     + Duyệt xem danh sách các sản phẩm và danh mục.
     + Sử dụng chức năng tìm kiếm và lọc sản phẩm.
     + Xem thông tin chi tiết của từng sản phẩm.
     + Xem các trang thông tin chung như "Giới thiệu" (AboutPage.tsx) và "Điều khoản dịch vụ" (TermsPage.tsx).
     + Đăng ký tài khoản mới để trở thành Người dùng.
2. **Người dùng đã đăng ký (Customer):**
   * **Mô tả:** Là khách hàng đã có tài khoản và đăng nhập vào hệ thống.
   * **Hành động:** Bao gồm tất cả các hành động của Khách vãng lai, cộng thêm:
     + Quản lý thông tin cá nhân trong hồ sơ (ProfilePage.tsx).
     + Thêm/xóa sản phẩm vào giỏ hàng và điều chỉnh số lượng (CartPage.tsx).
     + Sử dụng công cụ để tùy chỉnh bánh đặc biệt (CustomCakeBuilder.tsx).
     + Tiến hành quy trình thanh toán để đặt hàng (CheckoutPage.tsx).
     + Thanh toán qua các phương thức được hỗ trợ (Tiền mặt, VNPay) (paymentApi.ts).
     + Xem lại lịch sử các đơn hàng đã đặt (OrderHistoryPage.tsx) và chi tiết từng đơn hàng (OrderDetailPage.tsx).
     + Gửi đánh giá (review) cho các sản phẩm đã mua (reviewApi.ts).
3. **Quản trị viên (Administrator):**
   * **Mô tả:** Là người dùng có quyền cao nhất, chịu trách nhiệm quản lý và vận hành toàn bộ website.
   * **Hành động:** Bao gồm tất cả các hành động của Người dùng, cộng thêm các chức năng quản trị chuyên sâu thông qua Bảng điều khiển (Admin Dashboard):
     + **Quản lý Đơn hàng:** Xem danh sách toàn bộ đơn hàng, tìm kiếm/lọc đơn hàng, cập nhật trạng thái (Chờ xử lý, Đang xử lý, Hoàn thành, Đã hủy) (AdminOrderListPage.tsx).
     + **Quản lý Sản phẩm:** Thêm, sửa, xóa sản phẩm và các danh mục sản phẩm (AdminProductListPage.tsx, AdminCategoryListPage.tsx).
     + **Quản lý Người dùng:** Xem danh sách người dùng và quản lý thông tin tài khoản (AdminUserListPage.tsx).
     + **Quản lý Đánh giá:** Duyệt và quản lý các đánh giá từ khách hàng (AdminReviewListPage.tsx).
     + **Thống kê:** Truy cập trang tổng quan để xem các số liệu thống kê về doanh thu, đơn hàng, sản phẩm bán chạy (AdminDashboardPage.tsx).

# 2.2 Các chức năng của hệ thống:

**1. Module Quản lý Người dùng & Xác thực**

* **Đăng ký:** Cho phép người dùng mới tạo tài khoản bằng email và mật khẩu.
* **Đăng nhập/Đăng xuất:** Cung cấp cơ chế xác thực an toàn cho Người dùng và Quản trị viên.
* **Quản lý Hồ sơ:** Người dùng có thể xem và cập nhật thông tin cá nhân (tên, địa chỉ, số điện thoại).
* **Phân quyền:** Hệ thống sử dụng các route được bảo vệ (PrivateRoute, AdminRoute) để đảm bảo chỉ những người dùng có quyền phù hợp mới có thể truy cập các chức năng tương ứng.

**2. Module Quản lý Sản phẩm & Danh mục**

* **Hiển thị danh sách sản phẩm:** Liệt kê các sản phẩm với hình ảnh, tên, giá. Hỗ trợ phân trang, tìm kiếm theo tên, và lọc theo danh mục, giá cả.
* **Hiển thị chi tiết sản phẩm:** Cung cấp trang chi tiết cho mỗi sản phẩm, bao gồm nhiều hình ảnh, mô tả, giá, thông tin khuyến mãi và các đánh giá từ khách hàng khác.
* **Quản lý danh mục (Admin):** Admin có thể tạo, đổi tên, và xóa các danh mục sản phẩm (ví dụ: Bánh sinh nhật, Bánh mì, Phụ kiện).
* **Quản lý sản phẩm (Admin):** Admin có thể thêm sản phẩm mới, cập nhật thông tin (tên, mô tả, giá, hình ảnh, số lượng tồn kho), và xóa sản phẩm khỏi hệ thống.

**3. Module Giỏ hàng & Đặt hàng (Checkout)**

* **Quản lý giỏ hàng:** Người dùng có thể thêm sản phẩm vào giỏ, thay đổi số lượng, hoặc xóa sản phẩm khỏi giỏ hàng. Giỏ hàng được lưu lại giữa các phiên đăng nhập.
* **Tùy chỉnh bánh:** Một chức năng đặc biệt cho phép người dùng tự thiết kế chiếc bánh của riêng mình bằng cách chọn kích cỡ, hương vị, và các chi tiết trang trí.
* **Quy trình đặt hàng:** Hướng dẫn người dùng qua các bước: xác nhận sản phẩm, điền thông tin giao hàng, chọn phương thức thanh toán.
* **Xem lại lịch sử đơn hàng:** Người dùng có thể tra cứu tất cả các đơn hàng đã đặt, xem trạng thái và thông tin chi tiết.

**4. Module Thanh toán**

* **Tích hợp đa phương thức:** Hỗ trợ nhiều lựa chọn thanh toán, bao gồm thanh toán khi nhận hàng (COD) và thanh toán trực tuyến qua cổng VNPay.
* **Xử lý thanh toán an toàn:** Giao dịch trực tuyến được xử lý an toàn, hệ thống cập nhật trạng thái đơn hàng dựa trên kết quả trả về từ cổng thanh toán (VnPayReturnPage.tsx).

**5. Module Quản lý Đơn hàng (Admin)**

* **Bảng điều khiển đơn hàng:** Giao diện tập trung cho Admin để xem tất cả đơn hàng theo thời gian thực.
* **Cập nhật trạng thái đơn hàng:** Admin có thể thay đổi trạng thái của đơn hàng (Chờ xử lý -> Đang xử lý -> Hoàn thành). Hệ thống sử dụng các màu sắc khác nhau để dễ dàng phân biệt trạng thái (AdminOrderListPage.tsx).
* **Xem chi tiết đơn hàng:** Admin có thể xem thông tin chi tiết của bất kỳ đơn hàng nào, bao gồm sản phẩm, thông tin khách hàng, và lịch sử trạng thái.

**6. Module Đánh giá & Phản hồi**

* **Gửi đánh giá:** Sau khi mua hàng, người dùng có thể quay lại trang sản phẩm để để lại đánh giá (rating) và bình luận.
* **Quản lý đánh giá (Admin):** Admin có quyền xem, duyệt, và ẩn các đánh giá không phù hợp.

**7. Module Báo cáo & Thống kê (Admin)**

* **Bảng điều khiển tổng quan:** Cung cấp cho Admin cái nhìn nhanh về hiệu suất kinh doanh qua các biểu đồ và số liệu chính: tổng doanh thu, số lượng đơn hàng mới, khách hàng mới, sản phẩm bán chạy nhất.

# 2.3 Biểu đồ use-case tổng quan:

A diagram of a diagram

AI-generated content may be incorrect.

*Hình 2.1: Sơ đồ Usecase tổng quan*

| **Tác nhân (Actor)** | **Use Case** |
| --- | --- |
| **User** (Người dùng) | • Đăng kí  • Đăng nhập  • Quản lí tài khoản  • Xem và tìm kiếm sản phẩm  • Quản lí Giỏ hàng  • Thiết kế Bánh Tùy chỉnh  • Thực hiện Thanh toán  • Theo dõi Đơn hàng  • Viết Đánh giá Sản phẩm |
| **Admin** | • Quản lí Người dùng  • Quản lí Sản phẩm  • Quản lí Danh mục  • Quản lí Đơn hàng (Xem, Cập nhật trạng thái)  • Quản lí Đánh giá  • Xem Thống kê |
| **VNPay** | • Xử lý Giao dịch Thanh toán  • Gửi Tín hiệu Xác nhận Thanh toán (IPN) |
| **Mail Server** | • Gửi Email (Xác nhận đăng kí, xác nhận đơn hàng, cập nhật trạng thái) |

# 2.4 Đặc tả use-case:

# 2.4.1 Đăng nhập vào hệ thống

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Đăng nhập vào hệ thống** |
| --- | --- |
| **Use Case ID** | UC-01 |
| **Mô tả** | Use case này cho phép một người dùng đã có tài khoản (Khách hàng hoặc Quản trị viên) cung cấp thông tin xác thực để truy cập vào các chức năng được bảo vệ của hệ thống Cake Shop. |
| **Tác nhân** | Khách vãng lai (Guest) |
| **Tiền điều kiện** | 1. Tác nhân phải truy cập được vào website Cake Shop.  2. Tác nhân đã có một tài khoản được đăng ký thành công trước đó trong hệ thống. |
| **Hậu điều kiện (Thành công)** | 1. Hệ thống xác thực thành công danh tính của người dùng.  2. Hệ thống tạo và duy trì một phiên làm việc (session) cho người dùng.  3. Người dùng được chuyển hướng đến trang chủ với giao diện đã được cá nhân hóa (ví dụ: hiển thị tên người dùng).  4. Người dùng được cấp quyền truy cập các chức năng tương ứng với vai trò của mình (Customer hoặc Administrator). |
| **Hậu điều kiện (Thất bại)** | 1. Hệ thống hiển thị thông báo lỗi cụ thể.  2. Người dùng vẫn ở lại trang Đăng nhập.  3. Hệ thống không tạo bất kỳ phiên làm việc nào. |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trên thanh điều hướng (Navbar), người dùng nhấp vào nút "Đăng nhập". |  |
| **2** |  | Hệ thống điều hướng người dùng đến trang Đăng nhập (LoginPage.tsx). Giao diện hiển thị hai trường nhập liệu: "Email" và "Mật khẩu", cùng với một nút "Đăng nhập". |
| **3** | Nhập địa chỉ email và mật khẩu hợp lệ đã đăng ký vào các trường tương ứng. |  |
| **4** | Nhấp vào nút "Đăng nhập". |  |
| **5** |  | 1. Hệ thống thu thập dữ liệu từ form.  2. Gửi yêu cầu xác thực (POST request) đến user-service thông qua API endpoint (tham chiếu: authApi.ts). Dữ liệu gửi đi bao gồm email và mật khẩu. |
| **6** |  | 1. user-service tiếp nhận yêu cầu.  2. Kiểm tra xem email có tồn tại trong cơ sở dữ liệu hay không.  3. Nếu tồn tại, so sánh hash của mật khẩu được cung cấp với hash được lưu trong database (user.entity.ts). |
| **7** |  | 1. Dữ liệu xác thực hợp lệ. user-service tạo một JSON Web Token (JWT) chứa thông tin định danh và vai trò của người dùng.  2. Dịch vụ trả về một phản hồi thành công (HTTP 200) kèm theo token (JWT) và thông tin cơ bản của người dùng. |
| **8** |  | 1. Lưu trữ JWT vào bộ nhớ của trình duyệt (ví dụ: localStorage) để duy trì trạng thái đăng nhập.  2. Cập nhật trạng thái xác thực trong toàn ứng dụng (sử dụng AuthContext.tsx).  3. Tự động chuyển hướng người dùng về trang chủ (HomePage.tsx).  4. Cập nhật giao diện thanh điều hướng để hiển thị tên người dùng và menu tài khoản, thay thế cho các nút "Đăng nhập" và "Đăng ký". |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **E1: Sai Email hoặc Mật khẩu** | - **Trigger:** Tại bước 7 của Luồng chính.  - **Kịch bản:**  1. user-service xác định rằng email không tồn tại hoặc mật khẩu không khớp.  2. Dịch vụ trả về một lỗi xác thực (ví dụ: HTTP 401 Unauthorized).  3. Hệ thống (frontend) nhận lỗi và hiển thị một thông báo rõ ràng cho người dùng, ví dụ: "Email hoặc mật khẩu không chính xác."  4. Các trường nhập liệu được giữ nguyên (hoặc chỉ xóa trường mật khẩu). Người dùng có thể thử lại. |
| **E2: Nhập liệu không hợp lệ (Client-side)** | - **Trigger:** Tại bước 3 của Luồng chính.  - **Kịch bản:**<br>  1. Người dùng nhập email không đúng định dạng (ví dụ: test@gmail) hoặc để trống một trong hai trường.  2. Người dùng nhấp "Đăng nhập".  3. Hệ thống thực hiện kiểm tra phía client và ngay lập tức hiển thị thông báo lỗi bên dưới trường nhập liệu không hợp lệ (ví dụ: "Vui lòng nhập một địa chỉ email hợp lệ"). Yêu cầu không được gửi đến server. |
| **E3: Lỗi Hệ thống hoặc Mạng** | - **Trigger:** Tại bước 5 của Luồng chính.  - **Kịch bản:** 1. Hệ thống không thể thiết lập kết nối đến user-service (do lỗi mạng, server down, v.v.).  2. Hệ thống hiển thị một thông báo lỗi chung, ví dụ: "Đã có lỗi xảy ra. Vui lòng thử lại sau." |
| **A1: Người dùng muốn đăng ký tài khoản mới** | - **Trigger:** Tại bước 2 của Luồng chính.  - **Kịch bản:**  1. Người dùng nhận ra mình chưa có tài khoản.  2. Trên trang Đăng nhập, người dùng nhấp vào liên kết "Chưa có tài khoản? Đăng ký ngay".  3. Hệ thống chuyển hướng người dùng đến trang Đăng ký (RegisterPage.tsx).  4. Use case "Đăng ký tài khoản" (UC-02) bắt đầu. |

A diagram of a flowchart

AI-generated content may be incorrect.

# 2.4.2 Đăng kí vào hệ thống

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Đăng ký tài khoản mới** |
| --- | --- |
| **Use Case ID** | UC-02 |
| **Mô tả** | Use case này cho phép một khách vãng lai cung cấp thông tin cá nhân cần thiết để tạo một tài khoản người dùng mới trong hệ thống Cake Shop, cho phép họ truy cập các chức năng dành cho khách hàng đã đăng nhập. |
| **Tác nhân** | Khách vãng lai (Guest) |
| **Tiền điều kiện** | 1. Tác nhân phải truy cập được vào website Cake Shop.  2. Tác nhân chưa có tài khoản trên hệ thống (hoặc muốn tạo một tài khoản mới). |
| **Hậu điều kiện (Thành công)** | 1. Một bản ghi người dùng mới được tạo và lưu trữ trong cơ sở dữ liệu của user-service.  2. Hệ thống tự động xác thực và đăng nhập cho người dùng vừa đăng ký.  3. Hệ thống tạo và duy trì một phiên làm việc (session) cho người dùng.  4. Người dùng được chuyển hướng đến trang chủ với tư cách là người dùng đã đăng nhập. |
| **Hậu điều kiện (Thất bại)** | 1. Không có bản ghi người dùng nào được tạo.  2. Hệ thống hiển thị thông báo lỗi cụ thể.  3. Người dùng vẫn ở lại trang Đăng ký để có thể sửa lỗi và thử lại. |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trên thanh điều hướng (Navbar), người dùng nhấp vào nút "Đăng ký" (hoặc từ trang Đăng nhập, nhấp vào liên kết "Đăng ký ngay"). |  |
| **2** |  | Hệ thống điều hướng người dùng đến trang Đăng ký (RegisterPage.tsx). Giao diện hiển thị các trường: "Họ và Tên", "Email", "Mật khẩu", và "Xác nhận Mật khẩu", cùng với một checkbox "Tôi đồng ý với Điều khoản Dịch vụ" và nút "Đăng ký". |
| **3** | Nhập đầy đủ thông tin hợp lệ vào các trường. |  |
| **4** | Đọc và đánh dấu tích vào ô "Tôi đồng ý với Điều khoản Dịch vụ". |  |
| **5** | Nhấp vào nút "Đăng ký". |  |
| **6** |  | 1. Hệ thống thực hiện kiểm tra phía client: email có đúng định dạng không, mật khẩu có đủ mạnh không, mật khẩu xác nhận có trùng khớp không.  2. Nếu hợp lệ, hệ thống thu thập dữ liệu từ form. |
| **7** |  | 1. Gửi yêu cầu tạo tài khoản (POST request) đến user-service (tham chiếu: userApi.ts, create-user.dto.ts).  2. Dữ liệu gửi đi bao gồm tên, email, và mật khẩu. |
| **8** |  | 1. user-service nhận yêu cầu.  2. Kiểm tra xem email đã tồn tại trong cơ sở dữ liệu hay chưa.  3. Mã hóa (hash) mật khẩu người dùng để đảm bảo an toàn.  4. Tạo một bản ghi người dùng mới trong bảng users (user.entity.ts) với vai trò mặc định là "Customer".  5. Trả về một phản hồi thành công (ví dụ: HTTP 201 Created) cùng với thông tin người dùng vừa được tạo (hoặc thông báo thành công). |
| **9** |  | Hệ thống thực hiện luồng đăng nhập tự động:  1. Lưu trữ token (JWT) được trả về từ server (nếu có) vào localStorage.  2. Cập nhật AuthContext.tsx để phản ánh trạng thái đã đăng nhập.  3. Chuyển hướng người dùng đến trang chủ (HomePage.tsx). |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **E1: Email đã tồn tại** | - **Trigger:** Tại bước 8 của Luồng chính.  - **Kịch bản:**  1. user-service phát hiện email được cung cấp đã được sử dụng bởi một tài khoản khác.  2. Dịch vụ trả về một lỗi (ví dụ: HTTP 409 Conflict).  3. Hệ thống (frontend) nhận lỗi và hiển thị thông báo: "Địa chỉ email này đã được sử dụng."  4. Người dùng cần nhập một email khác để tiếp tục. |
| **E2: Mật khẩu xác nhận không khớp** | - **Trigger:** Tại bước 6 của Luồng chính.  - **Kịch bản:**  1. Người dùng nhập mật khẩu và mật khẩu xác nhận không giống nhau.  2. Hệ thống (client-side) phát hiện sự không khớp và ngay lập tức hiển thị lỗi bên dưới trường "Xác nhận Mật khẩu", ví dụ: "Mật khẩu không khớp."   3. Yêu cầu không được gửi đến server. |
| **E3: Không đồng ý với Điều khoản Dịch vụ** | - **Trigger:** Tại bước 5 của Luồng chính.  - **Kịch bản:**  1. Người dùng không đánh dấu vào ô "Tôi đồng ý với Điều khoản Dịch vụ".  2. Nút "Đăng ký" bị vô hiệu hóa (disabled).  3. Nếu người dùng vẫn có thể nhấp (do lỗi logic giao diện), hệ thống sẽ hiển thị một thông báo yêu cầu người dùng phải đồng ý với điều khoản. Yêu cầu không được gửi đến server. |
| **E4: Nhập liệu không hợp lệ (Client-side)** | - **Trigger:** Tại bước 6 của Luồng chính.  - **Kịch bản:**  1. Người dùng để trống trường bắt buộc, nhập email sai định dạng, hoặc nhập mật khẩu quá yếu.  2. Hệ thống hiển thị các thông báo lỗi tương ứng ngay bên dưới các trường nhập liệu không hợp lệ. Yêu cầu không được gửi đến server. |
| **A1: Người dùng muốn đọc Điều khoản Dịch vụ** | - **Trigger:** Tại bước 4 của Luồng chính.  - **Kịch bản:**  1. Người dùng nhấp vào liên kết "Điều khoản Dịch vụ".  2. Hệ thống mở trang Điều khoản (TermsPage.tsx) trong một tab mới hoặc một cửa sổ popup để người dùng đọc mà không làm mất dữ liệu đã nhập trên form đăng ký. |
| **A2: Người dùng đã có tài khoản** | - **Trigger:** Tại bất kỳ bước nào trên trang Đăng ký.  - **Kịch bản:**  1. Người dùng nhấp vào liên kết "Đã có tài khoản? Đăng nhập".  2. Hệ thống chuyển hướng người dùng đến trang Đăng nhập (LoginPage.tsx). Use case "Đăng nhập" (UC-01) bắt đầu. |

A diagram of a company

AI-generated content may be incorrect.

# 2.4.3 Xem, tìm kiếm, lọc danh sách sản phẩm

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Xem, tìm kiếm và lọc danh sách sản phẩm** |
| --- | --- |
| **Use Case ID** | UC-04 |
| **Mô tả** | Use case này mô tả cách người dùng (Khách vãng lai hoặc Người dùng đã đăng nhập) duyệt qua danh mục sản phẩm của cửa hàng. Người dùng có thể xem tất cả sản phẩm, hoặc thu hẹp kết quả bằng cách sử dụng bộ lọc (theo danh mục, khoảng giá) và chức năng tìm kiếm theo từ khóa. |
| **Tác nhân** | Khách vãng lai (Guest), Người dùng đã đăng ký (Customer) |
| **Tiền điều kiện** | Tác nhân đã truy cập thành công vào website Cake Shop. |
| **Hậu điều kiện (Thành công)** | 1. Hệ thống hiển thị một danh sách sản phẩm phù hợp với các tiêu chí tìm kiếm và bộ lọc mà người dùng đã chọn.  2. Người dùng có thể thấy các thông tin cơ bản của mỗi sản phẩm (hình ảnh, tên, giá).  3. Hệ thống hiển thị các điều khiển phân trang nếu kết quả trả về có nhiều hơn một trang. |
| **Hậu điều kiện (Thất bại)** | 1. Hệ thống hiển thị thông báo "Không tìm thấy sản phẩm nào".  2. Trong trường hợp lỗi server, hệ thống hiển thị một thông báo lỗi chung. |

**Luồng sự kiện chính (Luồng thành công - Xem tất cả sản phẩm)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Nhấp vào mục "Sản phẩm" hoặc "Cửa hàng" trên thanh điều hướng chính. |  |
| **2** |  | 1. Hệ thống điều hướng người dùng đến trang ProductListPage.tsx.  2. Ngay khi trang được tải, hệ thống gửi một yêu cầu (GET request) đến product-service để lấy trang đầu tiên của tất cả sản phẩm (products.controller.ts).  3. Hiển thị các sản phẩm dưới dạng lưới. Mỗi sản phẩm hiển thị hình ảnh, tên và giá.  4. Hiển thị các tùy chọn lọc (danh mục, giá) và một ô tìm kiếm.  5. Hiển thị thanh điều khiển phân trang (ví dụ: các số trang, nút "Trang trước", "Trang sau"). |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Người dùng lọc sản phẩm theo danh mục** | - **Trigger:** Tại bước 2 của Luồng chính, người dùng nhấp vào một danh mục (ví dụ: "Bánh sinh nhật").  - **Kịch bản:**  1. Hệ thống gửi một yêu cầu mới đến product-service, đính kèm categoryId trong query parameters.  2. Hệ thống nhận về danh sách sản phẩm đã được lọc và cập nhật lại giao diện lưới sản phẩm.  3. Cập nhật lại thanh phân trang cho phù hợp với số lượng sản phẩm mới. |
| **A2: Người dùng sử dụng thanh tìm kiếm** | - **Trigger:** Người dùng nhập một từ khóa (ví dụ: "dâu tây") vào ô tìm kiếm và nhấn Enter.  - **Kịch bản:**  1. Hệ thống gửi yêu cầu đến product-service với từ khóa tìm kiếm (search\_query).  2. Hệ thống nhận về danh sách sản phẩm có tên hoặc mô tả khớp với từ khóa và cập nhật lại giao diện. |
| **A3: Người dùng chuyển trang** | - **Trigger:** Người dùng nhấp vào một số trang hoặc nút "Trang sau" trên thanh phân trang.  - **Kịch bản:**  1. Hệ thống gửi yêu cầu đến product-service để lấy dữ liệu cho trang được yêu cầu, giữ nguyên các bộ lọc và từ khóa tìm kiếm đang hoạt động.  2. Hệ thống cập nhật lại lưới sản phẩm với dữ liệu của trang mới. |
| **E1: Không tìm thấy sản phẩm nào** | - **Trigger:** Yêu cầu đến product-service trả về một danh sách rỗng.  - **Kịch bản:**  1. Hệ thống ẩn đi lưới sản phẩm và thanh phân trang.  2. Hiển thị một thông báo thân thiện: "Không tìm thấy sản phẩm nào phù hợp với lựa chọn của bạn." |
| **E2: Lỗi hệ thống** | - **Trigger:** Yêu cầu đến product-service thất bại (ví dụ: server không phản hồi).  - **Kịch bản:**1. Hệ thống hiển thị một thông báo lỗi chung: "Đã có lỗi xảy ra khi tải sản phẩm. Vui lòng thử lại sau." |

A diagram of a work flow

AI-generated content may be incorrect.

# 2.4.4 Quản lí tài khoản

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Cập nhật thông tin cá nhân** |
| --- | --- |
| **Use Case ID** | UC-12 |
| **Mô tả** | Use case này cho phép người dùng đã đăng nhập tự xem và chỉnh sửa các thông tin cá nhân của mình như tên hiển thị, số điện thoại, hoặc địa chỉ giao hàng mặc định. |
| **Tác nhân** | Khách hàng (Customer) |
| **Tiền điều kiện** | 1. Tác nhân phải đang trong một phiên đăng nhập hợp lệ. |
| **Hậu điều kiện (Thành công)** | 1. Thông tin mới của người dùng được cập nhật thành công vào cơ sở dữ liệu.<  2. Giao diện trang cá nhân hiển thị thông tin đã được cập nhật. |
| **Hậu điều kiện (Thất bại)** | 1. Thông tin của người dùng không thay đổi.  2. Hệ thống hiển thị thông báo lỗi tương ứng. |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| 1 | Điều hướng đến trang "Tài khoản của tôi" hoặc "Thông tin cá nhân". |  |
| 2 |  | 1. Hiển thị trang thông tin cá nhân.<br>2. Gửi yêu cầu đến user-service để lấy và hiển thị thông tin hiện tại của người dùng (Tên, Email, SĐT...). |
| 3 | Nhấn vào nút "Chỉnh sửa". |  |
| 4 |  | Chuyển các trường thông tin (ngoại trừ email) sang trạng thái có thể chỉnh sửa. |
| 5 | Thay đổi thông tin trong các ô nhập liệu (ví dụ: thay đổi Tên hiển thị). |  |
| 6 | Nhấn nút "Lưu thay đổi". |  |
| 7 |  | 1. Gửi dữ liệu đã cập nhật đến user-service.<br>2. user-service xác thực dữ liệu và cập nhật bản ghi người dùng trong cơ sở dữ liệu.<br>3. Hiển thị thông báo thành công (ví dụ: "Cập nhật thông tin thành công!").<br>4. Chuyển các trường về lại trạng thái chỉ đọc với thông tin mới. |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **E1: Dữ liệu nhập không hợp lệ** | - **Trigger:** Người dùng nhập dữ liệu không đúng định dạng (ví dụ: số điện thoại chứa chữ).  - **Kịch bản:**  1. Hệ thống thực hiện xác thực phía client, hiển thị lỗi ngay dưới ô nhập liệu và không cho phép gửi đi. |
| **E2: Lỗi từ máy chủ** | - **Trigger:** user-service gặp lỗi trong quá trình cập nhật.  - **Kịch bản:**  1. Hệ thống hiển thị một thông báo lỗi chung (ví dụ: "Đã có lỗi xảy ra, vui lòng thử lại sau."). |

A diagram of a flowchart

AI-generated content may be incorrect.

# 2.4.5 Quản lí giỏ hàng

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý giỏ hàng** |
| --- | --- |
| **Use Case ID** | UC-07 |
| **Mô tả** | Use case này cho phép người dùng đã đăng nhập có thể thêm sản phẩm vào giỏ hàng, xem lại các sản phẩm trong giỏ, thay đổi số lượng, hoặc xóa sản phẩm khỏi giỏ hàng. Trạng thái giỏ hàng được duy trì giữa các phiên làm việc của người dùng. |
| **Tác nhân** | Người dùng đã đăng ký (Customer) |
| **Tiền điều kiện** | Người dùng đã đăng nhập thành công vào hệ thống. |
| **Hậu điều kiện (Thành công)** | 1. Giỏ hàng của người dùng được cập nhật (thêm/sửa/xóa sản phẩm).  2. Thay đổi được lưu lại trong cart-service và được đồng bộ trên giao diện. |
| **Hậu điều kiện (Thất bại)** | Giỏ hàng không thay đổi. Hệ thống hiển thị thông báo lỗi. |

**Luồng sự kiện chính (Luồng thêm và xem sản phẩm)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trên trang Chi tiết sản phẩm (ProductDetailPage.tsx), người dùng chọn số lượng và nhấp vào nút "Thêm vào giỏ hàng". |  |
| **2** |  | 1. Hệ thống gửi yêu cầu (POST request) đến cart-service để thêm sản phẩm vào giỏ hàng (cart.controller.ts).  2. Hiển thị một thông báo ngắn gọn xác nhận "Đã thêm sản phẩm vào giỏ hàng!".  3. Cập nhật số lượng hiển thị trên icon giỏ hàng ở thanh điều hướng (Navbar). |
| **3** | Nhấp vào icon giỏ hàng trên thanh điều hướng. |  |
| **4** |  | Hệ thống điều hướng người dùng đến trang Giỏ hàng (CartPage.tsx). Giao diện hiển thị danh sách tất cả các sản phẩm trong giỏ, cùng với hình ảnh, tên, đơn giá, số lượng, và thành tiền cho từng sản phẩm. Hiển thị tổng số tiền của giỏ hàng. |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Thay đổi số lượng sản phẩm** | - **Trigger:** Tại bước 4 của Luồng chính, trên trang Giỏ hàng, người dùng thay đổi số lượng của một sản phẩm (tăng hoặc giảm).  - **Kịch bản:**  1. Hệ thống gửi yêu cầu (PUT request) đến cart-service để cập nhật số lượng (update-item.dto.ts).  2. cart-service cập nhật dữ liệu.  3. Giao diện tự động tính toán lại thành tiền cho sản phẩm đó và tổng tiền của toàn bộ giỏ hàng. |
| **A2: Xóa sản phẩm khỏi giỏ hàng** | - **Trigger:** Tại bước 4, người dùng nhấp vào nút "Xóa" bên cạnh một sản phẩm.  - **Kịch bản:**  1. Hệ thống gửi yêu cầu (DELETE request) đến cart-service để xóa sản phẩm.  2. Sản phẩm đó biến mất khỏi danh sách trên giao diện.  3. Tổng tiền của giỏ hàng được tính toán lại. |
| **E1: Thêm sản phẩm đã hết hàng** | - **Trigger:** Tại bước 1, nếu sản phẩm đã hết hàng.  - **Kịch bản:**  1. Nút "Thêm vào giỏ hàng" sẽ bị vô hiệu hóa (disabled).  2. Nếu người dùng vẫn có thể gửi yêu cầu, cart-service sẽ từ chối và trả về lỗi, frontend sẽ hiển thị thông báo "Sản phẩm đã hết hàng." |

A diagram of a diagram

AI-generated content may be incorrect.

# 2.4.6 Đặt hàng

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Đặt hàng (Checkout)** |
| --- | --- |
| **Use Case ID** | UC-08 |
| **Mô tả** | Hướng dẫn người dùng qua các bước cuối cùng để hoàn tất việc mua hàng: xác nhận các mặt hàng, cung cấp địa chỉ giao hàng và chọn phương thức thanh toán để tạo một đơn hàng chính thức trong hệ thống. |
| **Tác nhân** | Người dùng đã đăng ký (Customer) |
| **Tiền điều kiện** | 1. Người dùng đã đăng nhập.  2. Giỏ hàng của người dùng phải có ít nhất một sản phẩm. |
| **Hậu điều kiện (Thành công)** | 1. Một đơn hàng mới được tạo trong order-service với trạng thái "Pending" (Chờ xử lý).  2. Giỏ hàng của người dùng được xóa sạch.  3. Người dùng được chuyển đến bước tiếp theo (trang thanh toán hoặc trang xác nhận đơn hàng thành công). |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trên trang Giỏ hàng, người dùng nhấp vào nút "Tiến hành thanh toán". |  |
| **2** |  | Hệ thống chuyển hướng người dùng đến trang Thanh toán (CheckoutPage.tsx). |
| **3** |  | Trang thanh toán hiển thị lại tóm tắt đơn hàng và các trường thông tin giao hàng (tên, địa chỉ, số điện thoại) được điền sẵn từ hồ sơ người dùng. |
| **4** | Người dùng kiểm tra lại thông tin, chỉnh sửa nếu cần, và chọn một phương thức thanh toán (ví dụ: "Thanh toán khi nhận hàng (COD)" hoặc "Thanh toán bằng VNPay"). |  |
| **5** | Nhấp vào nút "Đặt hàng". |  |
| **6** |  | 1. Hệ thống gửi yêu cầu tạo đơn hàng (POST request) đến order-service (create-order.dto.ts). Yêu cầu này chứa thông tin người nhận và danh sách sản phẩm.  2. order-service nhận yêu cầu, tạo một đơn hàng mới trong cơ sở dữ liệu với trạng thái là PENDING, và trả về thông tin đơn hàng vừa tạo (bao gồm orderId).  3. order-service cũng gửi yêu cầu xóa giỏ hàng đến cart-service. |
| **7** |  | Dựa trên phương thức thanh toán đã chọn ở bước 4:  - Nếu là "COD", hệ thống chuyển hướng người dùng đến trang "Đặt hàng thành công".  - Nếu là "VNPay", hệ thống bắt đầu **Use Case 9: Thanh toán trực tuyến qua VNPay**. |

A diagram of a diagram

AI-generated content may be incorrect.

# 2.4.7 Thanh toán qua VNPay

| **Tên Use Case** | **Thanh toán đơn hàng qua VNPay** |
| --- | --- |
| **Use Case ID** | UC-09 |
| **Mô tả** | Use case này mô tả toàn bộ quá trình từ khi người dùng xác nhận đơn hàng tại trang thanh toán, được chuyển hướng đến cổng thanh toán VNPay, thực hiện giao dịch, và nhận được xác nhận cuối cùng từ hệ thống một cách bất đồng bộ. |
| **Tác nhân** | Khách hàng (Customer) |
| **Tiền điều kiện** | 1. Tác nhân đã đăng nhập.  2. Có ít nhất một sản phẩm trong giỏ hàng.  3. Tác nhân đang ở trang Thanh toán (Checkout) và đã điền đầy đủ thông tin giao hàng.  4. Một đơn hàng đã được tạo trong hệ thống với trạng thái **PENDING**. |
| **Hậu điều kiện (Thành công)** | 1. Thanh toán được xác nhận thành công bởi VNPay.  2. Trạng thái đơn hàng trong hệ thống được cập nhật thành **PROCESSING** (hoặc COMPLETED).  3. Giỏ hàng của người dùng được xóa rỗng.  4. Người dùng nhận được email xác nhận đơn hàng thành công. |
| **Hậu điều kiện (Thất bại)** | 1. Thanh toán thất bại hoặc bị hủy.  2. Trạng thái đơn hàng được cập nhật thành **FAILED** hoặc **CANCELLED**.  3. Người dùng được thông báo về lỗi thanh toán trên giao diện. |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| 1 | Tại trang Thanh toán, chọn "VNPay" làm phương thức thanh toán và nhấn nút **"Hoàn tất đơn hàng"**. |  |
| 2 |  | 1. **Frontend** gửi yêu cầu tạo thanh toán đến payment-service với ID của đơn hàng vừa tạo.  2. **payment-service** gọi API của VNPay để tạo một URL thanh toán duy nhất, sau đó trả URL này về cho Frontend. |
| 3 |  | **Frontend** tự động điều hướng trình duyệt của người dùng đến URL của cổng thanh toán VNPay. |
| 4 | Nhập thông tin thẻ/tài khoản và hoàn tất các bước xác thực bảo mật (OTP) trên giao diện của VNPay. |  |
| 5 |  | 1. **(Phía Client)** VNPay điều hướng trình duyệt người dùng trở lại trang VnPayReturnPage.tsx của cửa hàng, kèm theo các tham số kết quả sơ bộ (ví dụ: vnp\_ResponseCode=00).  2. **(Phía Server)** **Đồng thời**, máy chủ của VNPay gửi một yêu cầu **IPN (Instant Payment Notification)** đến webhook của hệ thống (/api/payment/vnpay\_ipn). **Đây là luồng xác nhận cuối cùng.** |
| 6 |  | 1. **(Trên VnPayReturnPage.tsx)** Giao diện hiển thị thông báo "Thanh toán thành công!" dựa trên mã trả về sơ bộ. Chức năng clearCart() được gọi để xóa giỏ hàng.  2. **(Trên Backend)** payment-service nhận được IPN, xác thực chữ ký vnp\_SecureHash, và xác nhận giao dịch thành công.  3. payment-service **phát một sự kiện** (event) payment\_processed vào **Message Broker (RabbitMQ)**.  4. order-service nhận được sự kiện và cập nhật trạng thái đơn hàng thành PROCESSING.  5. notification-service nhận được sự kiện và gửi email xác nhận cho người dùng. |
| 7 | Nhấp vào liên kết để xem chi tiết đơn hàng hoặc quay về trang chủ. |  |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Người dùng hủy giao dịch** | - **Trigger:** Tại Bước 4, người dùng nhấn nút "Hủy" trên cổng thanh toán VNPay.  - **Kịch bản:**  1. VNPay điều hướng người dùng về trang VnPayReturnPage với mã lỗi tương ứng.  2. Giao diện hiển thị thông báo "Giao dịch đã bị hủy." Đơn hàng có thể được cập nhật trạng thái thành CANCELLED. |
| **E1: Thanh toán thất bại (từ VNPay)** | - **Trigger:** Tại Bước 4, thanh toán thất bại do sai thông tin thẻ, không đủ số dư, v.v.  - **Kịch bản:**  1. VNPay điều hướng người dùng về VnPayReturnPage với mã lỗi (vnp\_ResponseCode khác '00').  2. Giao diện hiển thị thông báo lỗi, ví dụ: "Thanh toán thất bại: Không đủ số dư."  3. Luồng IPN ở server cũng sẽ nhận được mã lỗi, giúp cập nhật trạng thái đơn hàng thành FAILED. |
| **E2: Mất kết nối trình duyệt sau khi thanh toán** | - **Trigger:** Người dùng hoàn tất thanh toán (Bước 4), nhưng đóng trình duyệt trước khi được chuyển về trang VnPayReturnPage.  - **Kịch bản:**  1. Người dùng không thấy trang "Thanh toán thành công" ngay.  2. Tuy nhiên, luồng IPN ở **Bước 6 vẫn diễn ra trên máy chủ**. Đơn hàng vẫn được xử lý, cập nhật trạng thái và email vẫn được gửi. Khi người dùng truy cập lại trang "Lịch sử đơn hàng", họ sẽ thấy đơn hàng đã ở trạng thái PROCESSING. |
| **E3: Lỗi xác thực IPN** | - **Trigger:** payment-service nhận được một cuộc gọi IPN nhưng chữ ký vnp\_SecureHash không hợp lệ (có thể là một cuộc gọi giả mạo).  - **Kịch bản:**  1. payment-service bỏ qua yêu cầu này và ghi lại nhật ký (log) về một cuộc gọi IPN không hợp lệ. Trạng thái đơn hàng không thay đổi để đảm bảo an toàn. |

A diagram of a company

AI-generated content may be incorrect.

# 2.4.8 Theo dõi đơn hàng

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Xem lịch sử đơn hàng** |
| --- | --- |
| **Use Case ID** | UC-10 |
| **Mô tả** | Use case này cho phép người dùng đã đăng nhập xem lại danh sách tất cả các đơn hàng họ đã đặt trước đây và xem thông tin chi tiết của một đơn hàng cụ thể. |
| **Tác nhân** | Người dùng đã đăng ký (Customer) |
| **Tiền điều kiện** | Người dùng đã đăng nhập vào hệ thống và đã đặt ít nhất một đơn hàng. |
| **Hậu điều kiện (Thành công)** | 1. Hệ thống hiển thị danh sách các đơn hàng đã đặt.  2. Khi được chọn, hệ thống hiển thị thông tin chi tiết của một đơn hàng cụ thể. |
| **Hậu điều kiện (Thất bại)** | Hệ thống hiển thị thông báo "Bạn chưa có đơn hàng nào." |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Nhấp vào menu tài khoản trên thanh điều hướng và chọn "Lịch sử đơn hàng". |  |
| **2** |  | Hệ thống điều hướng người dùng đến trang Lịch sử đơn hàng (OrderHistoryPage.tsx). |
| **3** |  | 1. Hệ thống gửi yêu cầu (GET request) đến order-service để lấy danh sách tất cả đơn hàng của người dùng hiện tại.  2. Hiển thị một danh sách các đơn hàng, mỗi đơn hàng hiển thị các thông tin tóm tắt như: Mã đơn hàng, Ngày đặt, Tổng tiền, và Trạng thái hiện tại. Trạng thái được hiển thị với màu sắc tươngứng(thamchiếu: AdminOrderListPage.tsx, getStatusColor). |
| **4** | Nhấp vào nút "Xem chi tiết" trên một đơn hàng cụ thể. |  |
| **5** |  | 1. Hệ thống lấy ID của đơn hàng được chọn.  2. Điều hướng người dùng đến trang Chi tiết đơn hàng (OrderDetailPage.tsx).  3. Hiển thị tất cả thông tin chi tiết của đơn hàng đó: thông tin người nhận, danh sách sản phẩm đã mua, và lịch sử các trạng thái của đơn hàng. |

A diagram of a activity

AI-generated content may be incorrect.

# 2.4.9 Gửi đánh giá sản phẩm

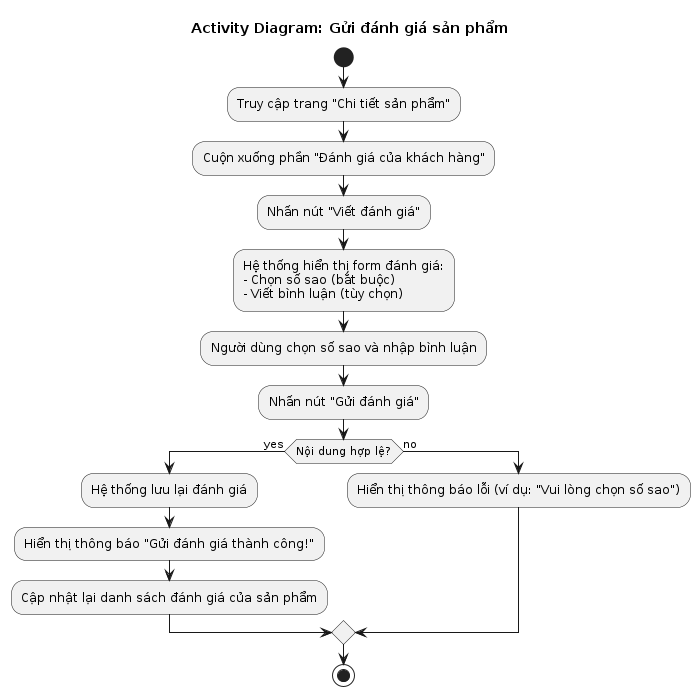
A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Gửi đánh giá sản phẩm** |
| --- | --- |
| **Use Case ID** | UC-11 |
| **Mô tả** | Cho phép người dùng để lại phản hồi (đánh giá sao và bình luận) cho một sản phẩm mà họ đã mua sau khi đơn hàng được hoàn tất. |
| **Tác nhân** | Người dùng đã đăng ký (Customer) |
| **Tiền điều kiện** | 1. Người dùng đã đăng nhập.  2. Người dùng đã mua sản phẩm và đơn hàng chứa sản phẩm đó đã ở trạng thái "Hoàn thành" (COMPLETED). |
| **Hậu điều kiện (Thành công)** | 1. Đánh giá của người dùng được lưu vào hệ thống (review-service).  2. Đánh giá sẽ hiển thị trên trang chi tiết của sản phẩm đó sau khi được quản trị viên duyệt (nếu có). |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Truy cập vào trang Chi tiết sản phẩm (ProductDetailPage.tsx) của một sản phẩm đã mua. |  |
| **2** | Cuộn đến phần "Đánh giá sản phẩm". |  |
| **3** | Chọn số sao đánh giá và nhập nội dung bình luận vào hộp văn bản. |  |
| **4** | Nhấp nút "Gửi đánh giá". |  |
| **5** |  | 1. Hệ thống gửi yêu cầu (POST request) đến reviewApi.ts để lưu đánh giá.  2. Hiển thị thông báo "Cảm ơn bạn đã gửi đánh giá!". |



# 2.4.10 Thiết kế bánh tùy chỉnh

A diagram of a diagram

AI-generated content may be incorrect.

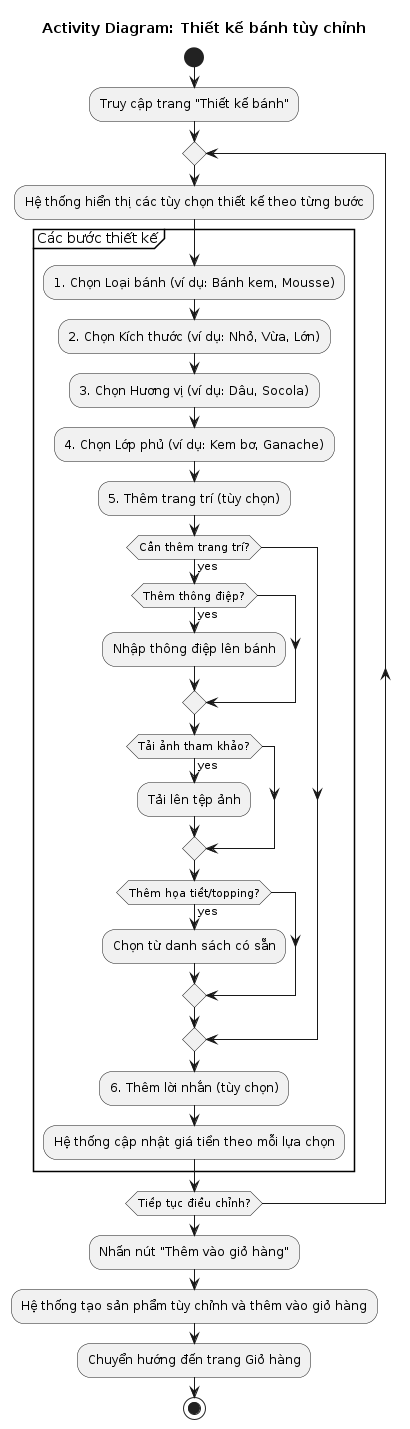
| **Tên Use Case** | **Thiết kế bánh tùy chỉnh** |
| --- | --- |
| **Use Case ID** | UC-14 |
| **Mô tả** | Use case này cho phép người dùng tự xây dựng một chiếc bánh kem theo yêu cầu riêng, từ các thành phần cơ bản đến trang trí chi tiết. Sau khi hoàn tất thiết kế, sản phẩm tùy chỉnh này sẽ được thêm vào giỏ hàng với một mức giá được tính toán tự động. |
| **Tác nhân** | Khách hàng (Customer) |
| **Tiền điều kiện** | 1. Tác nhân phải đang trong một phiên đăng nhập hợp lệ. |
| **Hậu điều kiện (Thành công)** | 1. Một sản phẩm "Bánh tùy chỉnh" với đầy đủ các thuộc tính người dùng đã chọn được tạo ra.  2. Sản phẩm này được thêm vào giỏ hàng của người dùng. |
| **Hậu điều kiện (Thất bại)** | 1. Sản phẩm không được tạo.  2. Hệ thống hiển thị thông báo lỗi nếu có. |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Người dùng)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| 1 | Nhấp vào mục "Đặt bánh theo yêu cầu" hoặc tương tự trên trang web. |  |
| 2 |  | Hiển thị giao diện thiết kế bánh, thường dưới dạng một biểu mẫu đa bước (multi-step form). |
| 3 | **(Chọn Loại bánh)** Chọn một loại cốt bánh từ danh sách (ví dụ: Bông lan, Mousse, Bánh phô mai). | Cập nhật các tùy chọn và hình ảnh minh họa tương ứng với loại bánh đã chọn. |
| 4 | **(Chọn Kích thước bánh)** Chọn một kích thước từ danh sách (ví dụ: 16cm, 20cm). | Cập nhật lại giá tiền dựa trên kích thước được chọn. |
| 5 | **(Chọn Hương vị)** Chọn một hương vị cho cốt bánh và kem (ví dụ: Vani, Sô cô la, Trà xanh). |  |
| 6 | **(Chọn Lớp phủ)** Chọn loại lớp phủ bên ngoài bánh (ví dụ: Kem bơ, Fondant, Kem tươi). |  |
| 7 | **(Thêm trang trí)** Điều hướng đến bước trang trí. | Hiển thị các tùy chọn trang trí chi tiết (Đây là điểm bắt đầu của các luồng <<extend>>). |
| 8 | **(Thêm lời nhắn)** Nhập lời nhắn để gửi kèm bánh (ví dụ: nội dung ghi trên thiệp). |  |
| 9 | Xem lại toàn bộ cấu hình bánh và tổng giá tiền cuối cùng. Nhấn nút "Thêm vào giỏ hàng". |  |
| 10 |  | 1. Tạo một mục sản phẩm mới trong giỏ hàng, lưu trữ tất cả các tùy chọn người dùng đã chọn.<br>  2. Điều hướng người dùng đến trang Giỏ hàng hoặc hiển thị thông báo "Thêm vào giỏ hàng thành công". |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Trang trí chi tiết (Luồng <<Extend>>)** | - **Trigger:** Tại Bước 7 của Luồng chính, người dùng muốn thêm các chi tiết trang trí cụ thể.  - **Kịch bản:** Hệ thống cung cấp các tùy chọn sau:  1. **Thêm thông điệp trên bánh:** Người dùng nhập một dòng chữ ngắn (ví dụ: "Chúc mừng sinh nhật Mẹ") để viết trực tiếp lên mặt bánh.  2. **Gửi ảnh bánh tham khảo:** Người dùng sử dụng nút "Tải ảnh lên" để chọn một tệp hình ảnh từ máy tính/điện thoại làm mẫu cho thợ làm bánh.  3. **Thêm họa tiết/topping:** Người dùng chọn từ danh sách các loại topping có sẵn (ví dụ: Trái cây tươi, Kẹo dẻo, Mảnh sô cô la) và giá tiền được cập nhật tương ứng. |
| **E1: Tùy chọn không tương thích** | - **Trigger:** Một lựa chọn ở bước trước khiến một tùy chọn ở bước sau không khả dụng (ví dụ: Cốt bánh Mousse không thể đi với lớp phủ Fondant).  - **Kịch bản:**  1. Hệ thống sẽ vô hiệu hóa (làm mờ) tùy chọn không tương thích.  2. Hiển thị một ghi chú giải thích lý do (ví dụ: "Fondant không phù hợp với cốt bánh Mousse"). |
| **E2: Tải lên tệp không hợp lệ** | - **Trigger:** Tại luồng A1, người dùng tải lên một tệp không phải hình ảnh hoặc có dung lượng quá lớn.  - **Kịch bản:**  1. Hệ thống từ chối tệp và hiển thị thông báo lỗi (ví dụ: "Vui lòng chỉ tải lên tệp ảnh JPG, PNG và nhỏ hơn 5MB."). |



# 2.4.11 Quản lí Đơn hàng (Admin)

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý Đơn hàng (Admin)** |
| --- | --- |
| **Use Case ID** | UC-13 |
| **Mô tả** | Use case này mô tả cách Quản trị viên (Admin) xem, tìm kiếm, lọc và quan trọng nhất là cập nhật trạng thái của các đơn hàng trong hệ thống. |
| **Tác nhân** | Quản trị viên (Administrator) |
| **Tiền điều kiện** | Người dùng đã đăng nhập thành công vào hệ thống với vai trò là Quản trị viên. |
| **Hậu điều kiện (Thành công)** | 1. Trạng thái của một đơn hàng được cập nhật thành công trong cơ sở dữ liệu.  2. Giao diện hiển thị danh sách đơn hàng được cập nhật với thông tin mới nhất. |
| **Hậu điều kiện (Thất bại)** | 1. Trạng thái đơn hàng không thay đổi.  2. Hệ thống hiển thị một thông báo lỗi cụ thể cho Admin. |

**Luồng sự kiện chính (Xem và cập nhật trạng thái đơn hàng)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trong giao diện quản trị, Admin nhấp vào mục "Đơn hàng". |  |
| **2** |  | Hệ thống điều hướng đến trang AdminOrderListPage.tsx. |
| **3** |  | 1. Ngay khi trang được tải, hệ thống gọi hàm getAllOrdersForAdmin để gửi yêu cầu (GET request) đến order-service và lấy về tất cả các đơn hàng. |
| **4** |  | Hệ thống hiển thị danh sách các đơn hàng dưới dạng bảng. Mỗi hàng hiển thị thông tin chính: Mã đơn hàng, Tên khách hàng, Ngày đặt, Tổng tiền và Trạng thái. Trạng thái được hiển thị dưới dạng một "badge" có màu sắc riêng biệt (định nghĩa trong getStatusColor). |
| **5** | Admin xác định một đơn hàng cần xử lý (ví dụ, một đơn hàng có trạng thái "Pending"). Admin nhấp vào một menu hành động bên cạnh đơn hàng đó. |  |
| **6** | Admin chọn một trạng thái mới từ menu (ví dụ: "Processing"). |  |
| **7** |  | 1. Hệ thống gửi một yêu cầu cập nhật (PUT/PATCH request) đến order-service (updateOrderAdmin).  2. Yêu cầu chứa ID của đơn hàng và trạng thái mới được chọn. |
| **8** |  | order-service thực hiện logic cập nhật (tham chiếu: orders.service.ts, dòng 440-447):  1. Tìm đơn hàng trong cơ sở dữ liệu.  2. **Kiểm tra quy tắc nghiệp vụ:** Nếu trạng thái cũ là COMPLETED hoặc CANCELLED, hệ thống sẽ từ chối thay đổi.  3. Nếu hợp lệ, hệ thống cập nhật trạng thái mới cho đơn hàng.  4. Lưu thay đổi vào cơ sở dữ liệu và ghi lại log (Admin: Trạng thái đơn hàng... được cập nhật...). |
| **9** |  | 1. order-service trả về phản hồi thành công.  2. Giao diện AdminOrderListPage.tsx tự động làm mới hoặc cập nhật lại hàng tương ứng. Badge trạng thái của đơn hàng đó sẽ đổi sang màu mới (ví dụ: từ màu vàng của "Pending" sang màu xanh dương của "Processing"). |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Tìm kiếm hoặc Lọc đơn hàng** | - **Trigger:** Tại bước 4, Admin sử dụng các biểu tượng FiSearch hoặc FiFilter trên giao diện.  - **Kịch bản:**  1. Admin nhập từ khóa tìm kiếm (mã đơn hàng, tên khách hàng) hoặc chọn các tiêu chí lọc (theo trạng thái, theo ngày).  2. Hệ thống gửi lại yêu cầu đến order-service với các tham số tìm kiếm/lọc.  3. Bảng dữ liệu được cập nhật chỉ với các đơn hàng phù hợp. |
| **A2: Xem chi tiết đơn hàng** | - **Trigger:** Tại bước 5, thay vì thay đổi trạng thái, Admin nhấp vào biểu tượng con mắt (FiEye) hoặc mã đơn hàng.  -**Kịch bản:**  1. Hệ thống điều hướng Admin đến trang Chi tiết đơn hàng (AdminOrderDetailPage.tsx).  2. Trang này hiển thị toàn bộ thông tin về đơn hàng, cho phép Admin có cái nhìn tổng thể trước khi ra quyết định. |
| **E1: Cập nhật trạng thái không hợp lệ** | - **Trigger:** Tại bước 8, nếu Admin cố gắng thay đổi trạng thái của một đơn hàng đã ở trạng thái cuối cùng.  - **Kịch bản:**  1. order-service ném ra một lỗi BadRequestException với thông báo: Đơn hàng [ID] đang ở trạng thái COMPLETED và không thể chuyển sang [trạng thái mới]. (tham chiếu: orders.service.ts, dòng 442).  2. Hệ thống hiển thị một thông báo lỗi tương ứng cho Admin trên giao diện. Trạng thái không bị thay đổi. |
| **E2: Đơn hàng không tồn tại** | - **Trigger:** Yêu cầu cập nhật từ client chứa một orderId không có trong cơ sở dữ liệu.  - **Kịch bản:**  1. order-service trả về lỗi 404 Not Found.  2. Hệ thống hiển thị thông báo "Không tìm thấy đơn hàng này." |

A screenshot of a diagram

AI-generated content may be incorrect.

# 2.4.12 Quản lí Sản phẩm (Admin)

A diagram of a company

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý Sản phẩm** |
| --- | --- |
| **Use Case ID** | UC-14 |
| **Mô tả** | Use case này mô tả cách Quản trị viên thực hiện các thao tác Tạo, Đọc, Cập nhật và Xóa (CRUD) đối với các sản phẩm trong cửa hàng. |
| **Tác nhân** | Quản trị viên (Administrator) |
| **Tiền điều kiện** | 1. Admin đã đăng nhập thành công.  2. Ít nhất phải có một danh mục sản phẩm đã được tạo trong hệ thống. |
| **Hậu điều kiện (Thành công)** | 1. Thao tác (tạo/sửa/xóa) được thực hiện thành công.  2. Danh sách sản phẩm trên giao diện được cập nhật để phản ánh sự thay đổi.  3. Hệ thống hiển thị một thông báo thành công. |
| **Hậu điều kiện (Thất bại)** | 1. Không có thay đổi nào được lưu vào cơ sở dữ liệu.  2. Hệ thống hiển thị một thông báo lỗi cụ thể. |

**Luồng sự kiện chính (Luồng tạo sản phẩm mới)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trong giao diện quản trị, Admin nhấp vào mục "Sản phẩm". |  |
| **2** |  | Hệ thống điều hướng đến trang AdminProductListPage.tsx và hiển thị danh sách các sản phẩm hiện có. |
| **3** | Admin nhấp vào nút "Thêm sản phẩm mới". |  |
| **4** |  | Hệ thống điều hướng đến trang Form Sản phẩm (AdminProductFormPage.tsx). |
| **5** | Admin điền đầy đủ thông tin vào các trường: Tên sản phẩm, Mô tả, Giá, Số lượng tồn kho, chọn một Danh mục từ danh sách thả xuống, và tải lên một hoặc nhiều hình ảnh. |  |
| **6** | Admin nhấp vào nút "Lưu sản phẩm". |  |
| **7** |  | 1. Hệ thống thực hiện kiểm tra dữ liệu phía client.  2. Nếu hợp lệ, hệ thống gửi yêu cầu tạo sản phẩm (POSTrequest) đến product-service (products.controller.ts), bao gồm cả dữ liệu hình ảnh. |
| **8** |  | 1. product-service nhận yêu cầu, xử lý việc tải lên và lưu trữ hình ảnh, sau đó tạo một bản ghi sản phẩm mới trong cơ sở dữ liệu (product.entity.ts).  2. Dịch vụ trả về phản hồi thành công. |
| **9** |  | 1. Hệ thống điều hướng Admin trở lại trang danhsáchsảnphẩm (AdminProductListPage.tsx).  2. Hiển thị một thông báo "Đã tạo sản phẩm thành công!". Sản phẩm mới xuất hiện ở đầu danh sách. |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Chỉnh sửa một sản phẩm hiện có** | - **Trigger:** Tại bước 2, Admin nhấp vào nút "Sửa" trên một sản phẩm.  - **Kịch bản:**  1. Hệ thống điều hướng đến trang AdminProductFormPage.tsx, nhưng các trường đã được điền sẵn với thông tin của sản phẩm đó.  2. Admin thay đổi thông tin và nhấp "Lưu thay đổi".  3. Hệ thống gửi yêu cầu cập nhật (PUT/PATCH) đến product-service.  4. Hệ thống quay về trang danh sách và hiển thị thông báo "Cập nhật thành công!". |
| **A2: Xóa một sản phẩm** | - **Trigger:** Tại bước 2, Admin nhấp vào nút "Xóa" trên một sản phẩm.  - **Kịch bản:**  1. Hệ thống hiển thị một hộp thoại xác nhận: "Bạn có chắc chắn muốn xóa sản phẩm này không?".  2. Admin nhấp "Xác nhận".  3. Hệ thống gửi yêu cầu (DELETE request) đến product-service.  4. Sản phẩm biến mất khỏi danh sách trên giao diện. |
| **E1: Lỗi xác thực dữ liệu** | - **Trigger:** Tại bước 5, Admin nhập dữ liệu không hợp lệ (ví dụ: để trống tên, giá không phải là số).  - **Kịch bản:** Khi Admin nhấp "Lưu", hệ thống sẽ hiển thị các thông báo lỗi ngay bên dưới các trường không hợp lệ. Yêu cầu sẽ không được gửi đến server. |

A diagram of a flowchart

AI-generated content may be incorrect.

# 2.4.12 Quản lí Danh mục Sản phẩm (Admin)

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý Danh mục sản phẩm** |
| --- | --- |
| **Use Case ID** | UC-15 |
| **Mô tả** | Cho phép Admin tạo, sửa và xóa các danh mục để phân loại sản phẩm (ví dụ: Bánh sinh nhật, Bánh mì, Phụ kiện). |
| **Tác nhân** | Quản trị viên (Administrator) |
| **Tiền điều kiện** | Admin đã đăng nhập thành công. |
| **Hậu điều kiện (Thành công)** | Thao tác trên danh mục được thực hiện thành công và giao diện được cập nhật. |
| **Hậu điều kiện (Thất bại)** | Không có thay đổi nào được lưu và hệ thống hiển thị lỗi. |

**Luồng sự kiện chính (Luồng tạo danh mục mới)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trong giao diện quản trị, Admin nhấp vào mục "Danh mục". |  |
| **2** |  | Hệ thống điều hướng đến trang AdminCategoryListPage.tsx và hiển thị các danh mục hiện có. |
| **3** | Admin nhấp "Thêm danh mục mới". |  |
| **4** |  | Hệ thống điều hướng đến trang AdminCategoryFormPage.tsx. |
| **5** | Admin nhập Tên danh mục, mô tả và có thể tải lên một hình ảnh đại diện. |  |
| **6** | Admin nhấp "Lưu danh mục". |  |
| **7** |  | Hệ thống gửi yêu cầu (POST request) đến product-service (categories.controller.ts). |
| **8** |  | Hệ thống quay về trang danh sách và hiển thị thông báo "Đã tạo danh mục thành công!". |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Chỉnh sửa/Xóa danh mục** | - **Mô tả:** Tương tự như luồng Chỉnh sửa/Xóa của sản phẩm. |
| **E1: Xóa một danh mục vẫn còn sản phẩm** | - **Trigger:** Tại luồng xóa, Admin cố gắng xóa một danh mục đang được gán cho một hoặc nhiều sản phẩm.  - **Kịch bản:**  1. Yêu cầu DELETE được gửi đến categories.service.ts.  2. Service kiểm tra xem có sản phẩm nào thuộc danh mục này không.  3. Vì có, service sẽ từ chối yêu cầu và trả về lỗi (ví dụ: 409 Conflict).  4. Hệ thống hiển thị một thông báo lỗi cho Admin: "Không thể xóa danh mục này vì vẫn còn sản phẩm thuộc về nó." |

A screenshot of a chat

AI-generated content may be incorrect.

# 2.4.13 Quản lí Người dùng (Admin)

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý Người dùng** |
| --- | --- |
| **Use Case ID** | UC-16 |
| **Mô tả** | Use case này cho phép Quản trị viên xem danh sách tất cả người dùng đã đăng ký trong hệ thống, tìm kiếm và thực hiện các hành động quản lý như thay đổi vai trò (ví dụ: nâng cấp một người dùng thành Quản trị viên khác) hoặc khóa tài khoản. |
| **Tác nhân** | Quản trị viên (Administrator) |
| **Tiền điều kiện** | Người dùng đã đăng nhập thành công với vai trò Quản trị viên. |
| **Hậu điều kiện (Thành công)** | 1. Thao tác quản lý (ví dụ: thay đổi vai trò) được cập nhật thành công trong cơ sở dữ liệu của user-service.  2. Giao diện bảng người dùng phản ánh sự thay đổi.  3. Hệ thống hiển thị thông báo thành công. |
| **Hậu điều kiện (Thất bại)** | 1. Thông tin người dùng không bị thay đổi.  2. Hệ thống hiển thị một thông báo lỗi cụ thể cho Admin. |

**Luồng sự kiện chính (Xem và thay đổi vai trò người dùng)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trong giao diện quản trị, Admin nhấp vào mục "Người dùng". |  |
| **2** |  | Hệ thống điều hướng Admin đến trang AdminUserListPage.tsx. |
| **3** |  | 1. Ngay khi trang được tải, hệ thống gửi một yêu cầu (GET request) đến user-service để lấy danh sách tất cả người dùng (users.controller.ts). |
| **4** |  | Hệ thống hiển thị danh sách người dùng dưới dạng bảng. Mỗi hàng bao gồm các thông tin như: Tên, Email, Ngày tham gia, và Vai trò hiện tại (Customer/Admin). |
| **5** | Admin tìm đến người dùng cần thay đổi vai trò và nhấp vào nút "Hành động" hoặc "Sửa" bên cạnh người dùng đó. |  |
| **6** | Admin chọn tùy chọn "Thay đổi vai trò". |  |
| **7** |  | Hệ thống hiển thị một hộp thoại hoặc menu cho phép chọn vai trò mới (ví dụ: một dropdown với "Customer" và "Admin"). |
| **8** | Admin chọn vai trò mới và nhấp "Lưu" hoặc "Xác nhận". |  |
| **9** |  | 1. Hệ thống gửi yêu cầu (PUT/PATCH) đến user-service để cập nhật thông tin người dùng, mang theo ID của người dùng và vai trò mới. |
| **10** |  | 1. user-service nhận yêu cầu, xác thực quyền của Admin, tìm người dùng tương ứng và cập nhật trường role trong cơ sở dữ liệu.  2. Dịch vụ trả về phản hồi thành công. |
| **11** |  | 1. Cột "Vai trò" của người dùng đó trên bảng được cập nhật.  2. Hệ thống hiển thị thông báo "Đã cập nhật vai trò người dùng thành công!". |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Tìm kiếm người dùng** | - **Trigger:** Tại bước 4, Admin sử dụng thanh tìm kiếm để nhập tên hoặc email của người dùng.  - **Kịch bản:** Bảng dữ liệu sẽ được lọc và chỉ hiển thị những người dùng khớp với tiêu chí tìm kiếm. |
| **E1: Admin tự hạ cấp vai trò của chính mình** | - **Trigger:** Tại bước 8, Admin cố gắng thay đổi vai trò của chính tài khoản của mình từ "Admin" thành "Customer". - **Kịch bản:**  1. Yêu cầu được gửi đến user-service.  2. user-service sẽ có một logic kiểm tra, nếu ID người dùng trong yêu cầu trùng với ID của Admin đang thực hiện hành động, nó sẽ từ chối thay đổi để ngăn chặn việc hệ thống không còn quản trị viên nào.  3. Hệ thống trả về lỗi và hiển thị thông báo: "Bạn không thể thay đổi vai trò của chính mình." |
| **E2: Vô hiệu hóa tài khoản người dùng** | - **Trigger:** Tại bước 6, Admin chọn tùy chọn "Vô hiệu hóa" hoặc "Khóa tài khoản".  - **Kịch bản:**  1. Hệ thống hiển thị hộp thoại xác nhận.  2. Admin xác nhận, hệ thống gửi yêu cầu đến user-service.  3. user-service cập nhật một trường isActive của người dùng thành false thay vì xóa hẳn. Người dùng bị vô hiệu hóa sẽ không thể đăng nhập được nữa. |

A diagram of a diagram

AI-generated content may be incorrect.

# 2.4.14 Quản lí Đánh giá (Admin)

A diagram of a company

AI-generated content may be incorrect.

| **Tên Use Case** | **Quản lý Đánh giá (Admin)** |
| --- | --- |
| **Use Case ID** | UC-17 |
| **Mô tả** | Use case này cho phép Quản trị viên xem, duyệt và xóa các đánh giá (reviews) do khách hàng gửi. Điều này đảm bảo rằng chỉ những phản hồi phù hợp, mang tính xây dựng mới được hiển thị công khai trên các trang chi tiết sản phẩm, duy trì chất lượng nội dung của website. |
| **Tác nhân** | Quản trị viên (Administrator) |
| **Tiền điều kiện** | 1. Admin đã đăng nhập thành công.  2. Có ít nhất một đánh giá đã được người dùng gửi lên hệ thống. |
| **Hậu điều kiện (Thành công)** | 1. Trạng thái của đánh giá được cập nhật (ví dụ: được chấp thuận) hoặc đánh giá bị xóa khỏi cơ sở dữ liệu.  2. Thay đổi được phản ánh trên trang chi tiết sản phẩm tương ứng.  3. Hệ thống hiển thị thông báo thành công. |
| **Hậu điều kiện (Thất bại)** | 1. Trạng thái của đánh giá không thay đổi.  2. Hệ thống hiển thị một thông báo lỗi cụ thể. |

**Luồng sự kiện chính (Xem và xóa đánh giá)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| **1** | Trong giao diện quản trị, Admin nhấp vào mục "Đánh giá". |  |
| **2** |  | Hệ thống điều hướng Admin đến trang AdminReviewListPage.tsx. |
| **3** |  | 1. Ngay khi trang được tải, hệ thống gửi một yêu cầu (GET request) qua reviewApi.ts để lấy danh sách tất cả các đánh giá. |
| **4** |  | Hệ thống hiển thị danh sách các đánh giá dưới dạng bảng. Mỗi hàng bao gồm: Tên sản phẩm, Tên người gửi, Số sao đánh giá, Nội dung bình luận và Ngày gửi. |
| **5** | Admin xác định một đánh giá không phù hợp hoặc chứa spam cần phải xóa. |  |
| **6** | Admin nhấp vào nút "Xóa" bên cạnh đánh giá đó. |  |
| **7** |  | Hệ thống hiển thị một hộp thoại xác nhận: "Bạn có chắc chắn muốn xóa đánh giá này không? Hành động này không thể hoàn tác." |
| **8** | Admin nhấp vào nút "Xác nhận xóa". |  |
| **9** |  | 1. Hệ thống gửi yêu cầu (DELETE request) qua reviewApi.ts, mang theo ID của đánh giá cần xóa. |
| **10** |  | 1. Service tương ứng nhận yêu cầu và xóa bản ghi đánh giá khỏi cơ sở dữ liệu.  2. Dịch vụ trả về phản hồi thành công. |
| **11** |  | 1. Hàng chứa đánh giá đó biến mất khỏi bảng trên giao diện.  2. Hệ thống hiển thị thông báo "Đã xóa đánh giá thành công!". |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **A1: Phê duyệt một đánh giá (Nếu có)** | - **Ghi chú:** Luồng này áp dụng nếu hệ thống có cơ chế phê duyệt. Nếu không, các đánh giá sẽ mặc định hiển thị.  - **Kịch bản:**  1. Tại bước 4, bảng hiển thị thêm cột "Trạng thái" (ví dụ: "Chờ duyệt", "Đã duyệt").  2. Tại bước 6, thay vì "Xóa", Admin nhấp vào nút "Duyệt".  3. Hệ thống gửi yêu cầu (PUT/PATCH) để thay đổi trạng thái của đánh giá thành "Đã duyệt". Đánh giá này sau đó sẽ xuất hiện trên trang chi tiết sản phẩm. |
| **A2: Lọc đánh giá** | - **Trigger:** Tại bước 4, Admin sử dụng bộ lọc để xem đánh giá cho một sản phẩm cụ thể, hoặc theo số sao đánh giá.  - **Kịch bản:** Bảng dữ liệu được cập nhật để chỉ hiển thị các đánh giá khớp với tiêu chí lọc. |
| **E1: Đánh giá đã bị xóa bởi người khác** | - **Trigger:** Tại bước 9, nếu một Admin khác đã xóa đánh giá này trong lúc Admin hiện tại đang xem xét.  - **Kịch bản:** reviewApi.ts trả về lỗi 404 Not Found. Hệ thống hiển thị thông báo "Đánh giá này không còn tồn tại." và làm mới lại danh sách. |

A diagram of a company

AI-generated content may be incorrect.

# 2.4.15 Thống kê Báo cáo (Admin)

A diagram of a diagram

AI-generated content may be incorrect.

| **Tên Use Case** | **Xem Thống kê và Báo cáo** |
| --- | --- |
| **Use Case ID** | UC-18 |
| **Mô tả** | Use case này cho phép Quản trị viên (Admin) xem các số liệu tổng quan về tình hình kinh doanh của cửa hàng thông qua các chỉ số và biểu đồ trực quan, cũng như xuất các dữ liệu này ra file để lưu trữ hoặc phân tích sâu hơn. |
| **Tác nhân** | Quản trị viên (Admin) |
| **Tiền điều kiện** | 1. Tác nhân phải đang trong một phiên đăng nhập hợp lệ với vai trò Admin. |
| **Hậu điều kiện (Thành công)** | 1. Admin nắm được các chỉ số quan trọng về hoạt động kinh doanh.  2. Admin có thể tải về một tệp báo cáo chứa dữ liệu đã được lọc. |
| **Hậu điều kiện (Thất bại)** | 1. Hệ thống hiển thị lỗi nếu không thể tải dữ liệu.  2. Báo cáo không được tạo ra. |
| **Mối quan hệ** | - <<include>> Tổng hợp và hiển thị chỉ số, biểu đồ (UC-18a)<br>- <<include>> Xuất Báo cáo (UC-18b) |

**Luồng sự kiện chính (Luồng thành công)**

| **Bước** | **Hành động của Tác nhân (Admin)** | **Hành động của Hệ thống** |
| --- | --- | --- |
| 1 | Điều hướng đến trang "Dashboard" hoặc "Báo cáo" trong khu vực quản trị. |  |
| 2 |  | **<<include>> Tổng hợp và hiển thị chỉ số, biểu đồ (UC-18a)**  1. Tự động tổng hợp dữ liệu từ các service (order-service, user-service...) cho một khoảng thời gian mặc định (ví dụ: 30 ngày qua).  2. Hiển thị các chỉ số chính (KPIs) như: Doanh thu, Tổng số đơn hàng, Khách hàng mới.  3. Vẽ các biểu đồ trực quan như: Biểu đồ đường về doanh thu theo ngày, biểu đồ cột cho các sản phẩm bán chạy nhất. |
| 3 | Tùy chọn thay đổi khoảng thời gian hoặc áp dụng các bộ lọc khác (ví dụ: lọc theo trạng thái đơn hàng). | Cập nhật lại toàn bộ các chỉ số và biểu đồ theo bộ lọc mới. |
| 4 | Nhấn vào nút "Xuất Báo cáo" hoặc "Tải về" (<FiDownload />). |  |
| 5 |  | **<<include>> Xuất Báo cáo (UC-18b)**  1. Tạo một tệp báo cáo (ví dụ: CSV, Excel) dựa trên dữ liệu đang được hiển thị với bộ lọc hiện tại.  2. Kích hoạt quá trình tải tệp xuống trên trình duyệt của Admin. |

**Các luồng sự kiện thay thế (Alternative & Exception Flows)**

| **Luồng** | **Mô tả** |
| --- | --- |
| **E1: Không có dữ liệu trong khoảng thời gian đã chọn** | - **Trigger:** Admin chọn một khoảng thời gian mà không có bất kỳ hoạt động kinh doanh nào.  **Kịch bản:**  1. Tại Bước 2 hoặc 3, thay vì hiển thị biểu đồ trống, hệ thống sẽ hiển thị một thông báo thân thiện: "Không có dữ liệu để hiển thị trong khoảng thời gian này." |
| **E2: Lỗi khi tổng hợp dữ liệu** | - **Trigger:** Một trong các microservice (ví dụ: order-service) không phản hồi khi hệ thống yêu cầu dữ liệu thống kê.  - **Kịch bản:**  1. Hệ thống hiển thị một thông báo lỗi chung: "Đã có lỗi xảy ra khi tải dữ liệu thống kê. Vui lòng thử lại." |
| **E3: Lỗi khi xuất báo cáo** | - **Trigger:** Có lỗi xảy ra trong quá trình tạo tệp báo cáo trên máy chủ  - **Kịch bản:**  1. Tại Bước 5, hệ thống sẽ hiển thị một thông báo lỗi: "Không thể tạo tệp báo cáo. Vui lòng thử lại." |

A diagram of a flowchart

AI-generated content may be incorrect.

# 2.5 Sơ đồ Sequence:

Sơ đồ tuần tự (Sequence Diagram) là một công cụ mô hình hóa quan trọng trong phân tích thiết kế hệ thống, mô tả chi tiết sự tương tác giữa các đối tượng trong hệ thống theo thời gian. Sơ đồ này thể hiện rõ ràng thứ tự các thông điệp được gửi và nhận giữa các đối tượng, giúp chúng ta hiểu được luồng xử lý nghiệp vụ một cách trực quan và chi tiết. Việc sử dụng sơ đồ tuần tự giúp làm rõ các tương tác phức tạp giữa các thành phần của hệ thống, đảm bảo tính nhất quán trong thiết kế và hỗ trợ hiệu quả quá trình phát triển phần mềm.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Hình 3.20: Sơ đồ tuần tự quy trình đăng ký tài khoản*

Hình trình bày sequence đăng ký tài khoản với *Customer*, *WebApp*, *Database* và *Mail Server*. Người dùng gửi yêu cầu đăng ký đến WebApp, hệ thống truy vấn Database để kiểm tra trùng SĐT/email và nhận về kết quả. Nhánh [Hợp lệ]: WebApp lưu tài khoản mới vào Database, sau đó gọi Mail Server để gửi email xác nhận; khi nhận phản hồi gửi mail thành công, WebApp trả thông báo đăng ký thành công cho người dùng. Nhánh [Không hợp lệ]: nếu phát hiện trùng SĐT/email, WebApp không ghi dữ liệu và trả về thông báo lỗi để người dùng chỉnh sửa và thử lại. Quy trình thể hiện rõ điểm kiểm tra hợp lệ, luồng tích hợp CSDL và dịch vụ email trong đăng ký

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 3.21: Quy trình đăng nhập*

Hình mô tả sequence đăng nhập với Customer – WebApp – Database: người dùng gửi SĐT và mật khẩu tới WebApp; hệ thống truy vấn Database để kiểm tra thông tin xác thực (tồn tại tài khoản, trạng thái hoạt động, mật khẩu khớp). Kết quả trả về tách hai nhánh alt: nhánh [Đúng] → WebApp thiết lập phiên đăng nhập và chuyển hướng về trang chủ/khu vực admin theo vai trò; nhánh [Sai] → WebApp không thay đổi dữ liệu, trả thông báo lỗi (mật khẩu sai, tài khoản bị khóa/chưa kích hoạt…) để người dùng nhập lại. Quy trình thể hiện rõ điểm kiểm tra trung tâm ở CSDL và phản hồi tương ứng cho từng tình huống.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Hình 3.23: Quy trình đặt hàng COD*

Hình thể hiện sequence đặt hàng COD với *Customer – WebApp – Database – Mail Server*. Khách hàng gửi yêu cầu đặt hàng (COD) tới WebApp; hệ thống ghi đơn vào Database với trạng thái ban đầu “Chờ thanh toán/COD” và nhận kết quả lưu. Từ đây rẽ nhánh alt: [Thành công] → WebApp gọi Mail Server để gửi email xác nhận đơn hàng, nhận kết quả gửi mail, sau đó trả thông báo đặt hàng thành công cho khách; [Thất bại] → nếu quá trình lưu đơn lỗi (hoặc sự cố gửi mail nghiêm trọng), WebApp không xác nhận và thông báo lỗi đặt hàng để người dùng thử lại. Quy trình làm rõ điểm ghi nhận đơn ở CSDL và bước gửi thông báo qua email trong luồng COD.

A diagram of a computer program

AI-generated content may be incorrect.

*Hình 3.24 Quy trình đặt hàng qua VNPay*

Hình thể hiện sequence đặt hàng COD với *Customer – WebApp – Database – Mail Server*. Khách hàng gửi yêu cầu đặt hàng (COD) tới WebApp; hệ thống ghi đơn vào Database với trạng thái ban đầu “Chờ thanh toán/COD” và nhận kết quả lưu. Từ đây rẽ nhánh alt: [Thành công] → WebApp gọi Mail Server để gửi email xác nhận đơn hàng, nhận kết quả gửi mail, sau đó trả thông báo đặt hàng thành công cho khách; [Thất bại] → nếu quá trình lưu đơn lỗi (hoặc sự cố gửi mail nghiêm trọng), WebApp không xác nhận và thông báo lỗi đặt hàng để người dùng thử lại. Quy trình làm rõ điểm ghi nhận đơn ở CSDL và bước gửi thông báo qua email trong luồng COD.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Hình 3.25: Quy trình quản lý đơn hàng của admin*

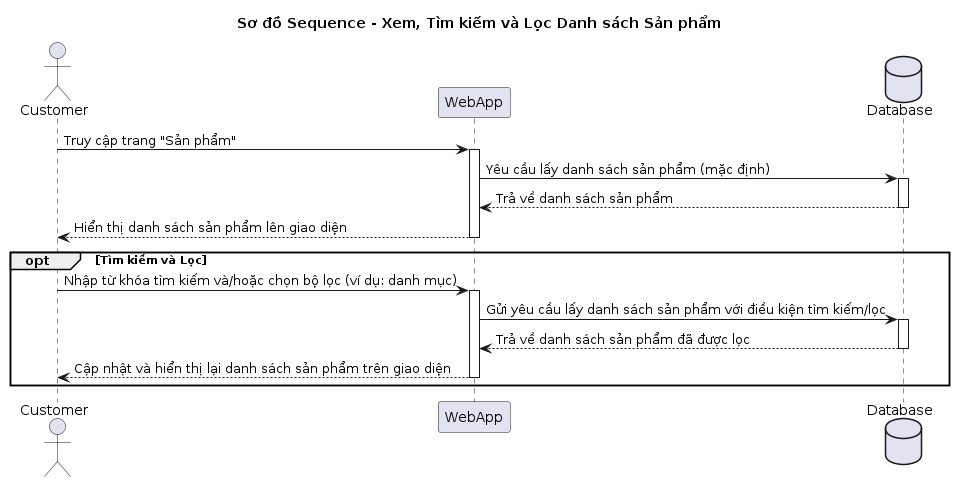
Hình mô tả sequence quản lý đơn hàng của Admin với ba lifeline *Admin – WebApp – Database*. Admin gửi yêu cầu xem danh sách đơn → WebApp truy vấn CSDL và trả về danh sách. Khi Admin xem chi tiết, WebApp tiếp tục truy vấn chi tiết đơn (sản phẩm, khách hàng, thanh toán, lịch sử trạng thái) và trả dữ liệu cho giao diện. Với thao tác cập nhật trạng thái (chờ xử lý/đang giao/đã giao/đã hủy…), WebApp ghi thay đổi vào Database, nhận kết quả cập nhật, sau đó thông báo thành công/thất bại cho Admin. Trình tự thể hiện rõ ba luồng chính: liệt kê, tra cứu chi tiết và cập nhật trạng thái, đảm bảo tính nhất quán dữ liệu suốt quá trình quản trị.

A screenshot of a computer screen

AI-generated content may be incorrect.

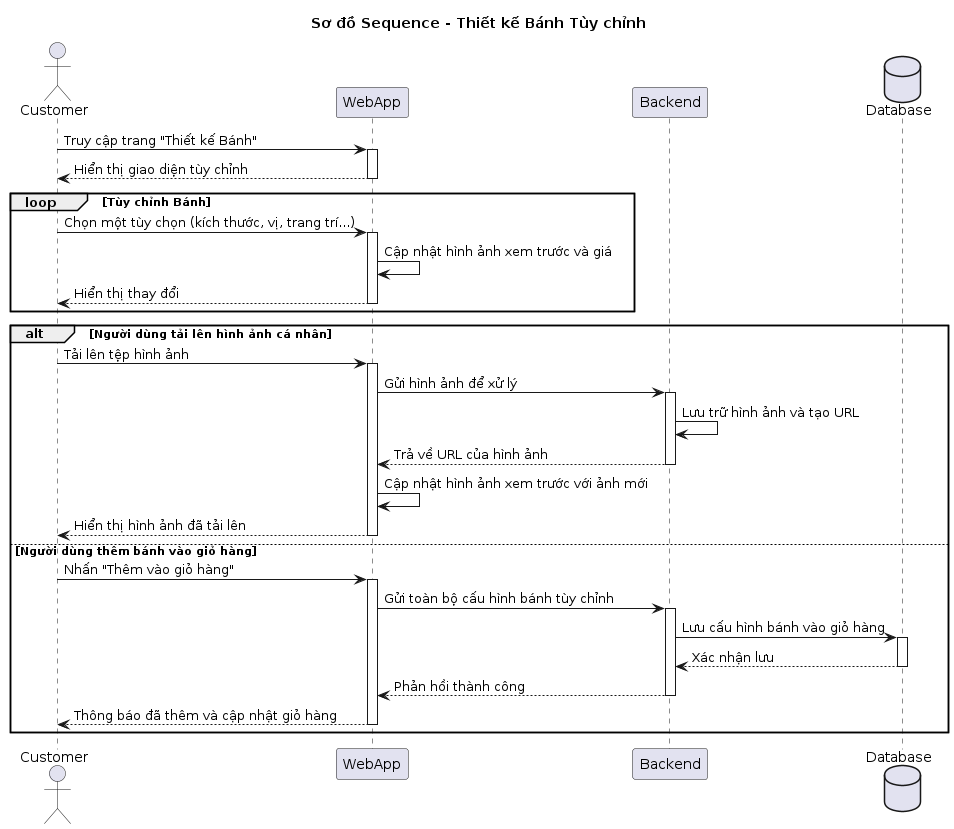
*Hình 3.26: Quy trình cập nhật thông tin cá nhân*

Hình mô tả sequence cập nhật thông tin cá nhân với ba lifeline Customer – WebApp – Database. Người dùng nhập thông tin mới (họ tên, email, địa chỉ…) và gửi lên WebApp. Hệ thống kiểm tra hợp lệ: định dạng email đúng, trường bắt buộc đủ và không trùng email với tài khoản khác bằng cách truy vấn Database. Kết quả tách nhánh alt: [Hợp lệ] → WebApp ghi cập nhật vào Database, nhận kết quả cập nhật và thông báo thành công cho người dùng; [Không hợp lệ] → WebApp không ghi dữ liệu, trả thông báo lỗi (email sai định dạng/trùng, thiếu thông tin…) để người dùng chỉnh sửa và thực hiện lại. Quy trình thể hiện rõ điểm kiểm tra trước khi ghi và phản hồi tương ứng cho từng tình huống.



Hình mô tả sequence diagram cho chức năng xem, tìm kiếm và lọc sản phẩm, bao gồm ba lifeline chính: **Customer**, **WebApp**, và **Database**. Quy trình bắt đầu khi người dùng (Customer) truy cập vào trang sản phẩm. Ngay lập tức, WebApp sẽ gửi yêu cầu đến Database để tải và hiển thị danh sách tất cả sản phẩm hiện có.

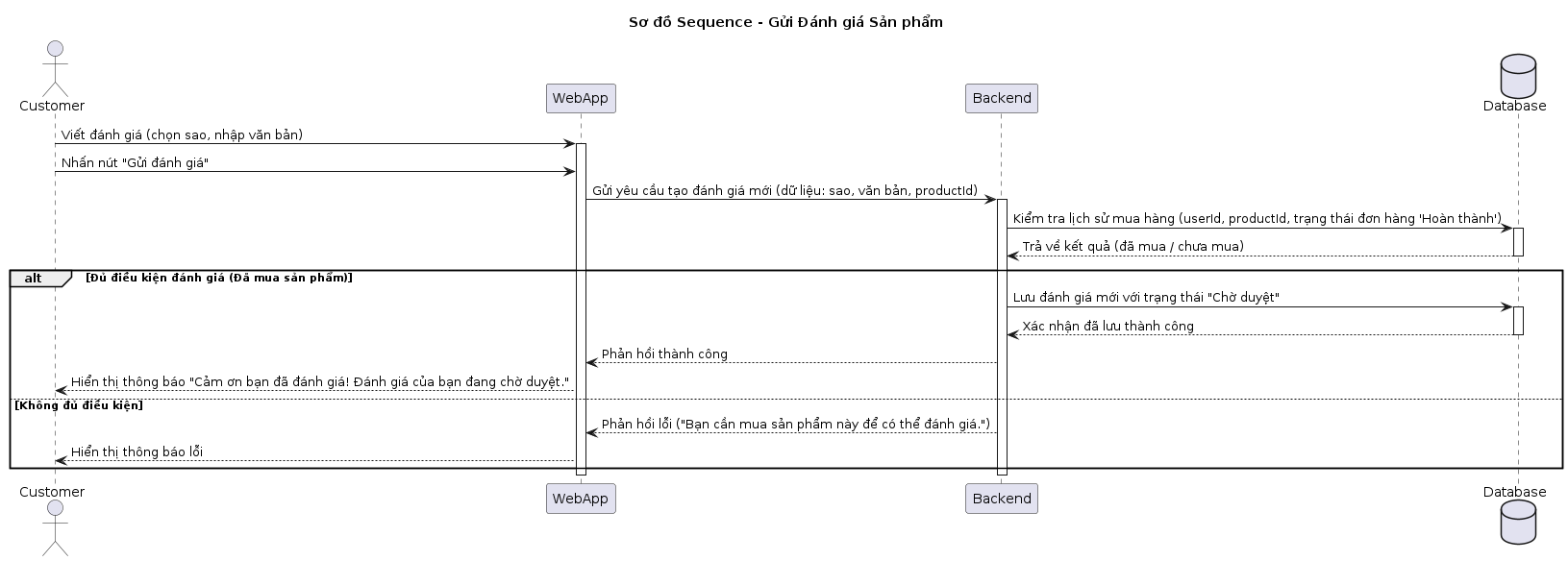
Điểm cốt lõi của quy trình được thể hiện trong khối tùy chọn (**opt**), khi người dùng thực hiện tìm kiếm hoặc lọc. Người dùng nhập từ khóa hoặc chọn các tiêu chí lọc (như danh mục, khoảng giá...). WebApp tiếp nhận các điều kiện này và gửi một truy vấn mới, có tham số, đến Database. Database sẽ xử lý, lọc ra những sản phẩm thỏa mãn điều kiện và trả kết quả về cho WebApp. Cuối cùng, WebApp cập nhật lại giao diện để hiển thị danh sách sản phẩm đã được lọc chính xác theo yêu cầu của người dùng. Quy trình này thể hiện sự tương tác động, giúp người dùng nhanh chóng tìm thấy sản phẩm mong muốn từ một danh sách lớn.



Hình mô tả sequence diagram cho chức năng thiết kế bánh kem tùy chỉnh, một quy trình tương tác cao với bốn lifeline: **Customer**, **WebApp**, **Backend**, và **Database**. Quy trình này cho phép người dùng tự do sáng tạo chiếc bánh của riêng mình thông qua một loạt các lựa chọn trực quan.

Khi người dùng truy cập trang, **WebApp** sẽ hiển thị một giao diện tương tác. Cốt lõi của quy trình nằm trong khối lặp (**loop**), nơi người dùng lần lượt chọn các thuộc tính của bánh như kích thước, hương vị, nhân, và kiểu trang trí. Với mỗi lựa chọn, **WebApp** sẽ tự động cập nhật hình ảnh xem trước và tính toán lại giá cả ngay trên trình duyệt, mang lại phản hồi tức thì cho người dùng mà không cần tải lại trang.

Khối **alt** thể hiện hai hành động chính của người dùng. Trong trường hợp **[Tải lên hình ảnh cá nhân]**, người dùng gửi một tệp ảnh từ thiết bị. **WebApp** chuyển tiếp tệp ảnh này đến **Backend** để xử lý, lưu trữ (ví dụ trên một dịch vụ cloud) và nhận lại một đường dẫn URL. URL này sau đó được dùng để cập nhật hình ảnh xem trước. Trong trường hợp **[Thêm vào giỏ hàng]**, **WebApp** sẽ đóng gói toàn bộ cấu hình của chiếc bánh (bao gồm cả URL hình ảnh nếu có) và gửi đến **Backend**. **Backend** sau đó sẽ lưu thông tin chi tiết của sản phẩm tùy chỉnh này vào **Database** như một mục trong giỏ hàng và gửi xác nhận thành công trở lại, kết thúc bằng một thông báo cho người dùng.



Hình mô tả sequence diagram cho chức năng gửi đánh giá sản phẩm, một quy trình đảm bảo tính xác thực của các phản hồi từ khách hàng. Quy trình bao gồm bốn lifeline: **Customer**, **WebApp**, **Backend**, và **Database**.

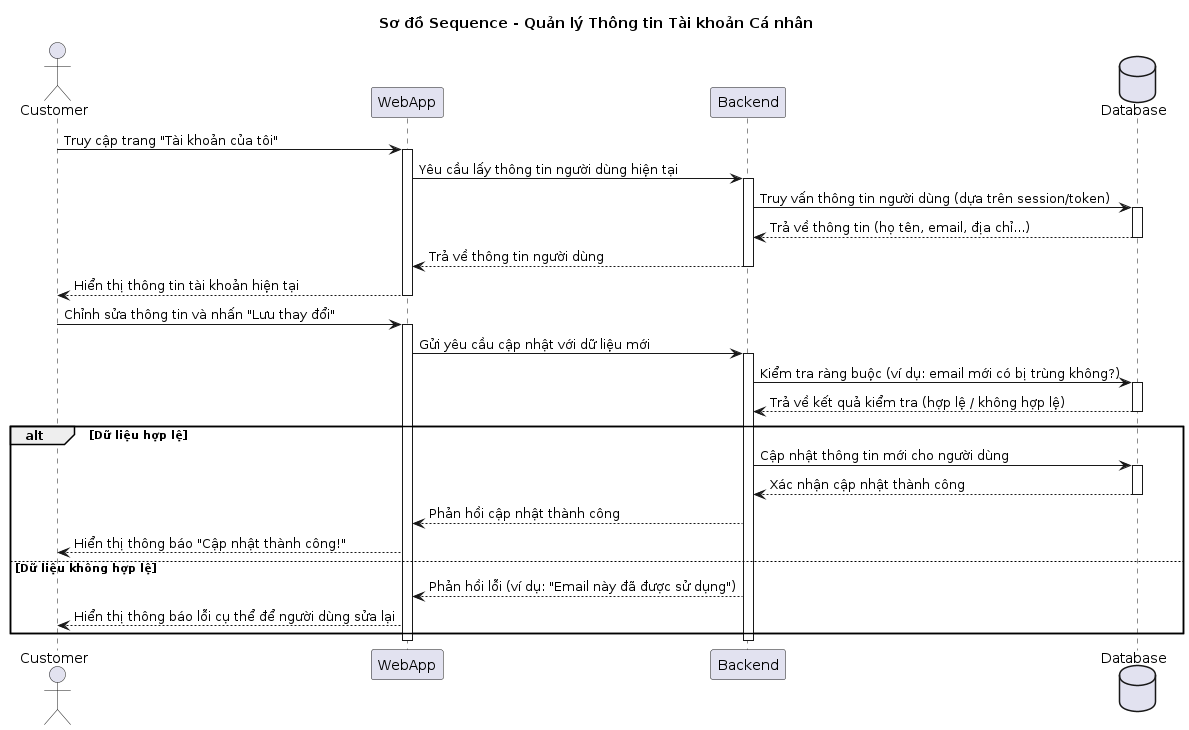
Sau khi người dùng (Customer) soạn thảo nội dung đánh giá (bao gồm xếp hạng sao và bình luận) và nhấn gửi, **WebApp** sẽ chuyển tiếp thông tin này đến **Backend**.

Điểm then chốt của quy trình là bước xác thực quyền đánh giá. Trước khi lưu lại bất kỳ thông tin nào, **Backend** sẽ thực hiện một truy vấn tới **Database** để kiểm tra xem người dùng này có thực sự đã mua sản phẩm đang được đánh giá hay không (dựa trên lịch sử đơn hàng đã hoàn thành).

Quy trình sau đó rẽ nhánh trong khối **alt**:

* **[Đủ điều kiện đánh giá]**: Nếu người dùng đã mua sản phẩm, **Backend** sẽ tiến hành lưu đánh giá vào **Database**, thường với trạng thái mặc định là "Chờ duyệt" để quản trị viên kiểm tra. Sau khi lưu thành công, một thông báo cảm ơn sẽ được hiển thị cho người dùng.
* **[Không đủ điều kiện]**: Nếu hệ thống không tìm thấy lịch sử mua hàng hợp lệ, **Backend** sẽ trả về một thông báo lỗi, giải thích rằng chỉ những người đã mua sản phẩm mới có thể gửi đánh giá.

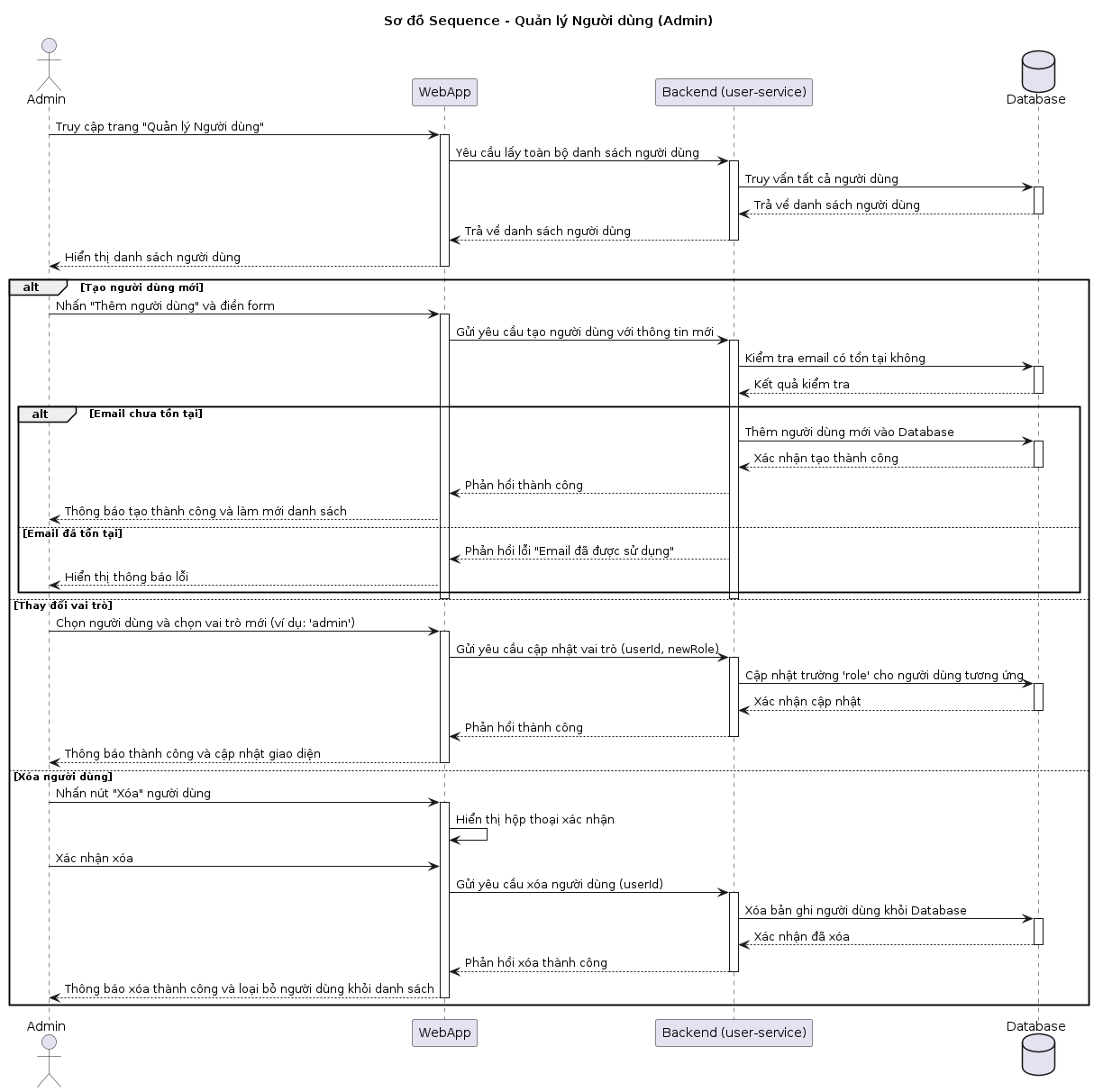
Quy trình này không chỉ thu thập phản hồi của khách hàng mà còn đảm bảo rằng tất cả các đánh giá hiển thị trên trang web đều đến từ những người có trải nghiệm thực tế với sản phẩm, qua đó tăng cường sự tin cậy cho các khách hàng khác.



Hình mô tả sequence diagram cho chức năng quản lý tài khoản cá nhân, thể hiện luồng tương tác giữa bốn lifeline: **Customer**, **WebApp**, **Backend**, và **Database**. Quy trình này cho phép người dùng đã đăng nhập có thể xem và cập nhật thông tin cá nhân của mình.

Đầu tiên, người dùng truy cập vào trang quản lý tài khoản. **WebApp** sẽ gửi một yêu cầu đến **Backend** để lấy thông tin hiện tại của người dùng, **Backend** sau đó truy vấn **Database** và trả dữ liệu về để hiển thị lên giao diện.

Khi người dùng chỉnh sửa và lưu thay đổi, **WebApp** gửi dữ liệu mới đến **Backend**. Tại đây, một bước kiểm tra tính hợp lệ quan trọng sẽ được thực hiện. **Backend** truy vấn **Database** để đảm bảo dữ liệu mới không vi phạm các ràng buộc, ví dụ như kiểm tra địa chỉ email mới có bị trùng với một tài khoản khác đã tồn tại hay không.

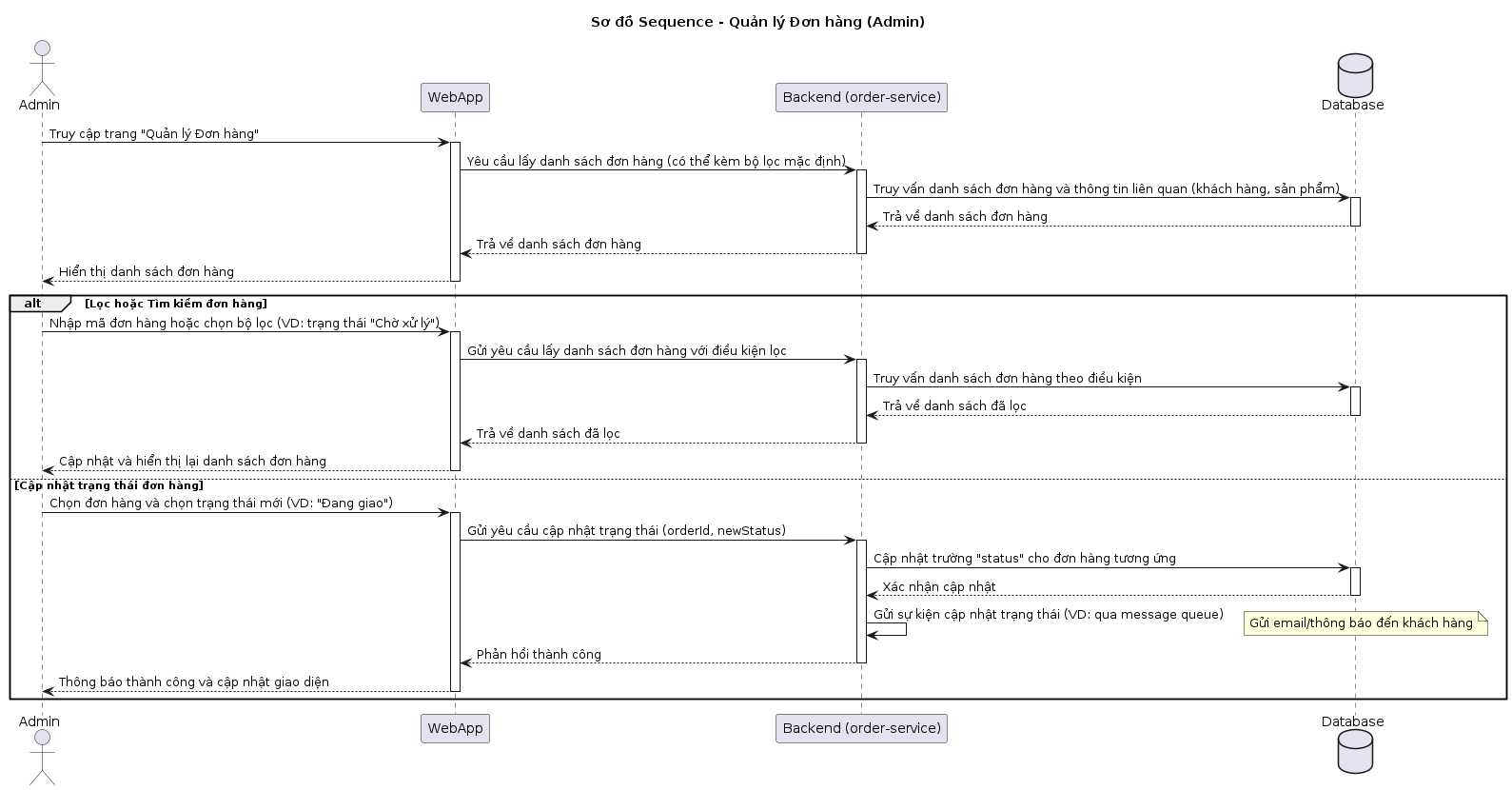


Hình mô tả sequence diagram cho chức năng quản lý người dùng, một công cụ quan trọng dành riêng cho Quản trị viên (Admin). Sơ đồ thể hiện sự tương tác giữa bốn lifeline: **Admin**, **WebApp**, **Backend (user-service)**, và **Database**.

Quy trình bắt đầu khi Admin truy cập trang quản lý, **WebApp** sẽ yêu cầu và hiển thị toàn bộ danh sách người dùng từ **Backend**. Khối **alt** mô tả ba hành động chính mà Admin có thể thực hiện:

1. **[Tạo người dùng mới]**: Admin điền thông tin vào form và gửi đi. **Backend** nhận yêu cầu và thực hiện bước kiểm tra quan trọng là xác thực xem email đã tồn tại trong **Database** hay chưa. Nếu email là duy nhất, người dùng mới sẽ được tạo; ngược lại, hệ thống sẽ báo lỗi.
2. **[Thay đổi vai trò]**: Admin chọn một người dùng và một vai trò mới. **Backend** nhận yêu cầu và trực tiếp cập nhật trường vai trò (role) của người dùng đó trong **Database**, cho phép phân quyền một cách linh hoạt.
3. **[Xóa người dùng]**: Sau khi Admin xác nhận hành động xóa, **Backend** sẽ gửi lệnh để loại bỏ hoàn toàn bản ghi của người dùng đó khỏi **Database**.

Mỗi hành động đều có sự phản hồi rõ ràng từ hệ thống, từ thông báo thành công cho đến các cảnh báo lỗi cụ thể, đảm bảo Admin có thể quản lý cộng đồng người dùng một cách hiệu quả và an toàn.

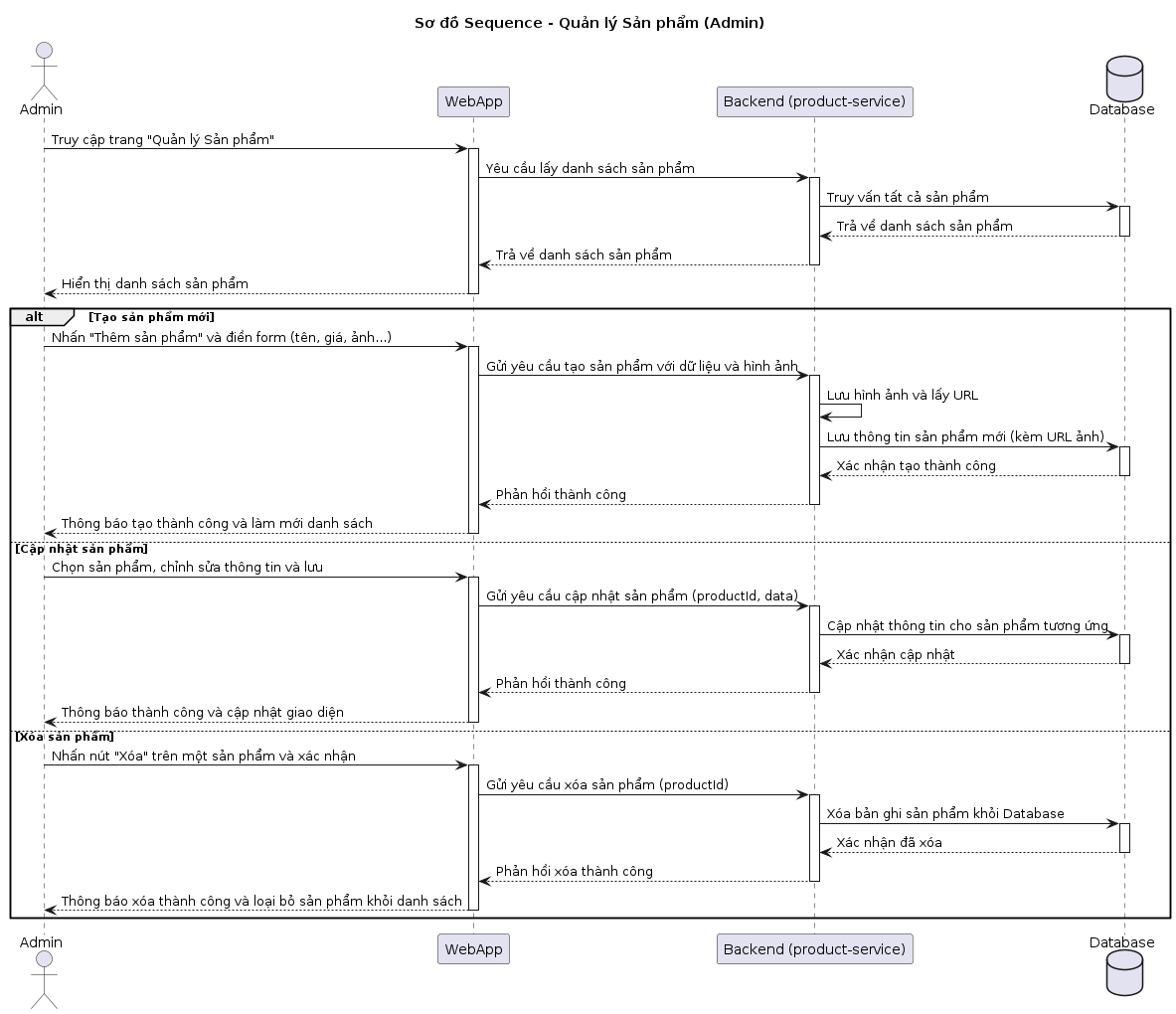


Hình mô tả sequence diagram cho chức năng quản lý đơn hàng, một trong những nhiệm vụ cốt lõi của Quản trị viên (Admin) để vận hành cửa hàng. Quy trình này thể hiện sự tương tác giữa **Admin**, **WebApp**, **Backend (order-service)**, và **Database**.

Khi Admin truy cập trang quản lý, **WebApp** ngay lập tức yêu cầu **Backend** cung cấp danh sách tất cả đơn hàng, sau đó hiển thị chúng một cách chi tiết. Khối **alt** mô tả hai hành động chính mà Admin có thể thực hiện:

1. **[Lọc hoặc Tìm kiếm đơn hàng]**: Để xử lý hiệu quả, Admin có thể tìm kiếm theo mã đơn hàng hoặc sử dụng bộ lọc để xem các đơn hàng theo một trạng thái cụ thể (ví dụ: "Chờ xác nhận", "Đã hủy"...). **WebApp** gửi các điều kiện này đến **Backend** để thực hiện truy vấn và cập nhật lại danh sách hiển thị.
2. **[Cập nhật trạng thái đơn hàng]**: Đây là hành động quan trọng nhất. Khi Admin thay đổi trạng thái của một đơn hàng (ví dụ: từ "Chờ xử lý" sang "Đang giao"), **WebApp** gửi yêu cầu đến **Backend**. **Backend** sẽ cập nhật trạng thái trong **Database** và đồng thời kích hoạt một quy trình phụ (thường là bất đồng bộ qua message queue) để gửi email hoặc thông báo đến cho khách hàng về sự thay đổi này. Sau khi hoàn tất, hệ thống sẽ phản hồi thành công và cập nhật giao diện cho Admin.

Quy trình này đảm bảo việc xử lý đơn hàng được thực hiện một cách có hệ thống, đồng thời giữ cho khách hàng luôn được cập nhật về tình trạng đơn hàng của họ.



Hình mô tả sequence diagram cho chức năng quản lý sản phẩm, thể hiện toàn bộ vòng đời của một sản phẩm trên hệ thống, được điều khiển bởi Quản trị viên (Admin). Sơ đồ bao gồm các lifeline: **Admin**, **WebApp**, **Backend (product-service)**, và **Database**.

Quy trình bắt đầu khi Admin vào trang quản lý và hệ thống tải toàn bộ danh sách sản phẩm. Khối **alt** thể hiện các nghiệp vụ chính mà Admin có thể thực hiện:

1. **[Tạo sản phẩm mới]**: Admin nhập thông tin chi tiết cho một sản phẩm mới. Đặc biệt, khi có hình ảnh được tải lên, **Backend** sẽ xử lý lưu trữ tệp ảnh, nhận về một đường dẫn (URL), và sau đó lưu toàn bộ thông tin sản phẩm cùng URL ảnh vào **Database**.
2. **[Cập nhật sản phẩm]**: Admin có thể chỉnh sửa thông tin của một sản phẩm đã có. **Backend** sẽ nhận dữ liệu mới và cập nhật bản ghi tương ứng trong **Database**.
3. **[Xóa sản phẩm]**: Sau khi Admin xác nhận, **Backend** sẽ gửi lệnh xóa sản phẩm khỏi **Database**, đảm bảo sản phẩm đó không còn xuất hiện trên cửa hàng.

Mỗi hành động đều được hệ thống phản hồi rõ ràng, cho phép Admin toàn quyền kiểm soát danh mục sản phẩm của cửa hàng, từ việc giới thiệu sản phẩm mới đến việc loại bỏ các sản phẩm đã ngừng kinh doanh.

# Các yêu cầu phi chức năng

# 3.1 Hiệu năng (Performance):

Hệ thống được thiết kế và cấu hình để đảm bảo thời gian phản hồi nhanh, mang lại trải nghiệm mượt mà và hiệu quả cho người dùng cuối, ngay cả khi lượng truy cập tăng cao. Các biện pháp tối ưu hóa hiệu năng được áp dụng ở nhiều cấp độ, từ frontend đến backend và hạ tầng triển khai.

# 3.1.1 Tối ưu hóa thời gian tải trang

Đây là yếu tố quan trọng nhất ảnh hưởng đến trải nghiệm người dùng lần đầu. Mục tiêu là giảm thiểu thời gian để người dùng có thể thấy và tương tác với trang web.

**a. Caching tài sản tĩnh phía trình duyệt (Browser-side Static Asset Caching)**

Hệ thống chỉ thị cho trình duyệt của người dùng lưu trữ các tài sản ít thay đổi (như hình ảnh, logo, fonts) vào bộ nhớ đệm (cache) trong một khoảng thời gian nhất định. Trong các lần truy cập sau hoặc khi điều hướng giữa các trang, trình duyệt sẽ sử dụng lại các file này từ cache thay vì phải tải lại từ server, giúp giảm đáng kể thời gian tải trang và tiết kiệm băng thông.

**b. Tối ưu hóa việc tải Font chữ (Font Loading Optimization)**

Font chữ là một tài nguyên có thể chặn hiển thị (render-blocking), nghĩa là trình duyệt sẽ hiển thị một trang trắng cho đến khi tải xong font. Để khắc phục điều này, hệ thống sử dụng chiến lược font-display: swap. Trình duyệt sẽ hiển thị văn bản ngay lập tức bằng một font hệ thống mặc định, và khi font web (Google Fonts) được tải xong, nó sẽ được "tráo đổi" vào. Điều này cải thiện đáng kể hiệu năng cảm nhận (Perceived Performance) vì người dùng có thể đọc nội dung ngay lập tức.

**c. Tối ưu hóa gói ứng dụng (Bundle Optimization)**

* **Mô tả:** Mã nguồn của frontend (React, TypeScript) được biên dịch và đóng gói thành các file JavaScript nhỏ gọn nhất có thể trước khi triển khai. Hệ thống sử dụng **Vite** làm công cụ xây dựng (build tool), một công cụ hiện đại có khả năng thực hiện các kỹ thuật tối ưu hóa tiên tiến như *tree-shaking* (loại bỏ code không được sử dụng) và *code-splitting* (tự động tách code ra thành các gói nhỏ và chỉ tải khi cần).
* **Bằng chứng:** Sự tồn tại của file vite.config.ts trong thư mục SRC/frontend.

# 3.1.2 Hiệu năng cảm nhận và tương tác

Ngoài tốc độ tải trang, trải nghiệm của người dùng còn phụ thuộc vào độ mượt mà khi tương tác với các thành phần trên giao diện.

* **Mô tả:** Thay vì các thay đổi trạng thái đột ngột, hệ thống sử dụng các hiệu ứng chuyển động (animations) tinh tế và được tối ưu hóa hiệu năng cho các hành động của người dùng như mở menu, hiển thị dữ liệu. Các animations này được xử lý bởi một thư viện chuyên dụng để đảm bảo chúng chạy mượt mà trên hầu hết các thiết bị mà không làm "giật" (jank) hay làm chậm giao diện.
* **Bằng chứng từ mã nguồn (SRC/frontend/src/pages/admin/AdminOrderListPage.tsx):**
* **Giải thích:** Việc sử dụng thư viện framer-motion đảm bảo rằng các hiệu ứng chuyển động được tối ưu bằng cách tận dụng tăng tốc phần cứng (GPU acceleration) của trình duyệt, giải phóng luồng xử lý chính để ứng dụng luôn phản hồi nhanh với tương tác của người dùng.

# 3.2 Tính khả dụng (Usability):

Hệ thống phải được thiết kế với giao diện người dùng (UI) trực quan, dễ học và dễ sử dụng, nhằm giảm thiểu rào cản cho cả khách hàng mới và nhân viên quản trị. Mục tiêu là cung cấp một trải nghiệm liền mạch, nơi người dùng có thể đạt được mục tiêu của mình một cách nhanh chóng và hiệu quả.

# 3.2.1 Giao diện trực quan và nhất quán

Giao diện phải cung cấp các phản hồi rõ ràng và sử dụng các biểu tượng (icons) quen thuộc để người dùng có thể hiểu và thao tác mà không cần hướng dẫn chi tiết.

**a. Phản hồi trực quan bằng màu sắc (Visual Feedback via Color-Coding)**

Đặc biệt trong trang quản trị, nơi lượng thông tin lớn, hệ thống sử dụng màu sắc để phân biệt trạng thái của các đối tượng. Ví dụ, mỗi trạng thái của đơn hàng (Chờ xử lý, Đang xử lý, Hoàn thành,...) được gán một màu riêng biệt. Điều này giúp Quản trị viên nắm bắt tình hình tổng quan một cách nhanh chóng chỉ bằng cách lướt nhìn, thay vì phải đọc kỹ từng dòng chữ.

**b. Sử dụng biểu tượng phổ quát (Use of Universal Icons)**

* **Mô tả:** Hệ thống sử dụng một bộ biểu tượng (icon) phổ biến và dễ hiểu cho các hành động chính, giúp người dùng nhận diện chức năng nhanh hơn so với việc đọc nhãn văn bản.
* **Giải thích:** Việc sử dụng các biểu tượng từ một bộ thư viện nhất quán (react-icons/fi) đảm bảo tính đồng bộ về thẩm mỹ và ý nghĩa trên toàn bộ ứng dụng.

# 3.2.2 Tính dễ đọc

Nội dung văn bản trên toàn bộ trang web phải rõ ràng, dễ đọc trên nhiều thiết bị và kích thước màn hình khác nhau.

Hệ thống đã lựa chọn và áp dụng font chữ **"Inter"**, một font được thiết kế đặc biệt cho giao diện người dùng trên màn hình máy tính. "Inter" có các đặc tính giúp tăng cường độ rõ nét của văn bản, ngay cả ở các kích thước chữ nhỏ, làm cho việc đọc các đoạn mô tả sản phẩm dài hoặc duyệt qua các bảng dữ liệu trong trang quản trị trở nên dễ chịu hơn.

# 3.2.3 Điều hướng và truy cập thôn tin rõ ràng

Cấu trúc của trang web phải logic để người dùng có thể dễ dàng tìm thấy thông tin họ cần.

Hệ thống sử dụng các đường dẫn (URL) thân thiện với người dùng, không chứa các đuôi file kỹ thuật như .html hay .php. Ví dụ, URL sẽ là /products/banh-kem-socola thay vì /products.php?id=123. Điều này không chỉ giúp người dùng dễ đọc, dễ nhớ và dễ chia sẻ liên kết mà còn tốt cho việc tối ưu hóa công cụ tìm kiếm (SEO).

# 3.3 Độ tin cậy và khả năng chịu lỗi (Reliability vs Fault Tolerance):

Hệ thống phải được xây dựng để đảm bảo tính toàn vẹn của dữ liệu và duy trì hoạt động ổn định ngay cả khi đối mặt với các lỗi không mong muốn, dù là lỗi nội tại hay từ các dịch vụ bên ngoài.

# 3.3.1 Ưu tiên luồng nghiệp vụ cốt lõi

* **Mô tả:** Trong một quy trình nghiệp vụ phức tạp gồm nhiều bước, hệ thống được thiết kế để ưu tiên hoàn thành bước quan trọng nhất, ngay cả khi các bước phụ trợ sau đó gặp lỗi. Đối với Cake Shop, việc tạo thành công một đơn hàng cho khách là quan trọng nhất.
* **Minh chứng trong hệ thống:** Logic này được thể hiện rõ trong file orders.service.ts. Khi một đơn hàng được tạo, hệ thống sẽ lưu đơn hàng vào cơ sở dữ liệu trước. Sau đó, nó mới gửi yêu cầu xóa giỏ hàng của người dùng. Trong trường hợp yêu cầu xóa giỏ hàng thất bại, hệ thống sẽ chỉ ghi nhận lỗi (log a fault) mà không hủy bỏ giao dịch tạo đơn hàng đã thành công. Điều này đảm bảo không một đơn hàng nào của khách bị mất chỉ vì một lỗi phụ.

# 3.3.2 Sự bảo vệ toàn vẹn của trạng thái

* **Mô tả:** Hệ thống phải thực thi nghiêm ngặt các quy tắc nghiệp vụ để ngăn chặn dữ liệu bị đưa vào các trạng thái không hợp lệ hoặc mâu thuẫn. Điều này cực kỳ quan trọng trong việc quản lý vòng đời của một đơn hàng.
* **Minh chứng trong hệ thống:** Logic cập nhật trạng thái đơn hàng của Admin, cũng nằm trong orders.service.ts, chứa một quy tắc bảo vệ quan trọng. Nó kiểm tra trạng thái hiện tại của đơn hàng trước khi cho phép thay đổi. Nếu một đơn hàng đã ở trạng thái cuối cùng như "Hoàn thành" (COMPLETED) hoặc "Đã hủy" (CANCELLED), hệ thống sẽ từ chối mọi yêu cầu thay đổi trạng thái của nó. Cơ chế này ngăn ngừa các sai sót vận hành và đảm bảo tính cuối cùng của một giao dịch.

# 3.3.3 Khử nhiễu qua giao tiếp bất đồng bộ

* **Mô tả:** Các microservices trong hệ thống không giao tiếp trực tiếp và chờ đợi nhau, mà thông qua một hàng đợi tin nhắn (message queue). payment-service không "gọi" trực tiếp order-service mà chỉ "bắn" một sự kiện (event) vào hàng đợi. Điều này làm cho hệ thống có khả năng chịu lỗi cao hơn.
* **Minh chứng trong hệ thống:** Kiến trúc này được mô tả trong file README.md. Khi một thanh toán thành công, payment-service sẽ phát ra một sự kiện. order-service sẽ lắng nghe và xử lý sự kiện đó. Nếu order-service tạm thời bị lỗi hoặc quá tải, sự kiện thanh toán vẫn nằm an toàn trong hàng đợi và sẽ được xử lý ngay khi order-service hoạt động trở lại. Điều này đảm bảo không có thông tin thanh toán nào bị mất.

# 3.3.4 Xử lí lỗi từ các dịch vụ bên ngoài

* **Mô tả:** Hệ thống không chỉ giả định kịch bản thành công từ các dịch vụ của bên thứ ba (như cổng thanh toán VNPay). Nó còn được thiết kế để xử lý các kịch bản thất bại một cách tường minh.
* **Minh chứng trong hệ thống:** Logic xử lý sự kiện trong orders.service.ts không chỉ lắng nghe sự kiện payment\_processed (thanh toán thành công) mà còn được thiết kế để xử lý cả sự kiện payment\_failed (thanh toán thất bại). Khi nhận được tín hiệu thất bại, hệ thống có thể chủ động cập nhật trạng thái đơn hàng thành "Thất bại", cung cấp thông tin rõ ràng cho cả quản trị viên và khách hàng.

# 3.4 Bảo mật (Security):

Hệ thống phải được thiết kế và xây dựng với các biện pháp bảo mật mạnh mẽ ở nhiều lớp, từ frontend đến backend và cơ sở dữ liệu, nhằm bảo vệ tính toàn vẹn của hệ thống, dữ liệu của người dùng và chống lại các hình thức tấn công phổ biến.

# 3.4.1 Xác thực và phân quyền

Hệ thống phải đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập, và họ chỉ có thể truy cập những tài nguyên và thực hiện những hành động mà họ được phép.

* **Xác thực dựa trên Token (Token-Based Authentication):**
  + **Mô tả:** Sau khi người dùng đăng nhập thành công, user-service sẽ cấp một JSON Web Token (JWT). Token này, được ký điện tử ở phía server, đóng vai trò như một "thẻ thông hành" bảo mật. Trong mọi yêu cầu tiếp theo đến các tài nguyên được bảo vệ, frontend phải gửi kèm JWT này trong header. API Gateway hoặc chính các microservice sẽ xác thực chữ ký của token trước khi xử lý yêu cầu.
  + **Minh chứng trong hệ thống:** Sự tồn tại của auth.controller.ts trong user-service chịu trách nhiệm cấp phát token. Các file như PrivateRoute.tsx và AdminRoute.tsx ở frontend chịu trách nhiệm kiểm tra sự tồn tại của token và điều hướng người dùng.
* **Phân quyền dựa trên vai trò (Role-Based Access Control - RBAC):**
  + **Mô tả:** Hệ thống định nghĩa các vai trò người dùng rõ ràng (ví dụ: Customer và Administrator). Token JWT sau khi được giải mã sẽ chứa thông tin về vai trò của người dùng. Các microservice backend sử dụng thông tin này để quyết định xem người dùng có được phép thực hiện một hành động cụ thể hay không (ví dụ: chỉ người có vai trò Administrator mới có thể truy cập API xóa sản phẩm).
  + **Minh chứng trong hệ thống:** Logic phân quyền này được cài đặt ở tầng controller của các microservice (ví dụ: users.controller.ts, products.controller.ts), thường thông qua các "Guard" trong NestJS.

# 3.4.2 Bảo vệ dữ liệu

Dữ liệu nhạy cảm của người dùng, đặc biệt là mật khẩu, phải được bảo vệ ở cả trạng thái lưu trữ và trạng thái truyền đi.

* **Băm mật khẩu (Password Hashing):**
  + **Mô tả:** Hệ thống **không bao giờ** lưu trữ mật khẩu của người dùng dưới dạng văn bản gốc. Thay vào đó, khi người dùng đăng ký, user-service sẽ sử dụng một thuật toán băm mật khẩu mạnh (như bcrypt) để chuyển mật khẩu thành một chuỗi ký tự không thể đảo ngược. Khi người dùng đăng nhập, hệ thống sẽ băm mật khẩu được nhập vào và so sánh với chuỗi đã lưu.
  + **Minh chứng trong hệ thống:** Logic này nằm trong user.service.ts hoặc user.entity.ts, sử dụng các thư viện như bcrypt để thực hiện việc băm và so sánh.
* **Mã hóa dữ liệu trên đường truyền (Data-in-Transit Encryption):**
  + **Mô tả:** Toàn bộ giao tiếp giữa trình duyệt của người dùng và các máy chủ của hệ thống được mã hóa bằng giao thức HTTPS. Điều này đảm bảo rằng bất kỳ dữ liệu nào được trao đổi, từ thông tin đăng nhập đến chi tiết đơn hàng, đều được bảo vệ khỏi việc bị nghe lén trên mạng.
  + **Minh chứng trong hệ thống:** Việc triển khai ứng dụng trên Firebase Hosting mặc định sẽ được cung cấp và ép buộc sử dụng chứng chỉ SSL/TLS, đảm bảo kết nối HTTPS.

# 3.4.3 Phòng chống các lỗ hổng web phổ biến

Hệ thống được xây dựng trên các framework hiện đại có sẵn các cơ chế để chống lại các loại tấn công phổ biến.

* **Chống SQL Injection:**
  + **Mô tả:** Hệ thống sử dụng một Object-Relational Mapper (ORM) như TypeORM. ORM này sử dụng các truy vấn được tham số hóa (parameterized queries) theo mặc định, thay vì nối chuỗi thủ công để xây dựng câu lệnh SQL. Cơ chế này loại bỏ hoàn toàn nguy cơ bị tấn công SQL Injection, một trong những lỗ hổng web nguy hiểm nhất.
  + **Minh chứng trong hệ thống:** Sự hiện diện của các file .entity.ts (ví dụ: user.entity.ts) và các phương thức của repository (ví dụ: this.usersRepository.find()) là dấu hiệu của việc sử dụng ORM.
* **Chính sách chia sẻ tài nguyên CORS (Cross-Origin Resource Sharing):**
  + **Mô tả:** Các microservice backend được cấu hình để chỉ chấp nhận các yêu cầu đến từ các domain (tên miền) đã được chỉ định (ví dụ: domain của ứng dụng frontend). Điều này ngăn chặn các trang web độc hại khác gọi trực tiếp đến API của hệ thống để đánh cắp dữ liệu.
  + **Minh chứng trong hệ thống:** Cấu hình app.enableCors() trong file main.ts của mỗi microservice backend là nơi thiết lập chính sách này.

# 3.5 Pháp luật và tuân thủ (Legal vs Compliance):

Hệ thống được phát triển để tuân thủ các quy định về thương mại điện tử và quyền sở hữu trí tuệ, đồng thời đảm bảo tính minh bạch và an toàn trong các giao dịch tài chính theo tiêu chuẩn của các đối tác thanh toán.

# 3.5.1 Bảo vệ quyền sở hữu trí tuệ

* **Mô tả:** Hệ thống có một cách tiếp cận rõ ràng để bảo vệ các tài sản trí tuệ độc quyền của doanh nghiệp. Một trang "Điều khoản Dịch vụ" chuyên biệt đã được tạo ra, trong đó khẳng định mạnh mẽ quyền sở hữu của Cake Shop đối với các nội dung sáng tạo như hình ảnh sản phẩm, công thức, và thiết kế. Người dùng bắt buộc phải đồng ý với các điều khoản này khi đăng ký, tạo ra một thỏa thuận pháp lý cơ bản.
* **Minh chứng trong hệ thống:** Logic và nội dung cho yêu cầu này được thể hiện trực tiếp trong file SRC/frontend/src/pages/TermsPage.tsx. Trang này liệt kê các điều khoản, và luồng đăng ký (Use Case UC-02) yêu cầu người dùng phải tích vào ô đồng ý trước khi có thể tạo tài khoản.

# 3.5.2 Tuân thủ tiêu chuẩn công thanh toán

* **Mô tả:** Khi tích hợp với một dịch vụ tài chính của bên thứ ba như VNPay, hệ thống phải tuân thủ nghiêm ngặt các đặc tả kỹ thuật và quy trình bảo mật mà họ đưa ra. Hệ thống Cake Shop thể hiện sự tuân thủ này bằng cách:
  1. **Định dạng dữ liệu chính xác:** Gửi các tham số theo đúng định dạng yêu cầu, ví dụ như giá trị vnp\_Amount phải được nhân với 100.
  2. **Xác thực giao dịch bằng Chữ ký bảo mật:** Tạo và xác thực tham số vnp\_SecureHash. Đây là một chuỗi băm được tạo ra từ dữ liệu giao dịch và một "secret key" chỉ được chia sẻ giữa Cake Shop và VNPay. Việc này đảm bảo rằng dữ liệu thanh toán không bị thay đổi trên đường truyền và các thông báo xác nhận (IPN) thực sự đến từ VNPay, chứ không phải một kẻ tấn công mạo danh.
* **Minh chứng trong hệ thống:**
  1. Các quy tắc về định dạng dữ liệu và quy trình gọi IPN được ghi chú chi tiết trong file SRC/README.md.
  2. Logic xử lý các tham số trả về từ VNPay, bao gồm mã phản hồi vnp\_ResponseCode, được cài đặt trong file SRC/frontend/src/pages/VnPayReturnPage.tsx.
  3. Phần quan trọng nhất, logic tạo và xác thực vnp\_SecureHash, nằm ở phía backend trong payment-service, đảm bảo secret key không bị lộ ra phía client.

# 3.6 Khả năng bảo trì (Maintainability):

Hệ thống phải được xây dựng với mã nguồn có cấu trúc rõ ràng, dễ hiểu và dễ sửa đổi. Điều này nhằm mục đích giảm chi phí và thời gian cho việc sửa lỗi, nâng cấp công nghệ, hoặc phát triển các tính năng mới trong tương lai.

# 3.6.1 Kiến trúc phân tách rõ ràng

* **Mô tả:** Hệ thống được xây dựng dựa trên kiến trúc **Microservices**. Thay vì một khối ứng dụng nguyên khối (monolith), các chức năng nghiệp vụ chính được tách thành các dịch vụ độc lập, mỗi dịch vụ có một trách nhiệm duy nhất (ví dụ: user-service chỉ quản lý người dùng, product-service chỉ quản lý sản phẩm). Cách tiếp cận này mang lại lợi ích to lớn cho khả năng bảo trì:
  + **Giảm độ phức tạp:** Lập trình viên có thể tập trung vào một phần nhỏ của hệ thống mà không cần hiểu toàn bộ logic phức tạp.
  + **Phát triển độc lập:** Các đội nhóm khác nhau có thể làm việc song song trên các dịch vụ khác nhau.
  + **Cô lập lỗi:** Một lỗi trong order-service sẽ không làm sập toàn bộ hệ thống; các chức năng như duyệt sản phẩm hay đăng nhập vẫn hoạt động bình thường.
* **Minh chứng trong hệ thống:** Cấu trúc thư mục gốc của dự án cho thấy rõ các thư mục riêng biệt cho cart-service, order-service, payment-service, product-service, và user-service.

# 3.6.2 Mã nguồn dễ đọc, tính module hóa cao

* **Mô tả:** Mã nguồn được tổ chức theo các nguyên tắc thực hành tốt nhất, ưu tiên tính rõ ràng và khả năng tái sử dụng.
  + **Tách biệt logic và giao diện:** Các logic phức tạp được tách ra khỏi phần hiển thị (JSX). Ví dụ, thay vì nhúng một khối switch-case lớn để quyết định màu sắc ngay trong component, logic đó được đóng gói vào một hàm tiện ích getStatusColor. Điều này làm cho component chính gọn gàng hơn và hàm tiện ích có thể được tái sử dụng ở nơi khác.
  + **Styling nhất quán với Utility-First CSS:** Hệ thống sử dụng Tailwind CSS. Cách tiếp cận này thúc đẩy một hệ thống thiết kế nhất quán bằng cách sử dụng các lớp tiện ích được định nghĩa trước, thay vì viết các quy tắc CSS tùy chỉnh cho mỗi thành phần. Điều này giúp cho việc bảo trì giao diện trở nên dễ dàng hơn rất nhiều.
* **Minh chứng trong hệ thống:**
  + Hàm getStatusColor trong file AdminOrderListPage.tsx là một ví dụ hoàn hảo cho việc tách biệt logic.
  + File index.css cho thấy việc import và thiết lập Tailwind CSS làm nền tảng cho toàn bộ hệ thống styling.

# 3.6.3 Công nghệ hiện đại, cộng đồng mạnh

* **Mô tả:** Việc lựa chọn các công nghệ và framework hiện đại, được sử dụng rộng rãi, là một yếu tố then chốt cho khả năng bảo trì lâu dài.
  + **Static Typing với TypeScript:** Toàn bộ dự án, từ frontend đến backend, đều sử dụng TypeScript. Việc này giúp bắt lỗi ngay trong quá trình phát triển thay vì ở môi trường thực tế. Nó cũng làm cho mã nguồn tự mô tả (self-documenting), giúp các lập trình viên mới dễ dàng hiểu được cấu trúc dữ liệu và các hàm mong đợi loại đầu vào nào.
  + **Framework có cấu trúc rõ ràng:** Việc sử dụng React cho frontend và đặc biệt là **NestJS** cho backend mang lại một cấu trúc dự án được tổ chức tốt. NestJS, với kiến trúc module, controller, và service, "ép" các lập trình viên phải tuân theo một khuôn mẫu rõ ràng, giúp mã nguồn luôn có tổ chức và dễ dàng mở rộng.
* **Minh chứng trong hệ thống:** Sự hiện diện của các file có đuôi .ts và .tsx trên toàn bộ dự án, cùng với các file cấu hình như vite.config.ts và cấu trúc module của các service backend.