**CHAPTER 1: REQUIREMENTS OF E-COMMERCE**

**1.1 Determine Requirements**

**1.1.1 Actors**

Trong hệ thống thương mại điện tử, các **actors** chính bao gồm:

1. **Customer (Khách hàng)**
   * Đối tượng chính của hệ thống, sử dụng nền tảng để tìm kiếm, mua sắm và thanh toán các sản phẩm.
   * Họ có thể tạo tài khoản, duyệt qua danh mục sản phẩm, thêm sản phẩm vào giỏ hàng và thực hiện mua hàng.
   * Ảnh hưởng: Họ đóng vai trò quyết định trong doanh thu của nền tảng, do đó cần đảm bảo trải nghiệm mua sắm mượt mà.
   * Các kiểu customer:

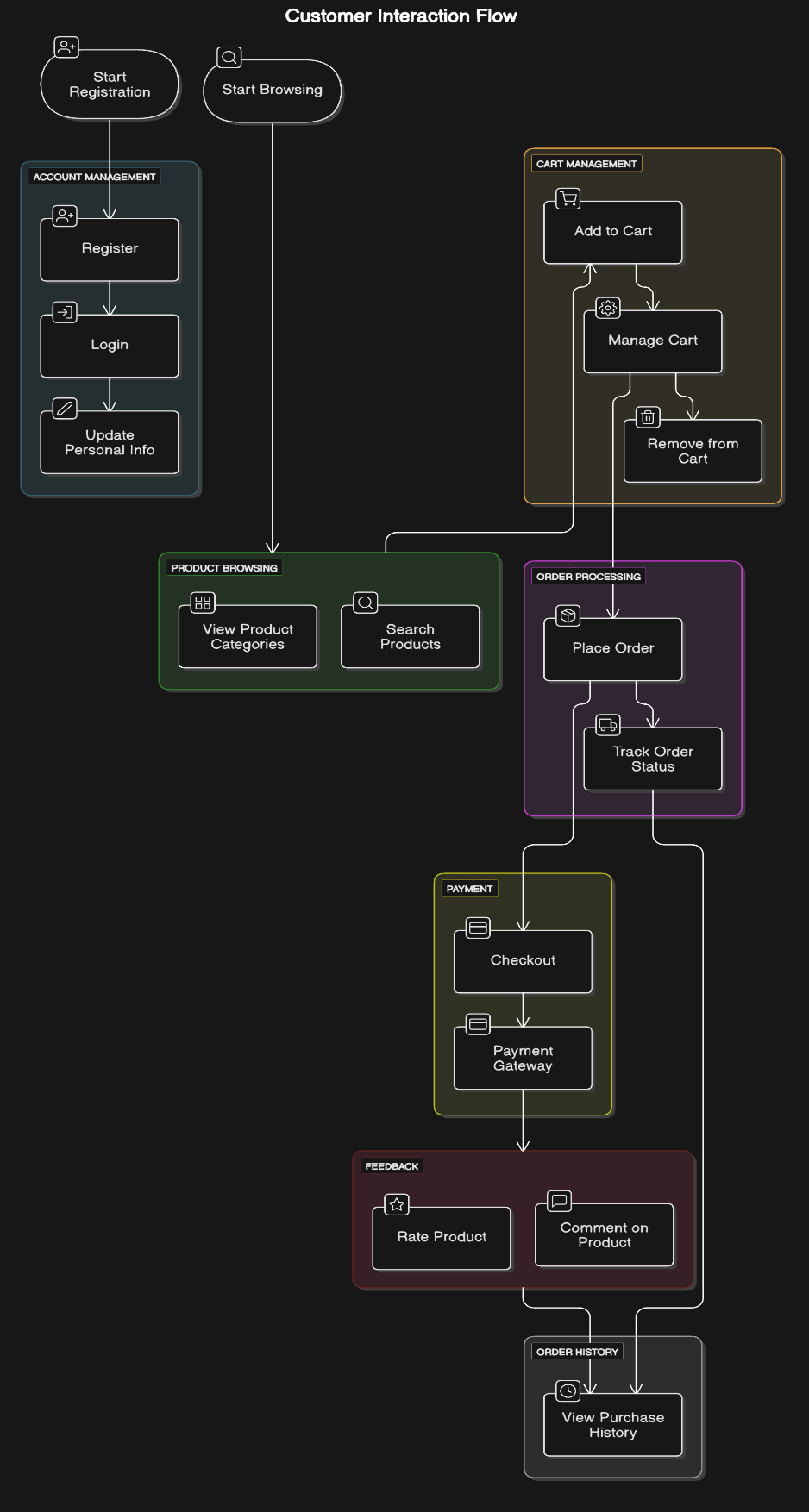
| **Loại Customer** | **Mô tả** |  |
| --- | --- | --- |
| **Khách hàng mới (New Customer)** | Chưa từng mua hàng, mới đăng ký hoặc chỉ duyệt sản phẩm. |  |
| **Khách hàng quay lại (Returning Customer)** | Đã mua hàng ít nhất 1 lần trước đó. |  |
| **Khách hàng trung thành (Loyal Customer)** | Mua hàng nhiều lần, thường xuyên. |  |
| **Khách hàng giá trị cao (High-Value Customer)** | Giá trị đơn hàng trung bình lớn, mua hàng định kỳ. |  |
| **Khách hàng tiềm năng thấp (Low Engagement)** | Đăng ký nhưng không tương tác nhiều, không mua. |  |

1. **Seller (Người bán)**
   * Người cung cấp sản phẩm trên nền tảng, chịu trách nhiệm quản lý cửa hàng của mình.
   * Họ có thể đăng tải, chỉnh sửa, xóa sản phẩm cũng như quản lý đơn hàng và doanh thu.
   * Ảnh hưởng: Số lượng và chất lượng sản phẩm phụ thuộc vào người bán, do đó cần có cơ chế kiểm soát chất lượng sản phẩm.
2. **Admin (Quản trị viên)**
   * Người quản lý toàn bộ hệ thống, có quyền kiểm soát nội dung, người dùng và xử lý các vấn đề phát sinh.
   * Họ có quyền xét duyệt sản phẩm mới, giải quyết khiếu nại của khách hàng và quản lý phương thức thanh toán.
   * Ảnh hưởng: Vai trò quan trọng để duy trì sự ổn định và an toàn của nền tảng.
3. **Delivery Staff (Nhân viên giao hàng)**
   * Người thực hiện giao đơn hàng từ kho hoặc cửa hàng đến khách hàng.
   * Họ có thể cập nhật trạng thái đơn hàng, xác nhận khi hàng đã giao thành công.
   * Ảnh hưởng: Quá trình vận chuyển ảnh hưởng đến thời gian và trải nghiệm khách hàng, do đó cần hệ thống cập nhật chính xác trạng thái đơn hàng.
4. **Payment System (Hệ thống thanh toán)**
   * Cổng thanh toán bên thứ ba hoặc hệ thống nội bộ xử lý các giao dịch mua hàng.
   * Xác nhận thanh toán thành công hoặc thất bại và thông báo lại cho hệ thống.
   * Ảnh hưởng: Nếu hệ thống thanh toán gặp sự cố, có thể gây mất doanh thu và làm giảm trải nghiệm khách hàng.

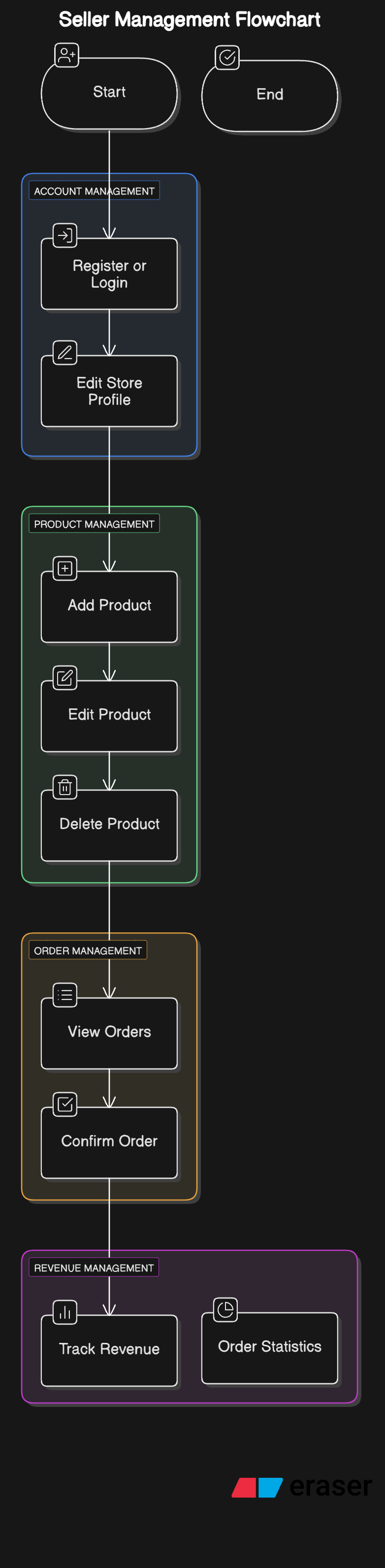
**1.1.2 Functions with Respect to Actors**

|  |  |  |
| --- | --- | --- |
| **Actor** | **Functions** | **Description** |
| **Customer** | Đăng ký, đăng nhập, cập nhật thông tin cá nhân | Khách hàng có thể tạo tài khoản, đăng nhập và chỉnh sửa thông tin cá nhân của mình. |
|  | Xem danh mục sản phẩm, tìm kiếm sản phẩm | Khách hàng có thể duyệt qua danh sách sản phẩm hoặc tìm kiếm theo từ khóa. |
|  | Thêm sản phẩm vào giỏ hàng, quản lý giỏ hàng | Cho phép khách hàng thêm/xóa sản phẩm trong giỏ hàng trước khi thanh toán. |
|  | Đặt hàng, theo dõi trạng thái đơn hàng | Sau khi đặt hàng, khách hàng có thể kiểm tra trạng thái vận chuyển của đơn hàng. |
|  | Thanh toán đơn hàng thông qua cổng thanh toán | Hỗ trợ nhiều phương thức thanh toán (thẻ, ví điện tử). |
|  | Đánh giá, bình luận sản phẩm | Khách hàng có thể để lại đánh giá và nhận xét về sản phẩm đã mua. |
|  | Xem lịch sử mua hàng | Cho phép khách hàng xem lại các đơn hàng trước đó. |
| **Seller** | Đăng ký, đăng nhập, quản lý hồ sơ cửa hàng | Người bán có thể tạo tài khoản và chỉnh sửa thông tin cửa hàng của mình. |
|  | Thêm, chỉnh sửa, xóa sản phẩm | Người bán có thể quản lý danh mục sản phẩm của mình trên nền tảng. |
|  | Xem danh sách đơn hàng và xác nhận đơn hàng | Người bán có thể kiểm tra các đơn hàng nhận được và xác nhận giao hàng. |
|  | Quản lý doanh thu, thống kê đơn hàng | Người bán có thể theo dõi doanh thu từ các giao dịch đã hoàn tất. |
| **Admin** | Quản lý người dùng (customer, seller, delivery staff) | Admin có thể thêm, chỉnh sửa hoặc khóa tài khoản người dùng. |
|  | Quản lý danh mục sản phẩm, xét duyệt sản phẩm mới | Xác nhận hoặc từ chối các sản phẩm do người bán đăng lên. |
|  | Quản lý đơn hàng, xử lý khiếu nại của khách hàng | Giải quyết các tranh chấp hoặc vấn đề phát sinh liên quan đến đơn hàng. |
|  | Quản lý phương thức thanh toán | Cấu hình và kiểm tra hoạt động của hệ thống thanh toán. |
| **Delivery Staff** | Xem danh sách đơn hàng cần giao | Nhận danh sách các đơn hàng cần giao từ hệ thống. |
|  | Cập nhật trạng thái đơn hàng (đang giao, đã giao) | Cập nhật trạng thái giao hàng trong hệ thống khi đã hoàn thành. |
| **Payment System** | Xử lý thanh toán đơn hàng thông qua các phương thức (thẻ, ví điện tử) | Hệ thống kết nối với ngân hàng hoặc ví điện tử để thực hiện thanh toán. |
|  | Xác nhận thanh toán thành công và thông báo cho hệ thống | Gửi thông báo xác nhận khi giao dịch được thực hiện thành công hoặc thất bại. |

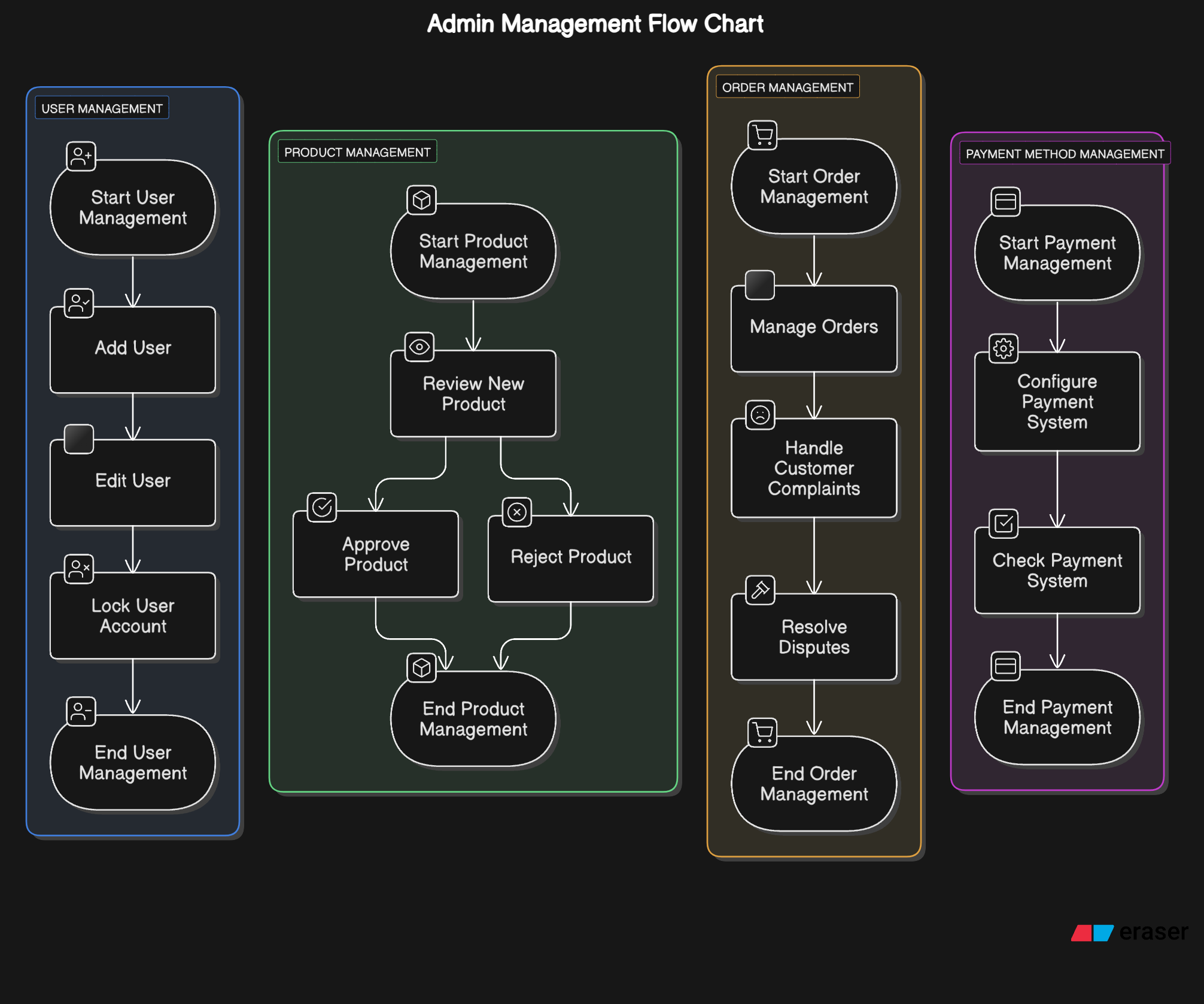
Customer



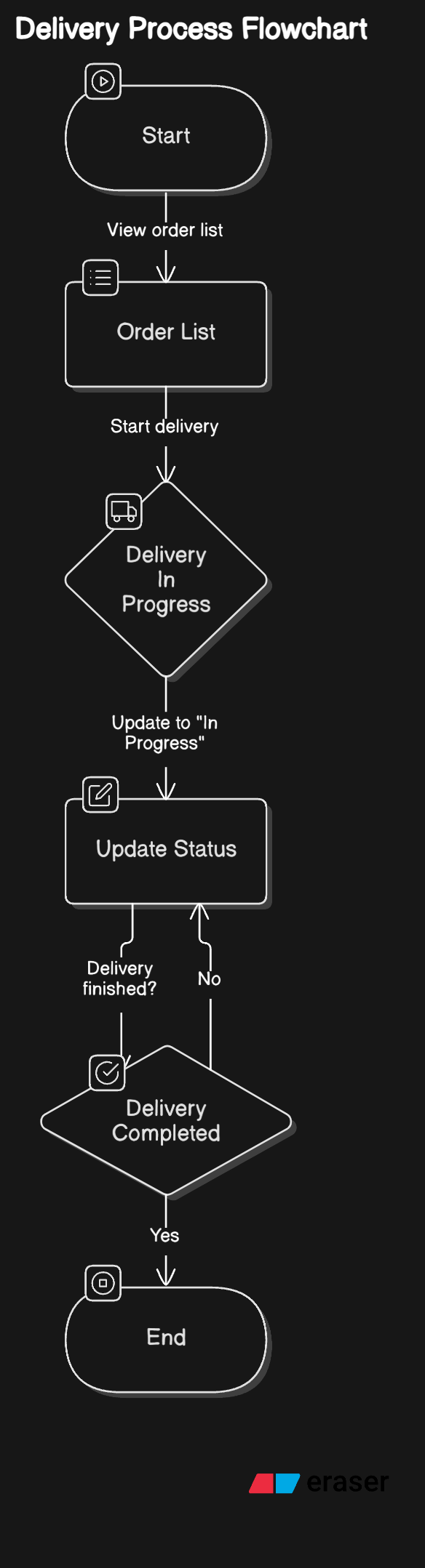
Seller



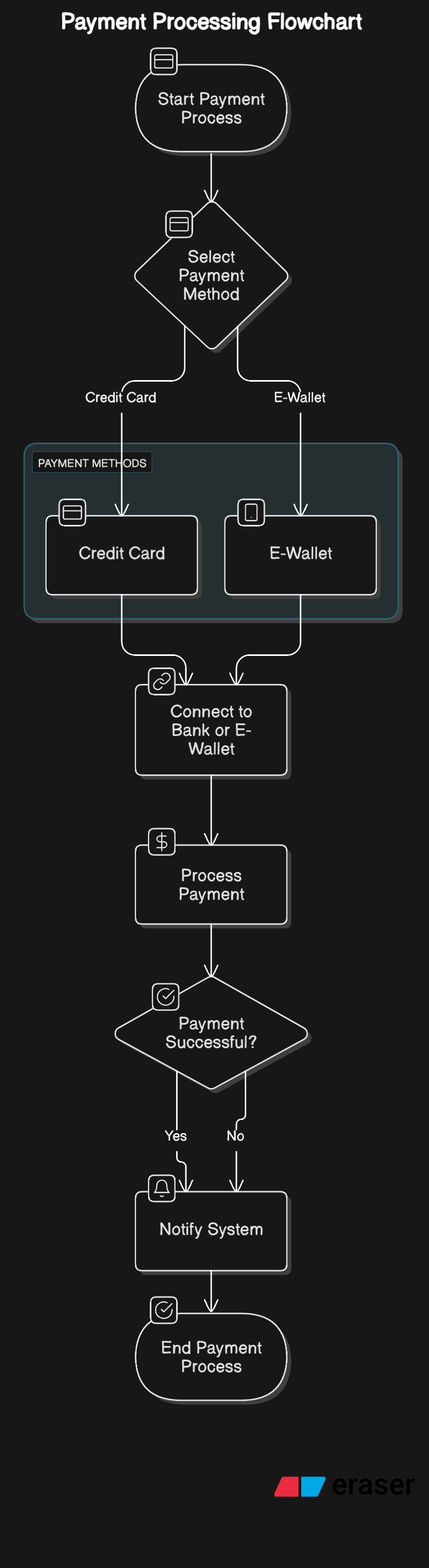
Admin



Delivery

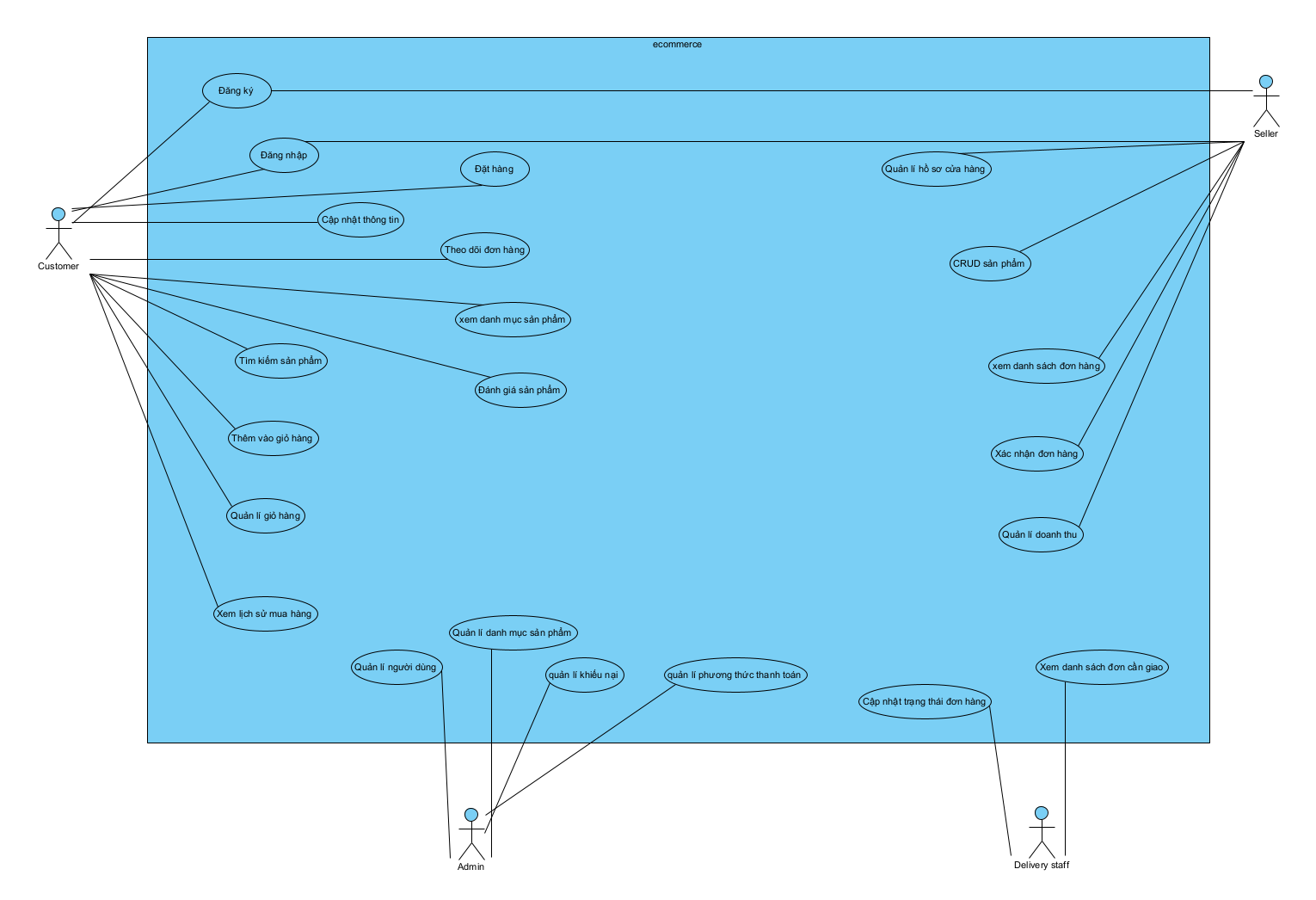


Payment



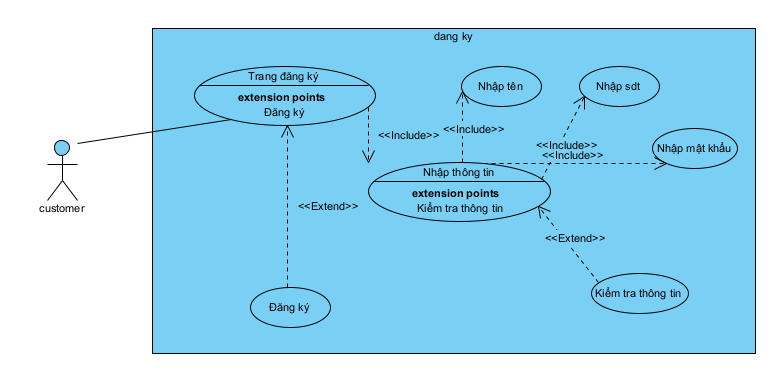
1.1.3 Use case diagram

Usecase tổng quát



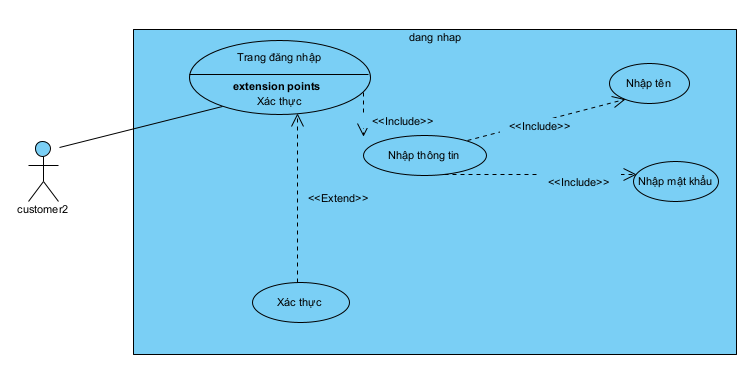
**1. Customer Registration, Login & Profile Update**

**Use Case: Customer Registration**



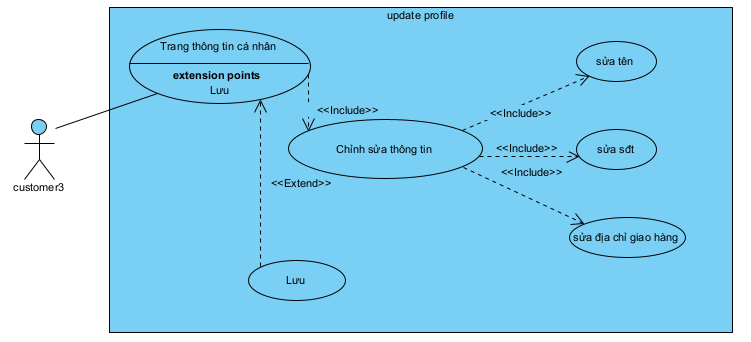
* **Mục đích:** Cho phép khách hàng mới tạo tài khoản trên hệ thống.
* **Preconditions:** Người dùng chưa có tài khoản; truy cập trang đăng ký.
* **Basic Flow:**
  1. Khách hàng truy cập trang đăng ký.
  2. Nhập thông tin bắt buộc (tên, email, mật khẩu, số điện thoại, …).
  3. Hệ thống kiểm tra tính hợp lệ của thông tin nhập vào (email, mật khẩu đủ mạnh, …).
  4. Hệ thống gửi email xác nhận (nếu cần) và tạo tài khoản mới.
  5. Khách hàng nhận thông báo đăng ký thành công.
* **Postconditions:** Tài khoản của khách hàng được tạo và lưu trữ trong hệ thống.
* **Alternative Flows:**
  1. Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu chỉnh sửa.
  2. Nếu email đã tồn tại, hiển thị cảnh báo và đề nghị đăng nhập hoặc dùng email khác.

**Use Case: Customer Login**



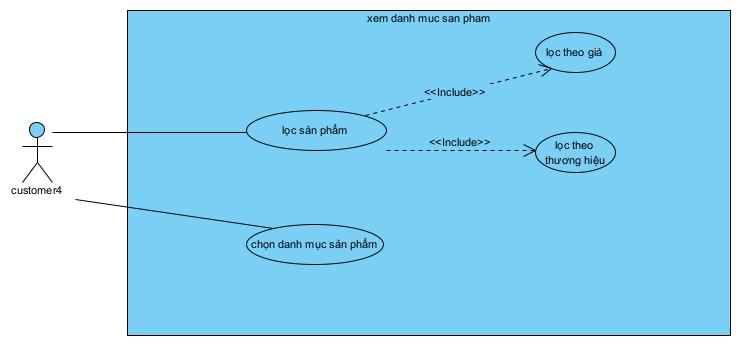
* **Mục đích:** Cho phép khách hàng truy cập vào hệ thống với tài khoản đã đăng ký.
* **Preconditions:** Tài khoản đã được tạo, khách hàng nhớ thông tin đăng nhập.
* **Basic Flow:**
  1. Khách hàng truy cập trang đăng nhập.
  2. Nhập email và mật khẩu.
  3. Hệ thống xác thực thông tin.
  4. Nếu đúng, chuyển hướng khách hàng đến trang chủ hoặc dashboard cá nhân.
* **Postconditions:** Khách hàng được đăng nhập thành công và có thể sử dụng các chức năng riêng của mình.
* **Alternative Flows:**
  1. Nếu thông tin đăng nhập không chính xác, hệ thống hiển thị thông báo lỗi.
  2. Tùy chọn “Quên mật khẩu” sẽ hướng khách hàng qua quy trình khôi phục mật khẩu.

**Use Case: Update Profile**



* **Mục đích:** Cho phép khách hàng cập nhật thông tin cá nhân, bao gồm địa chỉ giao hàng, số điện thoại, …
* **Preconditions:** Khách hàng đã đăng nhập vào hệ thống.
* **Basic Flow:**
  1. Khách hàng truy cập trang “Thông tin cá nhân” hoặc “Cài đặt tài khoản”.
  2. Chỉnh sửa các trường thông tin cần cập nhật.
  3. Nhấn nút “Lưu” để xác nhận thay đổi.
  4. Hệ thống cập nhật thông tin và hiển thị thông báo thành công.
* **Postconditions:** Thông tin cá nhân của khách hàng được cập nhật và lưu trữ trên hệ thống.
* **Alternative Flows:**
  1. Nếu có lỗi trong quá trình lưu (ví dụ: định dạng email không hợp lệ), hệ thống thông báo và yêu cầu chỉnh sửa.

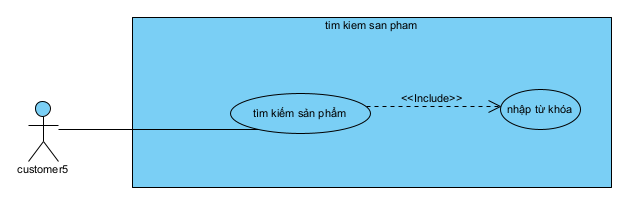
**2. Browse & Search Products**

****

**Use Case: Browse Products**

* **Mục đích:** Cho phép khách hàng duyệt qua danh mục sản phẩm để tìm kiếm sản phẩm cần mua.
* **Preconditions:** Khách hàng đã truy cập vào trang chính hoặc trang danh mục sản phẩm.
* **Basic Flow:**
  1. Khách hàng chọn danh mục sản phẩm hoặc duyệt theo các bộ lọc (giá, thương hiệu, …).
  2. Hệ thống hiển thị danh sách các sản phẩm phù hợp.
  3. Khách hàng có thể nhấp vào sản phẩm để xem chi tiết.
* **Postconditions:** Khách hàng nắm được thông tin cơ bản của sản phẩm và có thể tiếp tục vào các use case khác (thêm vào giỏ, xem chi tiết, …).
* **Alternative Flows:**
  1. Nếu không có sản phẩm nào phù hợp, hệ thống thông báo “Không tìm thấy sản phẩm phù hợp”.

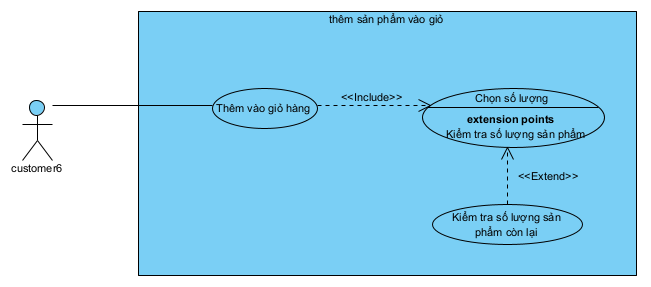
**Use Case: Search Products**



* **Mục đích:** Cho phép khách hàng tìm kiếm sản phẩm cụ thể thông qua từ khóa.
* **Preconditions:** Khách hàng có ý định tìm kiếm sản phẩm cụ thể; truy cập trang tìm kiếm.
* **Basic Flow:**
  1. Khách hàng nhập từ khóa tìm kiếm (tên sản phẩm, thương hiệu, …).
  2. Hệ thống xử lý và tìm kiếm sản phẩm khớp với từ khóa.
  3. Hệ thống hiển thị danh sách kết quả tìm kiếm.
* **Postconditions:** Khách hàng nhận được kết quả tìm kiếm chính xác và có thể chọn sản phẩm muốn xem chi tiết.
* **Alternative Flows:**
  1. Nếu từ khóa không khớp với bất kỳ sản phẩm nào, hiển thị thông báo “Không có kết quả phù hợp”.

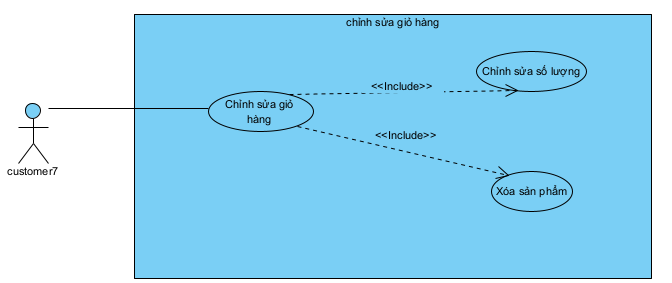
**3. Shopping Cart Management**

**Use Case: Add Product to Shopping Cart**



* **Mục đích:** Cho phép khách hàng thêm sản phẩm vào giỏ hàng để chuẩn bị mua.
* **Preconditions:** Khách hàng đang xem trang chi tiết sản phẩm; đã chọn số lượng hoặc các tùy chọn nếu có.
* **Basic Flow:**
  1. Khách hàng nhấp nút “Thêm vào giỏ hàng” trên trang chi tiết sản phẩm.
  2. Hệ thống kiểm tra tính khả dụng của sản phẩm (số lượng còn lại, …).
  3. Nếu sản phẩm khả dụng, hệ thống cập nhật giỏ hàng của khách hàng với sản phẩm mới.
  4. Hiển thị thông báo thành công và cập nhật số lượng sản phẩm trong giỏ hàng.
* **Postconditions:** Sản phẩm được thêm vào giỏ hàng, sẵn sàng cho quy trình đặt hàng.
* **Alternative Flows:**
  1. Nếu sản phẩm hết hàng hoặc không khả dụng, hệ thống thông báo lỗi và không thêm vào giỏ hàng.

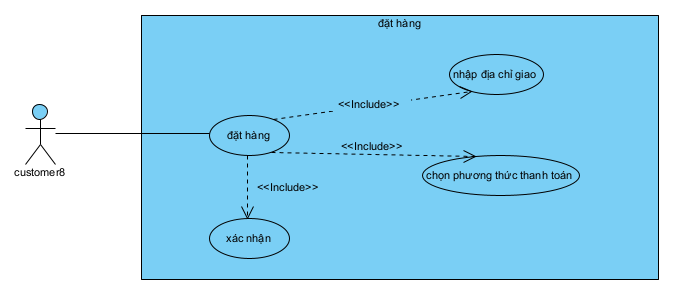
**Use Case: Manage Shopping Cart**



* **Mục đích:** Cho phép khách hàng chỉnh sửa giỏ hàng (cập nhật số lượng, xóa sản phẩm).
* **Preconditions:** Khách hàng đã thêm sản phẩm vào giỏ hàng; truy cập trang giỏ hàng.
* **Basic Flow:**
  1. Khách hàng truy cập trang “Giỏ hàng”.
  2. Chỉnh sửa số lượng của sản phẩm hoặc xóa sản phẩm không muốn mua.
  3. Hệ thống cập nhật lại giỏ hàng và tính toán tổng giá trị đơn hàng.
* **Postconditions:** Giỏ hàng được cập nhật theo yêu cầu của khách hàng, chuẩn bị cho quá trình thanh toán.
* **Alternative Flows:**
  1. Nếu cập nhật số lượng vượt quá tồn kho, hệ thống hiển thị thông báo lỗi và giới hạn số lượng tối đa.

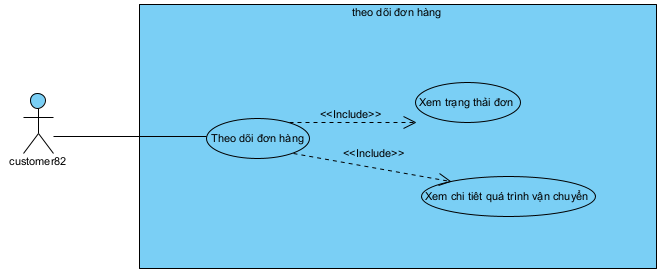
**4. Order Placement & Order Tracking**

**Use Case: Place Order**



* **Mục đích:** Cho phép khách hàng đặt hàng dựa trên các sản phẩm trong giỏ hàng.
* **Preconditions:** Giỏ hàng có chứa sản phẩm; khách hàng đã đăng nhập.
* **Basic Flow:**
  1. Khách hàng kiểm tra lại giỏ hàng và nhấn nút “Đặt hàng”.
  2. Hệ thống hiển thị trang xác nhận đơn hàng, cho phép khách hàng nhập địa chỉ giao hàng, phương thức vận chuyển và thanh toán.
  3. Khách hàng xác nhận thông tin và nhấn “Xác nhận đặt hàng”.
  4. Hệ thống tạo đơn hàng, lưu trữ chi tiết và chuyển trạng thái đơn hàng sang “Đang xử lý”.
* **Postconditions:** Đơn hàng được tạo và lưu trữ trong hệ thống; khách hàng nhận thông báo xác nhận đơn hàng.
* **Alternative Flows:**
  1. Nếu có lỗi trong quá trình tạo đơn hàng (ví dụ: thông tin giao hàng không hợp lệ), hệ thống thông báo và cho phép khách hàng chỉnh sửa lại.

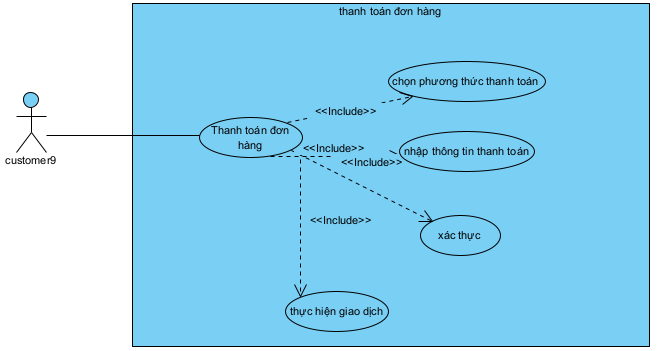
**Use Case: Track Order Status**



* **Mục đích:** Cho phép khách hàng theo dõi trạng thái của đơn hàng sau khi đã đặt.
* **Preconditions:** Đơn hàng đã được đặt và có trạng thái cập nhật từ hệ thống (đang xử lý, đang giao, đã giao, …).
* **Basic Flow:**
  1. Khách hàng đăng nhập và truy cập trang “Theo dõi đơn hàng” hoặc “Lịch sử mua hàng”.
  2. Hệ thống hiển thị danh sách đơn hàng cùng với trạng thái hiện tại của từng đơn.
  3. Khách hàng nhấp vào đơn hàng để xem chi tiết quá trình vận chuyển.
* **Postconditions:** Khách hàng nhận được thông tin cập nhật về trạng thái đơn hàng và có thể dựa vào đó để theo dõi giao hàng.
* **Alternative Flows:**
  1. Nếu không có thông tin cập nhật, hệ thống hiển thị trạng thái “Đang xử lý” hoặc thông báo “Thông tin cập nhật đang được xử lý”.

**5. Payment Processing**

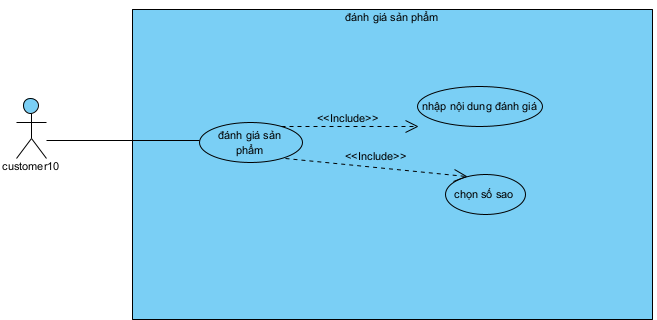
**Use Case: Payment for Order**



* **Mục đích:** Cho phép khách hàng thanh toán đơn hàng thông qua các cổng thanh toán (thẻ, ví điện tử, …).
* **Preconditions:** Đơn hàng đã được xác nhận; khách hàng đã chọn phương thức thanh toán.
* **Basic Flow:**
  1. Khách hàng chọn phương thức thanh toán trên trang xác nhận đơn hàng.
  2. Hệ thống chuyển hướng đến cổng thanh toán bên thứ ba hoặc thực hiện thanh toán nội bộ.
  3. Khách hàng nhập thông tin thanh toán cần thiết (số thẻ, mã OTP, …).
  4. Hệ thống xác thực và thực hiện giao dịch thanh toán.
  5. Nếu giao dịch thành công, hệ thống cập nhật trạng thái đơn hàng sang “Đã thanh toán” và gửi thông báo cho khách hàng.
* **Postconditions:** Thanh toán thành công, đơn hàng được chuyển sang trạng thái tiếp theo (chờ giao hàng).
* **Alternative Flows:**
  1. Nếu giao dịch không thành công, hệ thống hiển thị thông báo lỗi và cho phép khách hàng thử lại hoặc chọn phương thức thanh toán khác.

**6. Product Rating & Reviews**

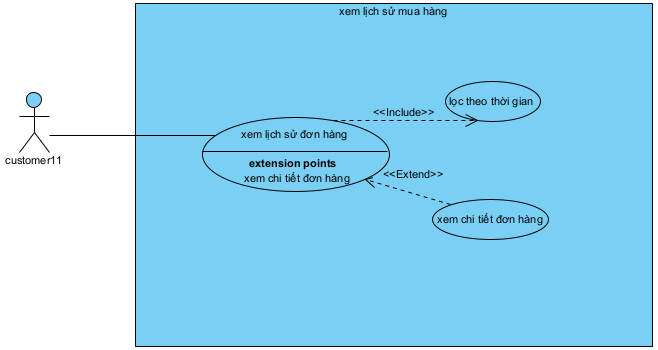
**Use Case: Rate and Review Product**



* **Mục đích:** Cho phép khách hàng chia sẻ đánh giá và nhận xét về sản phẩm đã mua, góp phần cải thiện chất lượng dịch vụ.
* **Preconditions:** Khách hàng đã mua sản phẩm và đơn hàng đã được giao thành công; khách hàng đăng nhập vào hệ thống.
* **Basic Flow:**
  1. Khách hàng truy cập trang chi tiết sản phẩm hoặc lịch sử mua hàng.
  2. Chọn chức năng “Đánh giá” và nhập nội dung nhận xét kèm theo số sao đánh giá.
  3. Hệ thống lưu thông tin đánh giá và cập nhật cho sản phẩm.
  4. Đánh giá được hiển thị cho các khách hàng khác khi xem thông tin sản phẩm.
* **Postconditions:** Đánh giá và nhận xét của khách hàng được lưu trữ, góp phần xây dựng uy tín sản phẩm.
* **Alternative Flows:**
  1. Nếu khách hàng chưa mua sản phẩm, chức năng đánh giá sẽ bị vô hiệu hóa hoặc hiển thị thông báo yêu cầu mua hàng trước khi đánh giá.

**7. Order History Viewing**

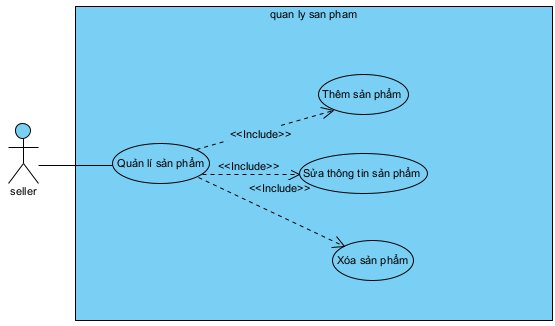
**Use Case: View Order History**



* **Mục đích:** Cho phép khách hàng xem lại lịch sử các đơn hàng đã đặt, bao gồm chi tiết đơn hàng, trạng thái, và thông tin giao dịch.
* **Preconditions:** Khách hàng đã đăng nhập và có lịch sử mua hàng trong hệ thống.
* **Basic Flow:**
  1. Khách hàng đăng nhập và truy cập trang “Lịch sử mua hàng”.
  2. Hệ thống hiển thị danh sách các đơn hàng theo thời gian, với các thông tin cơ bản (mã đơn hàng, ngày đặt, trạng thái).
  3. Khách hàng có thể nhấp vào từng đơn hàng để xem chi tiết hơn như sản phẩm, giá tiền, thông tin giao hàng.
* **Postconditions:** Khách hàng có thể kiểm tra lại các giao dịch đã thực hiện, phục vụ cho việc theo dõi hoặc yêu cầu hỗ trợ khi cần.
* **Alternative Flows:**
  1. Nếu không có đơn hàng nào, hệ thống hiển thị thông báo “Bạn chưa có đơn hàng nào”.

Seller

**Use Case: Quản lý sản phẩm**

****

**1. Tên Use Case: Quản lý sản phẩm**

**2. Mô tả: Cho phép seller thêm, sửa, xóa và cập nhật thông tin sản phẩm của họ trên nền tảng.**

**3. Actor: Seller**

**4. Luồng chính:**

1. Seller đăng nhập vào hệ thống.
2. Seller truy cập vào trang quản lý sản phẩm.
3. Seller có thể thực hiện các thao tác:
   * Thêm sản phẩm mới với đầy đủ thông tin (tên, mô tả, giá, hình ảnh, danh mục, v.v.).
   * Chỉnh sửa thông tin sản phẩm hiện có.
   * Xóa sản phẩm không còn bán.
4. Hệ thống cập nhật thông tin sản phẩm vào cơ sở dữ liệu.
5. Seller nhận thông báo về trạng thái thay đổi thành công.

**5. Luồng phụ:**

* Nếu thiếu thông tin bắt buộc khi thêm/sửa sản phẩm, hệ thống hiển thị lỗi.
* Nếu sản phẩm đang có đơn hàng đang xử lý, seller không thể xóa.

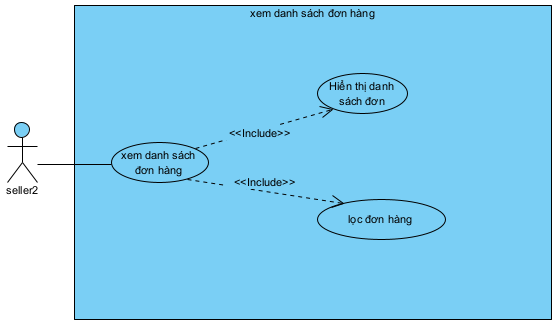
**6. Điều kiện tiên quyết:**

* Seller đã đăng nhập vào hệ thống.
* Seller có quyền quản lý sản phẩm.

**7. Điều kiện sau khi thực hiện:**

* Danh sách sản phẩm của seller được cập nhật trên nền tảng.
* Người mua có thể thấy sản phẩm mới hoặc các thay đổi về sản phẩm.

**Use Case: Xem danh sách đơn hàng**

****

**1. Tên Use Case: Xem danh sách đơn hàng**

**2. Mô tả: Seller có thể xem các đơn hàng mà khách hàng đã đặt để xử lý giao hàng.**

**3. Actor: Seller**

**4. Luồng chính:**

1. Seller đăng nhập vào hệ thống.
2. Seller truy cập vào trang danh sách đơn hàng.
3. Hệ thống hiển thị danh sách đơn hàng, bao gồm trạng thái, thông tin khách hàng, sản phẩm đã đặt.
4. Seller có thể lọc đơn hàng theo trạng thái (chờ xác nhận, đang giao, đã hoàn thành).

**5. Luồng phụ:**

* Nếu không có đơn hàng nào, hệ thống hiển thị thông báo "Không có đơn hàng nào".

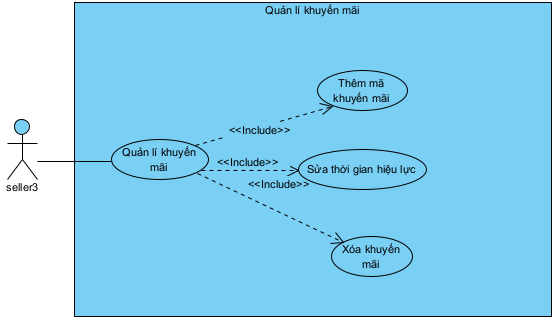
**6. Điều kiện tiên quyết:**

* Seller đã đăng nhập vào hệ thống.
* Có ít nhất một đơn hàng từ khách hàng.

**7. Điều kiện sau khi thực hiện:**

* Seller có thể xem các đơn hàng để xử lý giao hàng.

**Use Case: Quản lý khuyến mãi**

****

**1. Tên Use Case: Quản lý khuyến mãi**

**2. Mô tả: Seller có thể tạo, chỉnh sửa và xóa các chương trình khuyến mãi áp dụng cho sản phẩm của họ.**

**3. Actor: Seller**

**4. Luồng chính:**

1. Seller đăng nhập vào hệ thống.
2. Seller vào trang quản lý khuyến mãi.
3. Seller thực hiện các thao tác:
   * Tạo mã giảm giá hoặc khuyến mãi (ví dụ: giảm giá 10% cho sản phẩm X).
   * Chỉnh sửa thời gian hiệu lực của khuyến mãi.
   * Xóa khuyến mãi nếu không còn áp dụng.
4. Hệ thống lưu lại thông tin và áp dụng khuyến mãi trên sản phẩm.

**5. Luồng phụ:**

* Nếu khuyến mãi hết hạn, hệ thống tự động vô hiệu hóa.

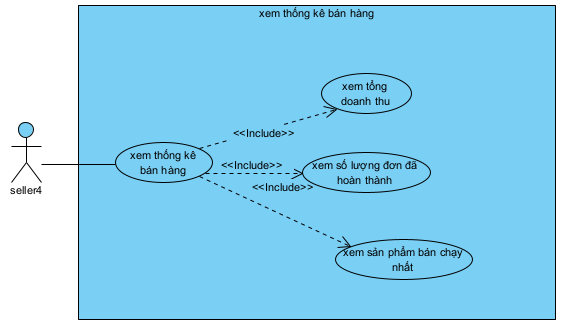
**6. Điều kiện tiên quyết:**

* Seller đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Khuyến mãi được áp dụng trên sản phẩm.
* Khách hàng có thể thấy giá giảm khi mua hàng.

**Use Case: Xem thống kê bán hàng**

****

**1. Tên Use Case: Xem thống kê bán hàng**

**2. Mô tả: Seller có thể xem doanh thu, số lượng đơn hàng và sản phẩm bán chạy.**

**3. Actor: Seller**

**4. Luồng chính:**

1. Seller đăng nhập vào hệ thống.
2. Seller vào trang thống kê bán hàng.
3. Hệ thống hiển thị các thông tin:
   * Tổng doanh thu theo ngày/tháng/năm.
   * Số lượng đơn hàng đã hoàn thành.
   * Sản phẩm bán chạy nhất.
4. Seller có thể lọc theo khoảng thời gian mong muốn.

**5. Luồng phụ:**

* Nếu chưa có đơn hàng nào, hệ thống hiển thị "Chưa có dữ liệu bán hàng".

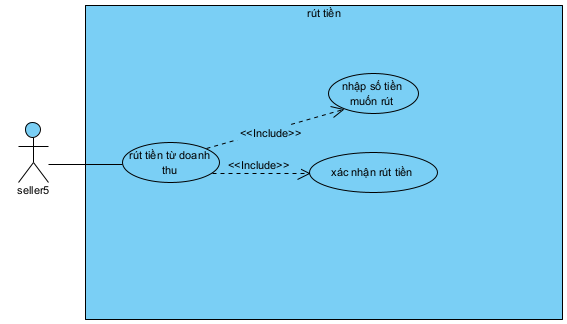
**6. Điều kiện tiên quyết:**

* Seller đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Seller có cái nhìn tổng quan về tình hình kinh doanh.

**Use Case: Rút tiền từ doanh thu**

****

**1. Tên Use Case: Rút tiền từ doanh thu**

**2. Mô tả: Seller có thể rút tiền từ số dư tài khoản doanh thu về tài khoản ngân hàng của họ.**

**3. Actor: Seller**

**4. Luồng chính:**

1. Seller đăng nhập vào hệ thống.
2. Seller vào trang rút tiền.
3. Seller nhập số tiền muốn rút và thông tin tài khoản ngân hàng.
4. Hệ thống xác nhận giao dịch và xử lý chuyển khoản.
5. Seller nhận thông báo giao dịch thành công.

**5. Luồng phụ:**

* Nếu số dư không đủ, hệ thống hiển thị lỗi.
* Nếu tài khoản ngân hàng không hợp lệ, seller phải nhập lại.

**6. Điều kiện tiên quyết:**

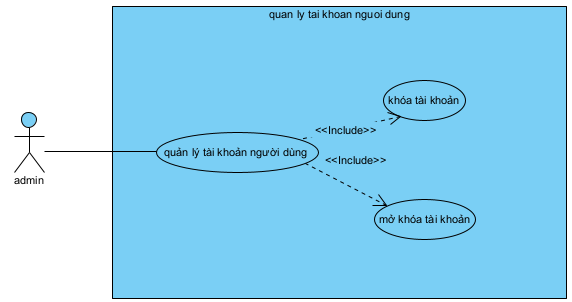
* Seller đã đăng nhập vào hệ thống.
* Seller có số dư trong tài khoản.

**7. Điều kiện sau khi thực hiện:**

* Số dư của seller giảm đi tương ứng.
* Tiền được chuyển vào tài khoản ngân hàng của seller.

Admin

**Use Case: Quản lý tài khoản người dùng**

****

**1. Tên Use Case: Quản lý tài khoản người dùng**

**2. Mô tả: Admin có thể xem danh sách người dùng, khóa/mở khóa tài khoản để đảm bảo an toàn cho hệ thống.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin truy cập trang quản lý người dùng.
3. Admin xem danh sách tài khoản của khách hàng và seller.
4. Admin có thể thực hiện:
   * Khóa tài khoản vi phạm quy định.
   * Mở khóa tài khoản nếu được xác minh lại.
5. Hệ thống lưu lại trạng thái tài khoản.

**5. Luồng phụ:**

* Nếu tài khoản bị khóa, người dùng không thể đăng nhập.
* Nếu mở khóa, người dùng có thể sử dụng tài khoản bình thường.

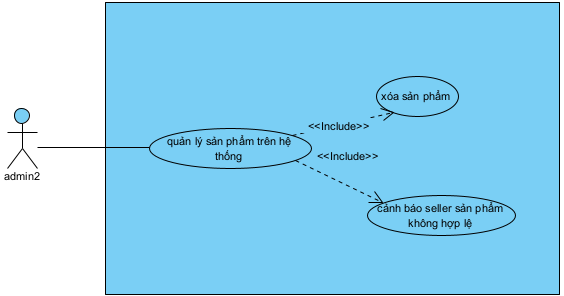
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Trạng thái tài khoản người dùng được cập nhật.

**Use Case: Quản lý sản phẩm trên hệ thống**

****

**1. Tên Use Case: Quản lý sản phẩm trên hệ thống**

**2. Mô tả: Admin có thể kiểm tra sản phẩm được đăng bán để đảm bảo tuân thủ quy định.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang quản lý sản phẩm.
3. Admin xem danh sách sản phẩm của tất cả seller.
4. Admin có thể thực hiện:
   * Xóa sản phẩm vi phạm chính sách.
   * Cảnh báo seller nếu sản phẩm không hợp lệ.
5. Hệ thống cập nhật trạng thái sản phẩm.

**5. Luồng phụ:**

* Nếu sản phẩm bị xóa, khách hàng không thể mua sản phẩm đó.

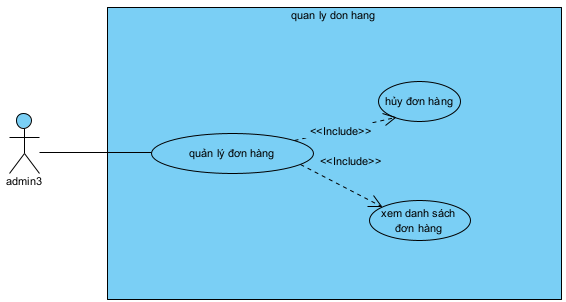
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Danh sách sản phẩm tuân thủ quy định hệ thống.

**Use Case: Quản lý đơn hàng trên hệ thống**

****

**1. Tên Use Case: Quản lý đơn hàng trên hệ thống**

**2. Mô tả: Admin có thể theo dõi và xử lý các vấn đề liên quan đến đơn hàng.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang quản lý đơn hàng.
3. Admin xem danh sách đơn hàng, lọc theo trạng thái.
4. Admin có thể:
   * Hủy đơn hàng nếu phát hiện gian lận.
   * Can thiệp nếu có tranh chấp giữa khách hàng và seller.
5. Hệ thống cập nhật trạng thái đơn hàng.

**5. Luồng phụ:**

* Nếu đơn hàng bị hủy, hệ thống hoàn tiền cho khách hàng.

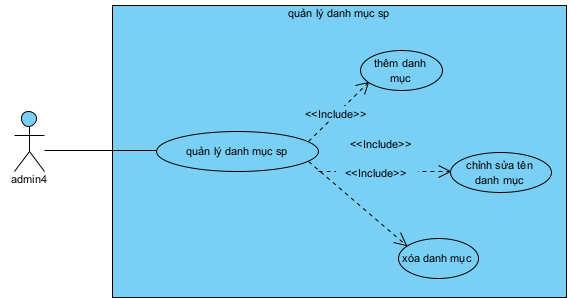
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Đơn hàng được xử lý theo đúng quy định.

**Use Case: Quản lý danh mục sản phẩm**

****

**1. Tên Use Case: Quản lý danh mục sản phẩm**

**2. Mô tả: Admin có thể thêm, sửa, xóa danh mục sản phẩm trên hệ thống.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang quản lý danh mục sản phẩm.
3. Admin có thể thực hiện:
   * Thêm danh mục sản phẩm mới.
   * Chỉnh sửa tên danh mục.
   * Xóa danh mục không còn sử dụng.
4. Hệ thống cập nhật danh mục sản phẩm.

**5. Luồng phụ:**

* Nếu danh mục bị xóa, sản phẩm thuộc danh mục đó sẽ chuyển sang danh mục mặc định.

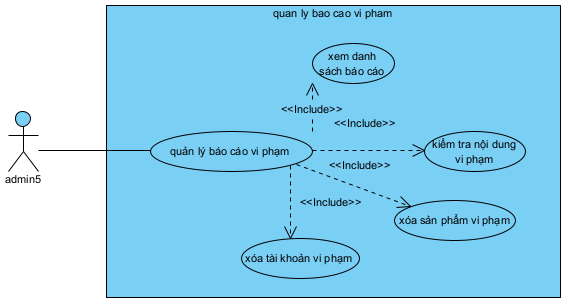
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Hệ thống có danh mục sản phẩm hợp lý và đầy đủ.

**Use Case: Quản lý báo cáo vi phạm**

****

**1. Tên Use Case: Quản lý báo cáo vi phạm**

**2. Mô tả: Admin xem các báo cáo về vi phạm từ người dùng và xử lý khi cần thiết.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang quản lý báo cáo vi phạm.
3. Admin xem danh sách báo cáo từ khách hàng hoặc seller.
4. Admin có thể:
   * Kiểm tra nội dung vi phạm.
   * Xóa sản phẩm/dịch vụ vi phạm.
   * Cảnh báo hoặc khóa tài khoản vi phạm.
5. Hệ thống lưu lại lịch sử xử lý vi phạm.

**5. Luồng phụ:**

* Nếu báo cáo không hợp lệ, admin có thể bỏ qua.

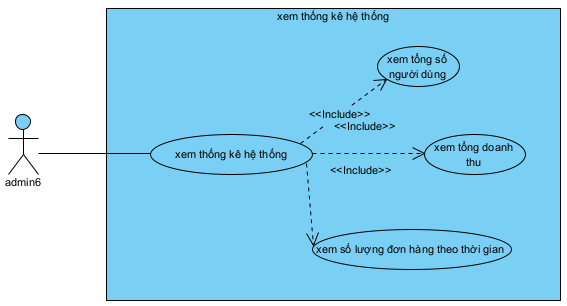
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Vi phạm được xử lý, đảm bảo tính minh bạch và công bằng.

**Use Case: Xem thống kê hệ thống**

****

**1. Tên Use Case: Xem thống kê hệ thống**

**2. Mô tả: Admin có thể theo dõi số lượng người dùng, đơn hàng, doanh thu tổng quan của hệ thống.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang thống kê hệ thống.
3. Hệ thống hiển thị:
   * Tổng số người dùng.
   * Số lượng đơn hàng theo thời gian.
   * Tổng doanh thu của hệ thống.
4. Admin có thể lọc dữ liệu theo ngày/tháng/năm.

**5. Luồng phụ:**

* Nếu hệ thống chưa có dữ liệu, hiển thị "Chưa có thống kê".

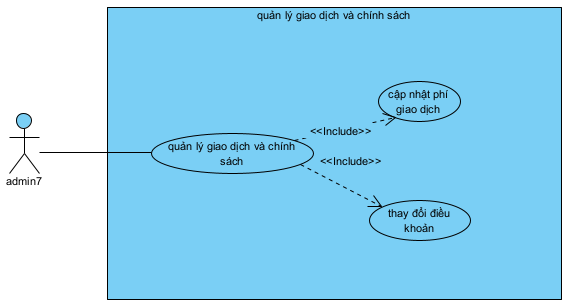
**6. Điều kiện tiên quyết:**

* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Admin có thông tin tổng quan về hoạt động hệ thống.

**Use Case: Quản lý phí giao dịch và chính sách**

****

**1. Tên Use Case: Quản lý phí giao dịch và chính sách**

**2. Mô tả: Admin có thể thay đổi phí giao dịch, chính sách bán hàng trên hệ thống.**

**3. Actor: Admin**

**4. Luồng chính:**

1. Admin đăng nhập vào hệ thống.
2. Admin vào trang quản lý chính sách.
3. Admin có thể:
   * Cập nhật mức phí giao dịch cho seller.
   * Thay đổi điều khoản sử dụng.
4. Hệ thống cập nhật và áp dụng chính sách mới.

**5. Luồng phụ:**

* Nếu có thay đổi lớn, hệ thống gửi thông báo đến tất cả người dùng.

**6. Điều kiện tiên quyết:**

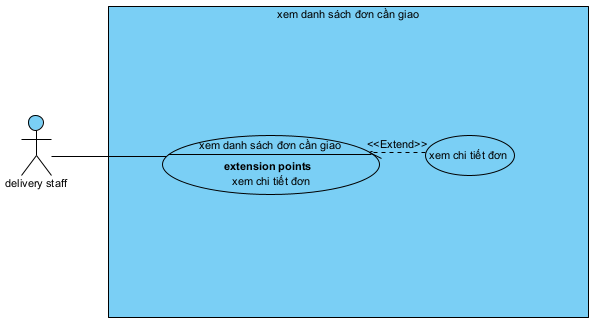
* Admin đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Chính sách mới được áp dụng.

Delivery staff

**Use Case: Xem danh sách đơn hàng cần giao**

****

**1. Tên Use Case: Xem danh sách đơn hàng cần giao**

**2. Mô tả: Nhân viên giao hàng có thể xem danh sách đơn hàng được phân công để giao.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Truy cập trang danh sách đơn hàng cần giao.
3. Hệ thống hiển thị danh sách đơn hàng với thông tin:
   * Mã đơn hàng
   * Địa chỉ giao hàng
   * Tên khách hàng và số điện thoại
   * Trạng thái đơn hàng
4. Delivery Staff chọn đơn hàng và xem chi tiết.

**5. Luồng phụ:**

* Nếu không có đơn hàng mới, hệ thống thông báo "Không có đơn hàng cần giao".

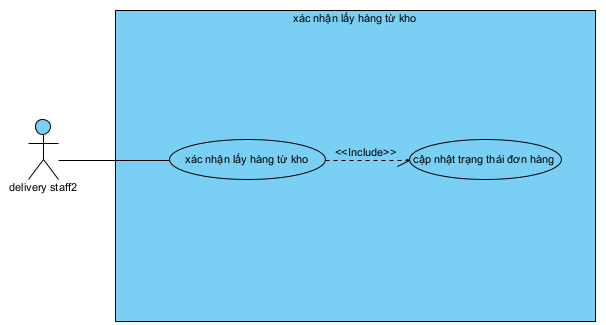
**6. Điều kiện tiên quyết:**

* Delivery Staff đã đăng nhập vào hệ thống.

**7. Điều kiện sau khi thực hiện:**

* Delivery Staff biết danh sách đơn hàng cần giao.

**Use Case: Xác nhận lấy hàng từ kho hoặc người bán**

****

**1. Tên Use Case: Xác nhận lấy hàng từ kho hoặc người bán**

**2. Mô tả: Nhân viên giao hàng xác nhận đã nhận hàng từ kho hoặc từ seller để giao.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Truy cập trang danh sách đơn hàng cần nhận.
3. Chọn đơn hàng và bấm "Đã nhận hàng".
4. Hệ thống cập nhật trạng thái đơn hàng sang **"Đang giao"**.

**5. Luồng phụ:**

* Nếu chưa nhận được hàng, nhân viên có thể báo lỗi cho hệ thống.

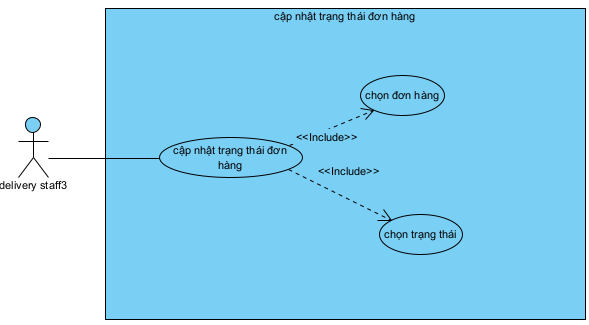
**6. Điều kiện tiên quyết:**

* Đơn hàng đã được duyệt để giao.

**7. Điều kiện sau khi thực hiện:**

* Trạng thái đơn hàng chuyển sang **"Đang giao"**.

**Use Case: Cập nhật trạng thái giao hàng**

****

**1. Tên Use Case: Cập nhật trạng thái giao hàng**

**2. Mô tả: Nhân viên giao hàng có thể cập nhật trạng thái đơn hàng trong quá trình giao.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Vào danh sách đơn hàng đang giao.
3. Chọn đơn hàng và cập nhật trạng thái:
   * **Đang trên đường giao**
   * **Giao hàng không thành công** (khách không nhận, sai địa chỉ, v.v.)
   * **Giao hàng thành công**
4. Hệ thống lưu trạng thái đơn hàng.

**5. Luồng phụ:**

* Nếu khách không nhận hàng, hệ thống yêu cầu nhập lý do.

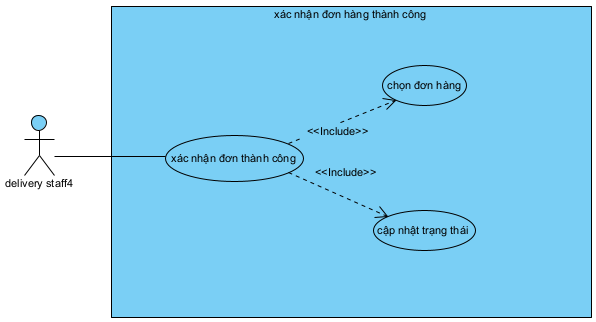
**6. Điều kiện tiên quyết:**

* Đơn hàng đang trong quá trình giao.

**7. Điều kiện sau khi thực hiện:**

* Trạng thái đơn hàng được cập nhật chính xác.

**Use Case: Xác nhận đơn hàng đã giao thành công**

****

**1. Tên Use Case: Xác nhận đơn hàng đã giao thành công**

**2. Mô tả: Nhân viên giao hàng cập nhật trạng thái sau khi giao hàng thành công.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Vào danh sách đơn hàng cần giao.
3. Chọn đơn hàng và bấm "Giao thành công".
4. Hệ thống cập nhật trạng thái đơn hàng sang **"Hoàn thành"**.
5. Khách hàng nhận được thông báo về đơn hàng đã hoàn tất.

**5. Luồng phụ:**

* Nếu khách hàng có phản hồi, nhân viên có thể nhập ghi chú.

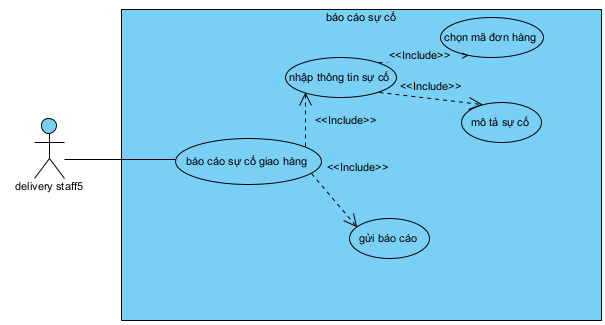
**6. Điều kiện tiên quyết:**

* Đơn hàng đã được giao cho khách hàng.

**7. Điều kiện sau khi thực hiện:**

* Đơn hàng được đánh dấu **"Hoàn thành"** trên hệ thống.

**Use Case: Báo cáo sự cố giao hàng**

****

**1. Tên Use Case: Báo cáo sự cố giao hàng**

**2. Mô tả: Nhân viên giao hàng có thể báo cáo các vấn đề trong quá trình giao hàng.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Truy cập phần "Báo cáo sự cố".
3. Nhập thông tin sự cố:
   * Mã đơn hàng
   * Mô tả sự cố (khách không nhận, địa chỉ sai, hàng bị hư hỏng, v.v.)
4. Gửi báo cáo.
5. Hệ thống gửi thông tin đến bộ phận hỗ trợ để xử lý.

**5. Luồng phụ:**

* Nếu sự cố nghiêm trọng, hệ thống có thể yêu cầu xác minh bổ sung.

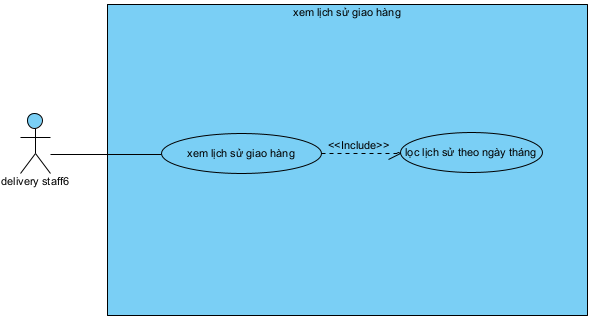
**6. Điều kiện tiên quyết:**

* Nhân viên đang thực hiện giao hàng.

**7. Điều kiện sau khi thực hiện:**

* Hệ thống ghi nhận sự cố để xử lý.

**Use Case: Xem lịch sử giao hàng**

****

**1. Tên Use Case: Xem lịch sử giao hàng**

**2. Mô tả: Nhân viên giao hàng có thể xem lại các đơn hàng đã giao.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Vào phần "Lịch sử giao hàng".
3. Hệ thống hiển thị danh sách đơn hàng đã giao cùng trạng thái:
   * Hoàn thành
   * Hủy do khách không nhận
4. Nhân viên có thể lọc lịch sử theo ngày/tháng.

**5. Luồng phụ:**

* Nếu không có đơn hàng trong lịch sử, hệ thống hiển thị "Chưa có đơn hàng nào".

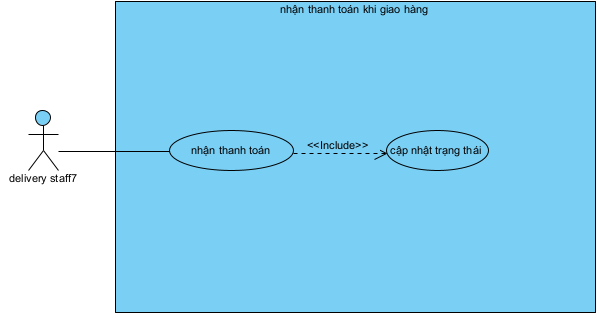
**6. Điều kiện tiên quyết:**

* Nhân viên đã giao ít nhất một đơn hàng.

**7. Điều kiện sau khi thực hiện:**

* Nhân viên có thể xem lại đơn hàng cũ để tham khảo.

**Use Case: Nhận thanh toán khi giao hàng (COD - Cash On Delivery)**

****

**1. Tên Use Case: Nhận thanh toán khi giao hàng (COD)**

**2. Mô tả: Nếu đơn hàng có hình thức thanh toán COD, nhân viên giao hàng sẽ thu tiền từ khách.**

**3. Actor: Delivery Staff**

**4. Luồng chính:**

1. Delivery Staff đăng nhập vào hệ thống.
2. Vào danh sách đơn hàng COD.
3. Giao hàng và thu tiền từ khách.
4. Cập nhật trạng thái đơn hàng: **"Đã thanh toán"**.
5. Hệ thống ghi nhận số tiền đã thu.

**5. Luồng phụ:**

* Nếu khách không có tiền thanh toán, nhân viên có thể cập nhật trạng thái "Không nhận được tiền" và hoàn hàng về kho.

**6. Điều kiện tiên quyết:**

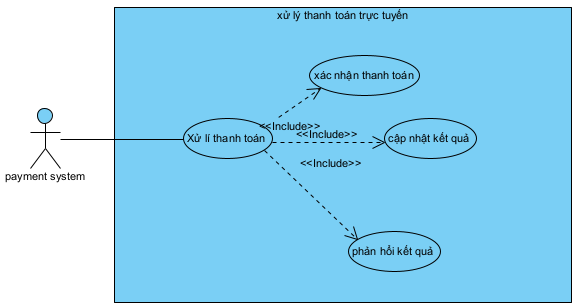
* Đơn hàng có phương thức thanh toán COD.

**7. Điều kiện sau khi thực hiện:**

* Hệ thống ghi nhận thanh toán COD thành công.

Payment system

**Use Case: Xử lý thanh toán trực tuyến**

****

**1. Tên Use Case: Xử lý thanh toán trực tuyến**

**2. Mô tả: Hệ thống thanh toán xử lý các giao dịch thanh toán trực tuyến thông qua thẻ tín dụng, thẻ ghi nợ hoặc ví điện tử.**

**3. Actor: Payment System**

**4. Luồng chính:**

1. Khách hàng chọn phương thức thanh toán trực tuyến.
2. Hệ thống chuyển hướng đến cổng thanh toán (VD: PayPal, Stripe, VNPay).
3. Khách hàng nhập thông tin thanh toán và xác nhận.
4. Cổng thanh toán xử lý giao dịch và phản hồi kết quả.
5. Hệ thống cập nhật trạng thái đơn hàng:
   * **Thành công**: Đơn hàng được xử lý tiếp.
   * **Thất bại**: Khách hàng được yêu cầu thử lại hoặc chọn phương thức khác.

**5. Luồng phụ:**

* Nếu giao dịch thất bại, hệ thống hiển thị thông báo lỗi.
* Nếu có lỗi kết nối với cổng thanh toán, hệ thống thông báo khách hàng thử lại.

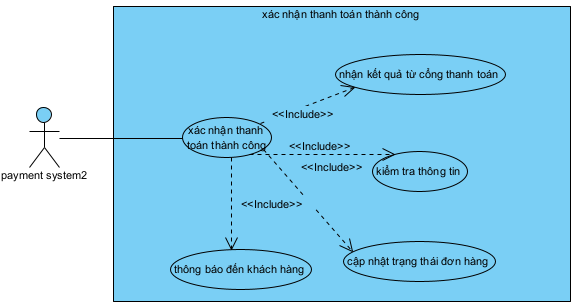
**6. Điều kiện tiên quyết:**

* Khách hàng đã thêm sản phẩm vào giỏ hàng và tiến hành thanh toán.

**7. Điều kiện sau khi thực hiện:**

* Nếu thanh toán thành công, đơn hàng được xác nhận.

**Use Case: Xác nhận thanh toán thành công**

****

**1. Tên Use Case: Xác nhận thanh toán thành công**

**2. Mô tả: Sau khi giao dịch thành công, hệ thống cập nhật trạng thái thanh toán và đơn hàng.**

**3. Actor: Payment System**

**4. Luồng chính:**

1. Cổng thanh toán gửi xác nhận giao dịch thành công.
2. Hệ thống kiểm tra thông tin giao dịch.
3. Cập nhật trạng thái đơn hàng thành "Đã thanh toán".
4. Gửi thông báo xác nhận thanh toán đến khách hàng.

**5. Luồng phụ:**

* Nếu có lỗi khi cập nhật trạng thái, hệ thống gửi cảnh báo đến admin.

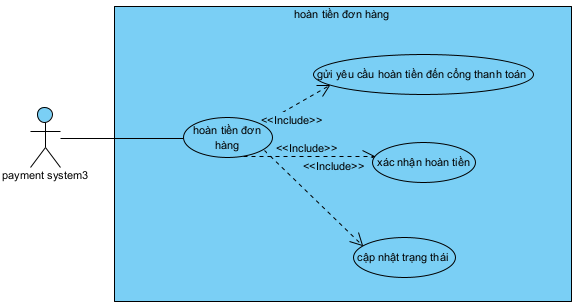
**6. Điều kiện tiên quyết:**

* Thanh toán đã được thực hiện và xác nhận từ cổng thanh toán.

**7. Điều kiện sau khi thực hiện:**

* Đơn hàng được xử lý tiếp theo (giao hàng hoặc chuẩn bị hàng).

**Use Case: Hoàn tiền đơn hàng**

****

**1. Tên Use Case: Hoàn tiền đơn hàng**

**2. Mô tả: Hệ thống xử lý yêu cầu hoàn tiền khi khách hàng hủy đơn hoặc trả hàng.**

**3. Actor: Payment System, Customer, Admin**

**4. Luồng chính:**

1. Khách hàng gửi yêu cầu hoàn tiền.
2. Admin kiểm tra và phê duyệt hoàn tiền.
3. Hệ thống gửi yêu cầu hoàn tiền đến cổng thanh toán.
4. Cổng thanh toán xử lý hoàn tiền và gửi phản hồi.
5. Hệ thống cập nhật trạng thái đơn hàng thành **"Đã hoàn tiền"**.

**5. Luồng phụ:**

* Nếu hoàn tiền thất bại, hệ thống gửi cảnh báo đến admin.

**6. Điều kiện tiên quyết:**

* Khách hàng đã thanh toán trước đó.

**7. Điều kiện sau khi thực hiện:**

* Khách hàng nhận lại tiền, đơn hàng bị hủy hoặc cập nhật trạng thái.

**Use Case: Kiểm tra trạng thái thanh toán**

**1. Tên Use Case: Kiểm tra trạng thái thanh toán**

**2. Mô tả: Hệ thống kiểm tra trạng thái thanh toán của một đơn hàng.**

**3. Actor: Payment System, Admin**

**4. Luồng chính:**

1. Admin hoặc hệ thống yêu cầu kiểm tra trạng thái thanh toán.
2. Hệ thống truy vấn cổng thanh toán để lấy thông tin giao dịch.
3. Cập nhật trạng thái đơn hàng nếu có thay đổi.

**5. Luồng phụ:**

* Nếu không lấy được trạng thái, hệ thống thử lại sau hoặc gửi cảnh báo.

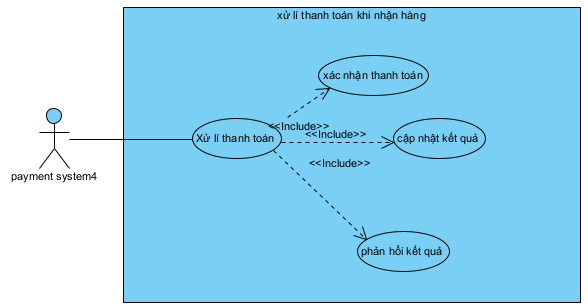
**6. Điều kiện tiên quyết:**

* Đơn hàng đã được tạo và có giao dịch thanh toán.

**7. Điều kiện sau khi thực hiện:**

* Hệ thống đảm bảo trạng thái thanh toán luôn đúng.

**Use Case: Xử lý thanh toán khi nhận hàng (COD - Cash On Delivery)**

****

**1. Tên Use Case: Xử lý thanh toán khi nhận hàng (COD)**

**2. Mô tả: Khi khách hàng chọn thanh toán COD, hệ thống theo dõi và xác nhận thanh toán.**

**3. Actor: Payment System, Delivery Staff**

**4. Luồng chính:**

1. Khách hàng chọn phương thức thanh toán **"Thanh toán khi nhận hàng"**.
2. Đơn hàng được xử lý và giao đi.
3. Nhân viên giao hàng thu tiền từ khách.
4. Nhân viên xác nhận đã thu tiền trên hệ thống.
5. Hệ thống cập nhật trạng thái đơn hàng thành **"Đã thanh toán"**.

**5. Luồng phụ:**

* Nếu khách từ chối thanh toán, đơn hàng bị hủy hoặc hoàn trả.

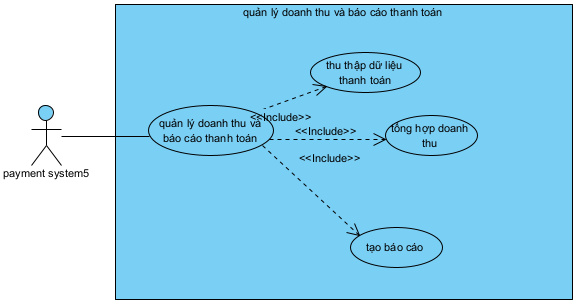
**6. Điều kiện tiên quyết:**

* Đơn hàng đã chọn phương thức thanh toán COD.

**7. Điều kiện sau khi thực hiện:**

* Đơn hàng được ghi nhận đã thanh toán, tiền thu được cập nhật.

**Use Case: Quản lý doanh thu và báo cáo thanh toán**

****

**1. Tên Use Case: Quản lý doanh thu và báo cáo thanh toán**

**2. Mô tả: Hệ thống tự động tổng hợp doanh thu và tạo báo cáo.**

**3. Actor: Payment System, Admin**

**4. Luồng chính:**

1. Hệ thống thu thập dữ liệu thanh toán.
2. Tổng hợp doanh thu theo ngày/tháng/năm.
3. Tạo báo cáo và hiển thị trên bảng điều khiển của admin.

**5. Luồng phụ:**

* Admin có thể lọc theo thời gian, phương thức thanh toán.

**6. Điều kiện tiên quyết:**

* Hệ thống đã ghi nhận nhiều giao dịch thanh toán.

**7. Điều kiện sau khi thực hiện:**

* Admin có thể theo dõi doanh thu chi tiết.

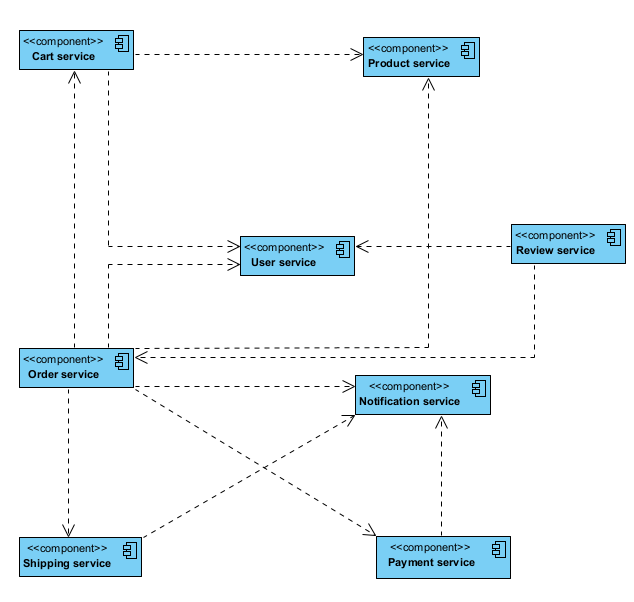
**1.2 Analyze Requirements**

Phân tích yêu cầu giúp chúng ta hiểu rõ cách tổ chức hệ thống thành các dịch vụ nhỏ (microservices) để đảm bảo khả năng mở rộng, bảo trì dễ dàng và hoạt động độc lập.

**1.2.1 Decompose the system into microservices with Django**

Hệ thống thương mại điện tử của chúng ta có thể được phân rã thành các **microservices** độc lập, mỗi service chịu trách nhiệm cho một chức năng cụ thể. Mỗi microservice có thể được xây dựng bằng **Django REST Framework (DRF)** để cung cấp API cho các thành phần khác của hệ thống.

**1. Kiến trúc tổng quan của hệ thống**

****

Hệ thống sẽ được chia thành các microservices chính:

1. **User Service** (Quản lý người dùng)
2. **Product Service** (Quản lý sản phẩm)
3. **Order Service** (Quản lý đơn hàng)
4. **Cart Service** (Quản lý giỏ hàng)
5. **Payment Service** (Xử lý thanh toán)
6. **Shipping Service** (Quản lý giao hàng)
7. **Notification Service** (Gửi thông báo)
8. **Review Service** (Đánh giá sản phẩm)

Mỗi microservice sẽ có **cơ sở dữ liệu riêng** và giao tiếp với nhau thông qua API **RESTful**

**2. Chi tiết từng Microservice**

**2.1 User Service (Dịch vụ Quản lý Người Dùng)**

* **Chức năng**:
  + Đăng ký, đăng nhập (JWT Authentication)
  + Quản lý thông tin người dùng (Customer, Seller, Admin, Delivery Staff)
  + Cấp quyền và xác thực
* **Công nghệ**:
  + Django REST Framework (DRF)
  + PostgreSQL (hoặc MongoDB)
  + JWT (JSON Web Token) để xác thực

**2.2 Product Service (Dịch vụ Quản lý Sản phẩm)**

* **Chức năng**:
  + Người bán có thể thêm, sửa, xóa sản phẩm
  + Hiển thị danh sách sản phẩm cho khách hàng
  + Tìm kiếm và lọc sản phẩm theo danh mục, giá, thương hiệu, v.v.
* **Công nghệ**:
  + Django REST Framework
  + PostgreSQL / MongoDB
  + Elasticsearch để tìm kiếm nhanh sản phẩm

**2.3 Order Service (Dịch vụ Quản lý Đơn hàng)**

* **Chức năng**:
  + Xử lý đơn hàng khi khách hàng thanh toán
  + Quản lý trạng thái đơn hàng (Đang xử lý, Đang giao, Đã nhận)
  + Hủy đơn hàng, hoàn tiền
* **Công nghệ**:
  + Django REST Framework
  + PostgreSQL
  + RabbitMQ / Celery để xử lý đơn hàng không đồng bộ

**2.4 Cart Service (Dịch vụ Giỏ hàng)**

* **Chức năng**:
  + Thêm/xóa sản phẩm vào giỏ hàng
  + Tính tổng tiền giỏ hàng
  + Chuyển đổi giỏ hàng thành đơn hàng khi thanh toán
* **Công nghệ**:
  + Django REST Framework
  + Redis (lưu giỏ hàng tạm thời)
  + PostgreSQL (nếu cần lưu lịch sử giỏ hàng)

**2.5 Payment Service (Dịch vụ Thanh toán)**

* **Chức năng**:
  + Kết nối với cổng thanh toán (VNPay, PayPal, Stripe)
  + Xử lý thanh toán online (thẻ tín dụng, ví điện tử)
  + Hỗ trợ thanh toán khi nhận hàng (COD)
  + Hoàn tiền nếu đơn hàng bị hủy
* **Công nghệ**:
  + Django REST Framework
  + Celery để xử lý giao dịch không đồng bộ
  + Bảo mật với **OAuth2 / API Key**

**2.6 Shipping Service (Dịch vụ Giao hàng)**

* **Chức năng**:
  + Quản lý vận chuyển đơn hàng
  + Kết nối API với đơn vị giao hàng (Giao Hàng Nhanh, Viettel Post)
  + Cập nhật trạng thái giao hàng theo thời gian thực
* **Công nghệ**:
  + Django REST Framework
  + Webhook để nhận thông tin trạng thái từ nhà vận chuyển

**2.7 Notification Service (Dịch vụ Gửi Thông báo)**

* **Chức năng**:
  + Gửi thông báo email, SMS, hoặc push notification
  + Gửi thông báo về trạng thái đơn hàng, khuyến mãi, ưu đãi
* **Công nghệ**:
  + Django REST Framework
  + Celery / RabbitMQ để xử lý hàng đợi
  + Firebase Cloud Messaging (FCM) cho push notifications

**2.8 Review Service (Dịch vụ Đánh giá sản phẩm)**

* **Chức năng**:
  + Khách hàng có thể đánh giá sản phẩm
  + Hiển thị đánh giá trên trang sản phẩm
  + Lọc đánh giá theo sao
* **Công nghệ**:
  + Django REST Framework
  + PostgreSQL / MongoDB

**3. Lợi ích của kiến trúc Microservices**

Việc phân rã hệ thống thành các microservices giúp:

1. **Dễ mở rộng**: Các dịch vụ có thể phát triển độc lập mà không ảnh hưởng đến toàn bộ hệ thống.
2. **Tăng hiệu suất**: Có thể triển khai riêng lẻ từng dịch vụ trên các máy chủ khác nhau.
3. **Bảo trì dễ dàng**: Nếu một service bị lỗi, chỉ cần sửa chữa nó mà không ảnh hưởng đến phần còn lại.
4. **Tăng độ tin cậy**: Nếu một microservice bị lỗi, các dịch vụ khác vẫn tiếp tục hoạt động.
5. **Tối ưu hóa công nghệ**: Mỗi microservice có thể sử dụng công nghệ phù hợp với chức năng của nó (VD: Redis cho giỏ hàng, Elasticsearch cho tìm kiếm).

**4. Cách triển khai Microservices với Django**

Mỗi microservice sẽ được triển khai dưới dạng một **Django project** độc lập. Ví dụ:

ecommerce-user-service/

├── user\_service/

│ ├── settings.py

│ ├── urls.py

│ ├── models.py

│ ├── views.py

│ ├── serializers.py

│ ├── api/

│ │ ├── authentication.py

│ │ ├── user\_management.py

│ ├── tests.py

│ ├── Dockerfile

│ ├── requirements.txt

│ ├── manage.py

Tương tự, **Product Service, Order Service, Payment Service** cũng sẽ có các Django project riêng. Các microservices sẽ giao tiếp với nhau thông qua **REST API**.

**1.2.2 Classes with Attributes of Service Models**

Trong hệ thống **microservices e-commerce**, mỗi dịch vụ có một tập hợp các **models** để quản lý dữ liệu riêng. Dưới đây là các models chính, tổ chức theo từng service.

**1. Customer Service (Dịch vụ Quản lý Người Dùng)**

Quản lý thông tin người dùng trong hệ thống.



**2. Product Service (Dịch vụ Quản lý Sản phẩm)**

Quản lý sản phẩm do người bán cung cấp.

class Category(models.Model):

name = models.CharField(max\_length=100, unique=True)

description = models.TextField(null=True, blank=True)

class Product(models.Model):

name = models.CharField(max\_length=255)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.PositiveIntegerField()

seller = models.ForeignKey('User', on\_delete=models.CASCADE, related\_name="products")

category = models.ForeignKey(Category, on\_delete=models.SET\_NULL, null=True)

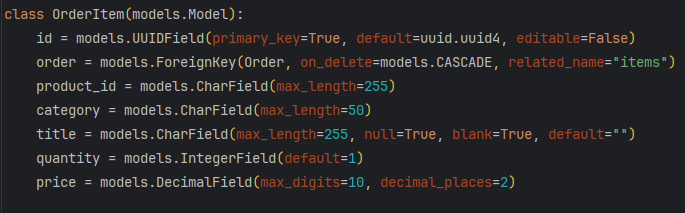
created\_at = models.DateTimeField(auto\_now\_add=True)

* **Category**: Quản lý danh mục sản phẩm.
* **Product**: Mô tả sản phẩm, liên kết với người bán.

**3. Order Service (Dịch vụ Quản lý Đơn hàng)**

Quản lý thông tin đơn hàng.

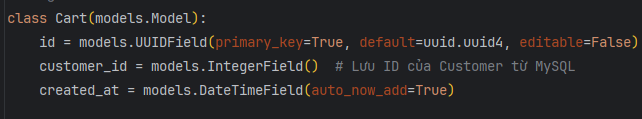


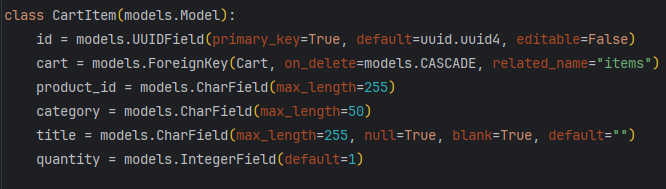


* **Order**: Quản lý đơn hàng của khách hàng.
* **OrderItem**: Liệt kê các sản phẩm trong đơn hàng.

**4. Cart Service (Dịch vụ Giỏ hàng)**

Lưu thông tin giỏ hàng tạm thời.





* **Cart**: Giỏ hàng của khách hàng.
* **CartItem**: Các sản phẩm trong giỏ hàng.

**5. Payment Service (Dịch vụ Thanh toán)**

Quản lý giao dịch thanh toán.

class Payment(models.Model):

PAYMENT\_METHODS = [

('credit\_card', 'Credit Card'),

('paypal', 'PayPal'),

('vnpay', 'VNPay'),

('cod', 'Cash on Delivery'),

]

order = models.OneToOneField('Order', on\_delete=models.CASCADE)

payment\_method = models.CharField(max\_length=20, choices=PAYMENT\_METHODS)

transaction\_id = models.CharField(max\_length=255, unique=True)

status = models.CharField(max\_length=20, default='pending') # pending, success, failed

created\_at = models.DateTimeField(auto\_now\_add=True)

* **Payment**: Lưu thông tin thanh toán của đơn hàng.
* **transaction\_id**: Mã giao dịch từ cổng thanh toán.

**6. Shipping Service (Dịch vụ Giao hàng)**

Quản lý vận chuyển đơn hàng.

class Shipping(models.Model):

order = models.OneToOneField('Order', on\_delete=models.CASCADE)

delivery\_staff = models.ForeignKey('User', on\_delete=models.SET\_NULL, null=True)

tracking\_number = models.CharField(max\_length=100, unique=True)

status = models.CharField(max\_length=20, default='in\_transit') # in\_transit, delivered

estimated\_delivery\_date = models.DateField()

* **Shipping**: Theo dõi quá trình giao hàng.
* **tracking\_number**: Mã vận đơn để theo dõi.

**7. Notification Service (Dịch vụ Gửi Thông báo)**

Gửi thông báo qua email/SMS.

class Notification(models.Model):

user = models.ForeignKey('User', on\_delete=models.CASCADE)

message = models.TextField()

is\_read = models.BooleanField(default=False)

created\_at = models.DateTimeField(auto\_now\_add=True)

* **Notification**: Lưu thông báo gửi đến người dùng.

**8. Review Service (Dịch vụ Đánh giá Sản phẩm)**

Khách hàng có thể đánh giá sản phẩm.

class Review(models.Model):

customer = models.ForeignKey('User', on\_delete=models.CASCADE)

product = models.ForeignKey('Product', on\_delete=models.CASCADE)

rating = models.PositiveSmallIntegerField()

comment = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

* **Review**: Lưu đánh giá của khách hàng cho sản phẩm.

**Tóm tắt kiến trúc Models**

| **Microservice** | **Models Chính** |
| --- | --- |
| **User Service** | User |
| **Product Service** | Category, Product |
| **Order Service** | Order, OrderItem |
| **Cart Service** | Cart, CartItem |
| **Payment Service** | Payment |
| **Shipping Service** | Shipping |
| **Notification Service** | Notification |
| **Review Service** | Review |

**1.2.3 Determine Functions in Services (views)**

Mỗi microservice có một tập hợp các API endpoints để thực hiện các chức năng CRUD và các chức năng chính khác. Django sử dụng **Django REST Framework (DRF)** để triển khai API. Dưới đây là danh sách các API chính theo từng microservice.

**1. User Service (Dịch vụ Quản lý Người Dùng)**

Chức năng: Quản lý người dùng, đăng ký, đăng nhập, cập nhật thông tin cá nhân.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Đăng ký | POST | /api/users/register/ | Đăng ký tài khoản mới. |
| Đăng nhập | POST | /api/users/login/ | Đăng nhập và nhận token. |
| Lấy danh sách | GET | /api/users/ | Lấy danh sách người dùng. |
| Xem chi tiết | GET | /api/users/{id}/ | Xem thông tin một người dùng. |
| Cập nhật thông tin | PUT/PATCH | /api/users/{id}/ | Cập nhật thông tin người dùng. |
| Xóa tài khoản | DELETE | /api/users/{id}/ | Xóa tài khoản người dùng. |

**2. Product Service (Dịch vụ Quản lý Sản phẩm)**

Chức năng: Quản lý sản phẩm do người bán cung cấp.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Tạo sản phẩm | POST | /api/products/ | Tạo sản phẩm mới. |
| Lấy danh sách | GET | /api/products/ | Lấy danh sách sản phẩm. |
| Xem chi tiết | GET | /api/products/{id}/ | Lấy thông tin sản phẩm cụ thể. |
| Cập nhật sản phẩm | PUT/PATCH | /api/products/{id}/ | Cập nhật thông tin sản phẩm. |
| Xóa sản phẩm | DELETE | /api/products/{id}/ | Xóa sản phẩm. |

**3. Order Service (Dịch vụ Quản lý Đơn hàng)**

Chức năng: Quản lý đơn hàng của khách hàng.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Tạo đơn hàng | POST | /api/orders/ | Tạo một đơn hàng mới. |
| Lấy danh sách | GET | /api/orders/ | Lấy danh sách đơn hàng. |
| Xem chi tiết | GET | /api/orders/{id}/ | Xem chi tiết một đơn hàng. |
| Cập nhật trạng thái | PUT/PATCH | /api/orders/{id}/ | Cập nhật trạng thái đơn hàng. |
| Hủy đơn hàng | DELETE | /api/orders/{id}/ | Hủy đơn hàng. |

**4. Cart Service (Dịch vụ Giỏ hàng)**

Chức năng: Quản lý giỏ hàng của khách hàng.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Xem giỏ hàng | GET | /api/cart/ | Xem giỏ hàng hiện tại. |
| Thêm sản phẩm | POST | /api/cart/add/ | Thêm sản phẩm vào giỏ. |
| Cập nhật số lượng | PUT/PATCH | /api/cart/{id}/ | Cập nhật số lượng sản phẩm. |
| Xóa sản phẩm | DELETE | /api/cart/{id}/ | Xóa sản phẩm khỏi giỏ hàng. |

**5. Payment Service (Dịch vụ Thanh toán)**

Chức năng: Xử lý thanh toán.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Thanh toán đơn hàng | POST | /api/payments/ | Xử lý thanh toán. |
| Kiểm tra trạng thái | GET | /api/payments/{id}/ | Kiểm tra trạng thái thanh toán. |

**6. Shipping Service (Dịch vụ Giao hàng)**

Chức năng: Theo dõi đơn hàng và giao hàng.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Tạo thông tin giao hàng | POST | /api/shipping/ | Tạo thông tin giao hàng. |
| Lấy danh sách | GET | /api/shipping/ | Xem danh sách đơn hàng đang giao. |
| Cập nhật trạng thái | PUT/PATCH | /api/shipping/{id}/ | Cập nhật trạng thái giao hàng. |

**7. Notification Service (Dịch vụ Gửi Thông báo)**

Chức năng: Gửi thông báo cho người dùng.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Lấy thông báo | GET | /api/notifications/ | Xem thông báo của người dùng. |
| Đánh dấu đã đọc | PATCH | /api/notifications/{id}/ | Đánh dấu thông báo đã đọc. |

**8. Review Service (Dịch vụ Đánh giá Sản phẩm)**

Chức năng: Khách hàng đánh giá sản phẩm.

| **Function** | **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- | --- |
| Tạo đánh giá | POST | /api/reviews/ | Viết đánh giá mới. |
| Lấy danh sách | GET | /api/reviews/ | Xem danh sách đánh giá. |
| Xem chi tiết | GET | /api/reviews/{id}/ | Xem một đánh giá cụ thể. |
| Xóa đánh giá | DELETE | /api/reviews/{id}/ | Xóa đánh giá. |

**1.2.4 Determine API Response Structure**

Hệ thống **E-Commerce Microservices** sẽ sử dụng **REST API** với Django REST Framework (**DRF**). Các API sẽ trả về dữ liệu dưới dạng **JSON**, giúp frontend hoặc ứng dụng di động dễ dàng xử lý. Dưới đây là các **mẫu API response** cho từng dịch vụ.

**1. User Service (Quản lý Người Dùng)**

Chức năng: Đăng ký, đăng nhập, quản lý tài khoản.

**1.1 Đăng ký tài khoản (Register)**

**Request (POST /api/users/register/)**

json

CopyEdit

{

"username": "john\_doe",

"email": "john@example.com",

"password": "securepassword"

}

**Response**

json

CopyEdit

{

"id": 1,

"username": "john\_doe",

"email": "john@example.com",

"message": "User registered successfully"

}

**1.2 Đăng nhập (Login)**

**Request (POST /api/users/login/)**

json

CopyEdit

{

"username": "john\_doe",

"password": "securepassword"

}

**Response**

json

CopyEdit

{

"token": "abcd1234efgh5678",

"user": {

"id": 1,

"username": "john\_doe",

"email": "john@example.com"

}

}

**2. Product Service (Quản lý Sản phẩm)**

Chức năng: Hiển thị danh sách, thêm, sửa, xóa sản phẩm.

**2.1 Lấy danh sách sản phẩm**

**Request (GET /api/products/)**

json

CopyEdit

{

"page": 1,

"limit": 10

}

**Response**

json

CopyEdit

{

"count": 100,

"next": "/api/products/?page=2",

"previous": null,

"results": [

{

"id": 1,

"name": "Laptop Dell XPS 13",

"price": 1200.99,

"stock": 10,

"description": "Laptop hiệu năng cao cho lập trình viên.",

"image": "https://example.com/images/dellxps.jpg"

},

{

"id": 2,

"name": "iPhone 15 Pro",

"price": 999.99,

"stock": 5,

"description": "Điện thoại cao cấp của Apple.",

"image": "https://example.com/images/iphone15.jpg"

}

]

}

**3. Order Service (Quản lý Đơn hàng)**

Chức năng: Đặt hàng, cập nhật trạng thái đơn hàng.

**3.1 Đặt hàng**

**Request (POST /api/orders/)**

json

CopyEdit

{

"user\_id": 1,

"items": [

{"product\_id": 1, "quantity": 2},

{"product\_id": 2, "quantity": 1}

],

"shipping\_address": "123 Đường ABC, Quận 1, TP. HCM"

}

**Response**

json

CopyEdit

{

"order\_id": 101,

"status": "pending",

"total\_price": 2401.97,

"created\_at": "2025-03-31T12:00:00Z",

"message": "Order placed successfully"

}

**4. Payment Service (Thanh toán)**

Chức năng: Xử lý thanh toán.

**4.1 Thanh toán đơn hàng**

**Request (POST /api/payments/)**

json

CopyEdit

{

"order\_id": 101,

"payment\_method": "credit\_card",

"card\_number": "4111111111111111",

"expiry\_date": "12/27",

"cvv": "123"

}

**Response**

json

CopyEdit

{

"payment\_id": 5001,

"order\_id": 101,

"status": "paid",

"message": "Payment processed successfully"

}

**5. Shipping Service (Giao hàng)**

Chức năng: Theo dõi đơn hàng.

**5.1 Kiểm tra trạng thái giao hàng**

**Request (GET /api/shipping/101/)**

**Response**

json

CopyEdit

{

"order\_id": 101,

"shipping\_status": "out\_for\_delivery",

"expected\_delivery": "2025-04-02",

"tracking\_number": "SHIP123456789"

}

**6. Review Service (Đánh giá sản phẩm)**

Chức năng: Người dùng đánh giá sản phẩm.

**6.1 Gửi đánh giá sản phẩm**

**Request (POST /api/reviews/)**

json

CopyEdit

{

"user\_id": 1,

"product\_id": 1,

"rating": 5,

"comment": "Laptop rất tốt, hiệu năng mạnh!"

}

**Response**

json

CopyEdit

{

"review\_id": 9001,

"product\_id": 1,

"user\_id": 1,

"rating": 5,

"comment": "Laptop rất tốt, hiệu năng mạnh!",

"created\_at": "2025-03-31T12:30:00Z"

}

**Kết luận**

* Thay vì sử dụng **HTML Templates**, hệ thống chỉ sử dụng **REST API** để giao tiếp với frontend hoặc mobile app.
* Mọi dữ liệu được trả về dưới dạng **JSON** để đảm bảo tính linh hoạt.
* Các API endpoints có cấu trúc nhất quán, dễ mở rộng và bảo trì.

**1.2.5 Determine REST API Connecting Services**

**Tổng quan**

Hệ thống **E-Commerce Microservices** sử dụng REST API để giao tiếp giữa các dịch vụ. Mỗi service sẽ hoạt động độc lập và trao đổi dữ liệu với nhau thông qua **HTTP requests** bằng **Django REST Framework (DRF)**.

Các microservices chính trong hệ thống:

1. **User Service** - Quản lý người dùng (authentication, profile).
2. **Product Service** - Quản lý sản phẩm.
3. **Order Service** - Xử lý đơn hàng.
4. **Payment Service** - Xử lý thanh toán.
5. **Shipping Service** - Theo dõi giao hàng.
6. **Review Service** - Đánh giá sản phẩm.

**Cách kết nối REST API giữa các microservices**

Các services liên kết với nhau theo sơ đồ dưới đây:

User Service ----> Order Service ----> Payment Service

| | |

v v v

Product Service Shipping Service Review Service

📌 **Nguyên tắc kết nối:**

* **Giao tiếp qua HTTP Requests:** Sử dụng requests hoặc httpx trong Django để gọi API giữa các microservices.
* **Bảo mật bằng JWT hoặc API Key:** Mọi request phải có token xác thực.
* **Kết nối thông qua Internal API URL:** Các microservices gọi nhau bằng **tên container (Docker) hoặc service name trong Kubernetes**.

**Cách các service gọi REST API giữa nhau**

**1. User Service ↔ Order Service**

**Tình huống:** Khi người dùng đặt hàng, **Order Service** cần lấy thông tin người dùng từ **User Service**.

📌 **Flow:**

1. **Order Service** gọi API /api/users/{user\_id}/ để lấy thông tin người dùng.
2. **User Service** trả về thông tin người dùng dưới dạng JSON.

📌 **Request từ Order Service đến User Service:**

python

CopyEdit

import requests

USER\_SERVICE\_URL = "http://user-service/api/users/"

def get\_user\_info(user\_id):

response = requests.get(f"{USER\_SERVICE\_URL}{user\_id}/")

return response.json()

📌 **API Response từ User Service:**

json

CopyEdit

{

"id": 1,

"username": "john\_doe",

"email": "john@example.com",

"address": "123 Đường ABC, TP.HCM"

}

**2. Order Service → Payment Service**

**Tình huống:** Khi người dùng đặt hàng, hệ thống cần gọi **Payment Service** để xử lý thanh toán.

📌 **Flow:**

1. **Order Service** gửi request đến **Payment Service** với thông tin thanh toán.
2. **Payment Service** xác nhận giao dịch và trả về trạng thái thanh toán.

📌 **Request từ Order Service đến Payment Service:**

python

CopyEdit

PAYMENT\_SERVICE\_URL = "http://payment-service/api/payments/"

def process\_payment(order\_id, amount):

payload = {

"order\_id": order\_id,

"amount": amount,

"payment\_method": "credit\_card"

}

response = requests.post(PAYMENT\_SERVICE\_URL, json=payload)

return response.json()

📌 **API Response từ Payment Service:**

json

CopyEdit

{

"payment\_id": 5001,

"order\_id": 101,

"status": "paid",

"message": "Payment successful"

}

**3. Order Service → Shipping Service**

**Tình huống:** Sau khi đơn hàng được thanh toán thành công, hệ thống gọi **Shipping Service** để tạo yêu cầu giao hàng.

📌 **Flow:**

1. **Order Service** gửi thông tin đơn hàng đến **Shipping Service**.
2. **Shipping Service** tạo vận đơn và trả về mã theo dõi đơn hàng.

📌 **Request từ Order Service đến Shipping Service:**

python

CopyEdit

SHIPPING\_SERVICE\_URL = "http://shipping-service/api/shipping/"

def create\_shipment(order\_id, address):

payload = {

"order\_id": order\_id,

"destination": address

}

response = requests.post(SHIPPING\_SERVICE\_URL, json=payload)

return response.json()

📌 **API Response từ Shipping Service:**

json

CopyEdit

{

"shipment\_id": 7001,

"order\_id": 101,

"status": "processing",

"tracking\_number": "SHIP123456789"

}

**4. Order Service → Product Service**

**Tình huống:** Khi người dùng đặt hàng, hệ thống cần kiểm tra số lượng sản phẩm còn lại.

📌 **Flow:**

1. **Order Service** gọi **Product Service** để lấy thông tin sản phẩm.
2. **Product Service** kiểm tra số lượng tồn kho.

📌 **Request từ Order Service đến Product Service:**

python

CopyEdit

PRODUCT\_SERVICE\_URL = "http://product-service/api/products/"

def check\_product\_availability(product\_id):

response = requests.get(f"{PRODUCT\_SERVICE\_URL}{product\_id}/")

return response.json()

📌 **API Response từ Product Service:**

json

CopyEdit

{

"id": 1,

"name": "Laptop Dell XPS 13",

"price": 1200.99,

"stock": 10

}

**5. Review Service → Product Service**

**Tình huống:** Khi người dùng viết đánh giá, hệ thống cần kiểm tra xem họ có mua sản phẩm hay không.

📌 **Flow:**

1. **Review Service** gọi **Order Service** để kiểm tra lịch sử mua hàng.
2. **Order Service** xác nhận và trả về danh sách sản phẩm đã mua.

📌 **Request từ Review Service đến Order Service:**

python

CopyEdit

ORDER\_SERVICE\_URL = "http://order-service/api/orders/user/"

def check\_purchase\_history(user\_id, product\_id):

response = requests.get(f"{ORDER\_SERVICE\_URL}{user\_id}/")

orders = response.json()

for order in orders:

if product\_id in [item["product\_id"] for item in order["items"]]:

return True

return False

📌 **API Response từ Order Service:**

json

CopyEdit

[

{

"order\_id": 101,

"items": [

{"product\_id": 1, "quantity": 2},

{"product\_id": 2, "quantity": 1}

]

}

]

**Tóm tắt cách các services kết nối qua REST API**

| **Source Service** | **Target Service** | **API Endpoint** | **Mô tả** |
| --- | --- | --- | --- |
| Order Service | User Service | /api/users/{user\_id}/ | Lấy thông tin người dùng. |
| Order Service | Product Service | /api/products/{product\_id}/ | Kiểm tra tồn kho sản phẩm. |
| Order Service | Payment Service | /api/payments/ | Xử lý thanh toán đơn hàng. |
| Order Service | Shipping Service | /api/shipping/ | Tạo yêu cầu giao hàng. |
| Review Service | Order Service | /api/orders/user/{user\_id}/ | Kiểm tra lịch sử mua hàng. |

**1.3 Conclusion**

**Hệ thống E-Commerce Microservices với Django** là một nền tảng thương mại điện tử được chia thành nhiều dịch vụ nhỏ, độc lập, mỗi service đảm nhiệm một chức năng chính. Hệ thống được thiết kế theo kiến trúc microservices với giao tiếp qua REST API, đảm bảo khả năng mở rộng, bảo trì dễ dàng và tích hợp linh hoạt.

**Các thành phần chính:**

1. **User Service:** Quản lý người dùng (Customer, Seller, Admin, Delivery Staff) với các API đăng ký, đăng nhập, quản lý hồ sơ và phân quyền.
2. **Product Service:** Quản lý sản phẩm và danh mục, cho phép người bán thêm, sửa, xóa sản phẩm, và cung cấp API tìm kiếm, lọc sản phẩm.
3. **Order Service:** Xử lý đơn hàng, theo dõi trạng thái, cập nhật thông tin giao dịch, và kết nối với các service khác như Payment và Shipping.
4. **Cart Service:** Quản lý giỏ hàng của khách hàng, hỗ trợ các thao tác thêm, xóa sản phẩm, và chuyển giỏ hàng thành đơn hàng.
5. **Payment Service:** Xử lý giao dịch thanh toán (thẻ, ví điện tử, COD), hoàn tiền và kiểm tra trạng thái giao dịch.
6. **Shipping Service:** Quản lý và theo dõi quá trình giao hàng, tích hợp với các đối tác vận chuyển qua API.
7. **Notification Service:** Gửi thông báo (email, SMS, push) về trạng thái đơn hàng, khuyến mãi, … cho người dùng.
8. **Review Service:** Cho phép khách hàng đánh giá, nhận xét sản phẩm, kèm theo xác thực đơn hàng đã mua.

**Quy trình hoạt động:**  
Các service giao tiếp với nhau qua REST API bằng Django REST Framework. Ví dụ, khi khách hàng đặt hàng, Order Service sẽ lấy thông tin người dùng từ User Service, kiểm tra sản phẩm từ Product Service, sau đó gọi Payment Service để xử lý thanh toán và Shipping Service để tạo yêu cầu giao hàng.

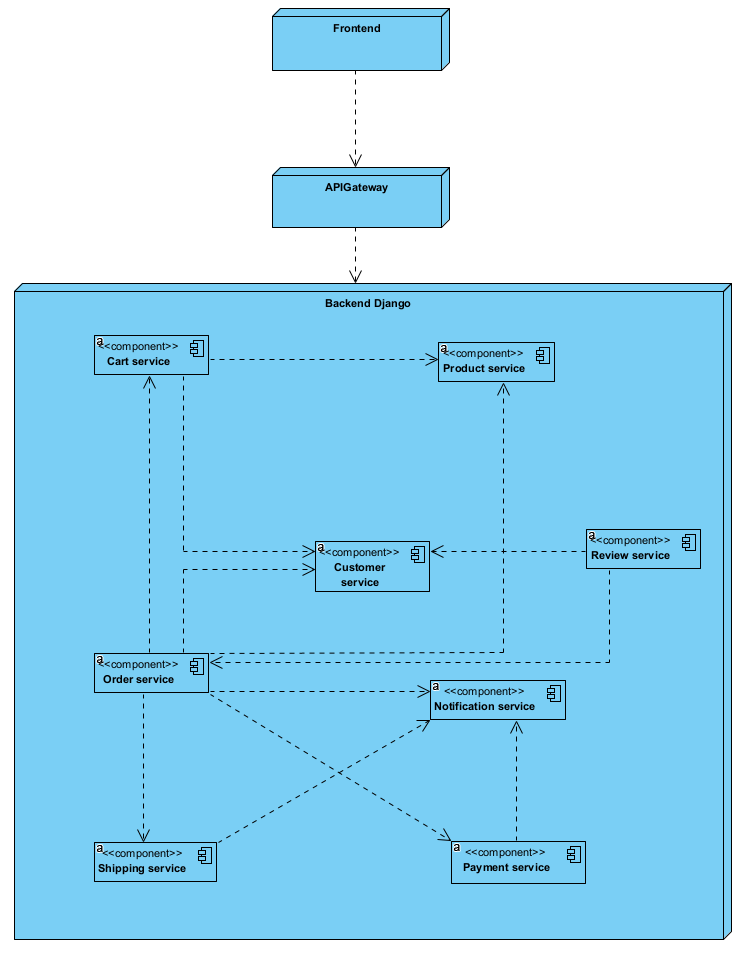
Hệ thống này hướng tới việc xây dựng một nền tảng thương mại điện tử đơn giản, dễ triển khai và mở rộng, phù hợp với sinh viên mới ra trường, đồng thời đáp ứng đầy đủ các yêu cầu chức năng cơ bản của một trang thương mại điện tử.

CHAPTER 2: DESIGN E-COMMERCE SYSTEM WITH MICROSERVICES AND DJANGO

**2.1 Design Services / Components**

**Overview**

Hệ thống e-commerce được thiết kế dựa trên kiến trúc **Microservice**, trong đó mỗi chức năng chính được tách thành một dịch vụ riêng biệt, hoạt động độc lập và giao tiếp với nhau thông qua RESTful APIs. Điều này giúp hệ thống dễ mở rộng, bảo trì và triển khai linh hoạt.



**List of Core Services**

| **Service** | **Chức năng chính** |
| --- | --- |
| **Customer Service** | Quản lý người dùng (Customer, Seller, Admin), xác thực và phân quyền. |
| **Product Service** | Quản lý sản phẩm: thêm, sửa, xóa, hiển thị danh mục sản phẩm. |
| **Cart Service** | Quản lý giỏ hàng cho từng người dùng: thêm/xóa sản phẩm, cập nhật số lượng. |
| **Order Service** | Tạo đơn hàng từ giỏ hàng, xác nhận đơn, kiểm tra trạng thái đơn hàng. |
| **Payment Service** | Tích hợp với cổng thanh toán, xử lý thanh toán đơn hàng và gửi kết quả về Order. |
| **Shipping Service** | Quản lý đơn giao hàng, theo dõi trạng thái và cập nhật trạng thái giao hàng. |
| **Review Service** | Quản lý đánh giá và bình luận của người dùng đối với sản phẩm đã mua. |
| **Notification Service** | Gửi thông báo đến người dùng qua email, SMS, hoặc notification system. |
| **API Gateway** | Là entry point của hệ thống, định tuyến request đến các service tương ứng. |
| **Auth Service** (Tùy chọn) | Tách riêng việc xác thực và cấp JWT token nếu không xử lý trong User Service. |
| **Config/Discovery Server** (Tùy chọn) | Quản lý cấu hình tập trung và tự động phát hiện service (Eureka, Nacos). |

**Component Interaction Flow (Ví dụ: Quy trình đặt hàng)**

1. **Customer** gửi request đến API Gateway.
2. API Gateway chuyển request tới Order Service.
3. Order Service:
   * Gọi Cart Service để lấy thông tin giỏ hàng.
   * Gọi Product Service để kiểm tra tồn kho.
   * Gọi User Service để lấy thông tin khách hàng.
   * Gọi Payment Service để xử lý thanh toán.
   * Gọi Shipping Service để tạo đơn giao hàng.
   * Gọi Notification Service để gửi thông báo xác nhận đơn hàng.

**Security & Authentication**

* Sử dụng **JWT Token** để xác thực người dùng.
* Mỗi request đều đi qua API Gateway, nơi kiểm tra token và xác định role (Customer, Seller, Admin).

**Database per Service**

Mỗi service quản lý cơ sở dữ liệu riêng biệt:

* UserDB, ProductDB, OrderDB, ReviewDB, ...
* Giúp giảm phụ thuộc và tăng khả năng mở rộng (scalability).

**External Integration**

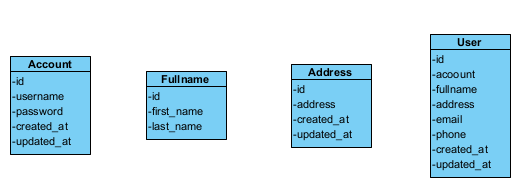
* Payment Service tích hợp với các cổng thanh toán như VNPay, MoMo, Stripe, v.v.
* Notification Service tích hợp với SMTP server, Firebase Cloud Messaging (FCM), hoặc dịch vụ bên thứ ba.

**Ưu điểm của Thiết kế**

* Dễ dàng **triển khai độc lập** từng service.
* **Tăng tính chịu lỗi**: một service có lỗi không ảnh hưởng toàn hệ thống.
* **Dễ mở rộng theo chiều ngang**: có thể scale riêng từng service theo nhu cầu.

2.2 Design classes and methods in component

User service



**1. Service Overview:**

user\_service chịu trách nhiệm quản lý người dùng trong hệ thống E-Commerce, bao gồm việc đăng ký, đăng nhập, cập nhật thông tin cá nhân và xác thực người dùng.

**2. Models (Classes with Attributes)**

**User**

class User(models.Model):

id = models.AutoField(primary\_key=True)

username = models.CharField(max\_length=100, unique=True)

password = models.CharField(max\_length=100)

email = models.EmailField(unique=True)

phone = models.CharField(max\_length=15)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

**Giải thích:** Lưu thông tin cơ bản của người dùng như tên đăng nhập, mật khẩu, email, số điện thoại.

**3. Views (Methods)**

**register\_user(request)**

* Chức năng: Xử lý yêu cầu đăng ký người dùng mới.
* Logic:
  + Nhận dữ liệu username, password, email, phone
  + Kiểm tra trùng lặp username hoặc email
  + Lưu vào cơ sở dữ liệu
* Phản hồi: Trạng thái thành công hoặc lỗi

**login\_user(request)**

* Chức năng: Xác thực thông tin người dùng đăng nhập.
* Logic:
  + Kiểm tra username và password
  + Nếu đúng, trả lại token hoặc thông tin xác thực

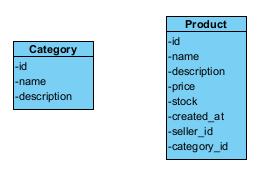
**get\_user\_info(request, user\_id)**

* Chức năng: Lấy thông tin người dùng theo user\_id
* Phản hồi: JSON chứa thông tin người dùng

**update\_user(request, user\_id)**

* Chức năng: Cập nhật thông tin người dùng
* Logic:
  + Cho phép cập nhật các trường như email, phone

Product Service



**1. Service Overview:**

product\_service quản lý toàn bộ danh mục và thông tin sản phẩm trong hệ thống. Các chức năng chính bao gồm:

* Tạo/sửa/xoá sản phẩm.
* Lấy danh sách hoặc chi tiết sản phẩm.
* Phân loại sản phẩm theo danh mục.

**2. Models (Classes with Attributes)**

**Category**

class Category(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=100, unique=True)

description = models.TextField(null=True, blank=True)

**Product**

class Product(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=255)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.PositiveIntegerField()

seller = models.ForeignKey('User', on\_delete=models.CASCADE, related\_name="products")

category = models.ForeignKey(Category, on\_delete=models.SET\_NULL, null=True)

created\_at = models.DateTimeField(auto\_now\_add=True)

**3. Views (Methods)**

**create\_product(request)**

* Chức năng: Tạo mới một sản phẩm
* Logic:
  + Nhận thông tin sản phẩm từ request
  + Liên kết với seller (User)
  + Gán vào danh mục (Category)
  + Lưu vào DB

**update\_product(request, product\_id)**

* Chức năng: Cập nhật thông tin sản phẩm
* Logic:
  + Kiểm tra sản phẩm có tồn tại và có phải của seller đó không
  + Cập nhật thông tin

**delete\_product(request, product\_id)**

* Chức năng: Xoá một sản phẩm

**get\_product(request, product\_id)**

* Chức năng: Lấy chi tiết một sản phẩm

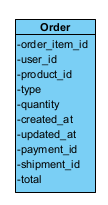
**list\_products(request)**

* Chức năng: Lấy danh sách tất cả sản phẩm

**list\_by\_category(request, category\_id)**

* Chức năng: Lọc sản phẩm theo danh mục

Order Service



**1. Service Overview**

order\_service chịu trách nhiệm xử lý toàn bộ logic liên quan đến đơn hàng trong hệ thống thương mại điện tử. Các chức năng chính gồm:

* Tạo đơn hàng khi khách hàng đặt mua.
* Lưu thông tin chi tiết sản phẩm trong đơn hàng.
* Liên kết với người dùng (customer\_id) và người bán (seller\_id).
* Cập nhật trạng thái đơn hàng (ví dụ: "pending", "shipped", "delivered").
* Xoá hoặc huỷ đơn hàng.

**2. Models (Classes with Attributes)**

| Field | Type | Description |
| --- | --- | --- |
| order\_item\_id | AutoField | Khóa chính tự tăng, định danh mỗi đơn hàng. |
| user\_id | IntegerField | ID người dùng đặt đơn hàng. |
| product\_id | CharField(255) | ID của sản phẩm được đặt. |
| type | CharField(10) | Loại sản phẩm hoặc đơn hàng. |
| quantity | IntegerField | Số lượng sản phẩm đặt mua. |
| created\_at | DateTimeField | Ngày tạo đơn hàng. |
| updated\_at | DateTimeField | Ngày cập nhật đơn hàng lần cuối. |
| payment\_id | IntegerField | ID giao dịch thanh toán liên kết (nếu có). |
| shipment\_id | IntegerField | ID giao hàng liên kết (nếu có). |
| total | IntegerField | Tổng tiền của đơn hàng (nếu tính sẵn). |

Relationships & Notes

* Không dùng khóa ngoại (foreign key) để đảm bảo tính tách biệt giữa các services trong mô hình microservices.
* product\_id, user\_id, payment\_id, shipment\_id đều là integer/char thay vì foreign key – phù hợp với nguyên tắc microservice (liên kết qua ID thay vì quan hệ trực tiếp).
* Có thể sử dụng các REST API để lấy thông tin từ các service liên quan như: UserService, ProductService, PaymentService, ShippingService.

**3. Views (Methods)**

**order\_service/views.py**

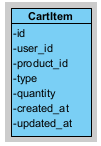
| **Method** | **Description** |
| --- | --- |
| create\_order(request) | Tạo đơn hàng mới dựa trên request từ người dùng |
| get\_all\_orders(request) | Lấy danh sách tất cả đơn hàng (cho seller hoặc customer) |
| get\_order\_by\_id(id) | Lấy thông tin chi tiết của 1 đơn hàng |
| update\_order(request) | Cập nhật đơn hàng (thường là trạng thái) |
| delete\_order(id) | Xoá đơn hàng khỏi hệ thống (nếu cần) |

Các view sử dụng trực tiếp JSON request và response, xử lý CRUD đơn giản.

Cart service

**1. Overview**

**Cart Service** chịu trách nhiệm quản lý giỏ hàng của người dùng. Dịch vụ này cho phép người dùng thêm, cập nhật, xóa sản phẩm trong giỏ và xem nội dung giỏ hàng hiện tại. Đây là một phần quan trọng trong quá trình mua sắm, hỗ trợ người dùng lưu lại sản phẩm trước khi thanh toán.



**Chức năng chính:**

* Thêm sản phẩm vào giỏ hàng
* Cập nhật số lượng sản phẩm
* Xóa sản phẩm khỏi giỏ hàng
* Xem danh sách sản phẩm trong giỏ
* Xóa toàn bộ giỏ hàng

**2. Models**

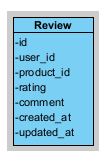
**2.1 Class: CartItem**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | AutoField | Khóa chính tự tăng cho mỗi mục trong giỏ hàng |
| user\_id | IntegerField | ID người dùng sở hữu giỏ hàng |
| product\_id | CharField(255) | ID của sản phẩm được thêm vào giỏ hàng |
| type | CharField(255) | Loại sản phẩm (ví dụ: màu sắc, kích cỡ,...) |
| quantity | IntegerField | Số lượng sản phẩm |
| created\_at | DateTimeField | Thời điểm mục được thêm vào giỏ hàng |
| updated\_at | DateTimeField | Thời điểm mục được cập nhật lần cuối |

**3. Methods (Views)**

| **Method** | **Description** |
| --- | --- |
| add\_to\_cart() | Thêm một sản phẩm mới vào giỏ hàng của người dùng |
| update\_quantity() | Cập nhật số lượng sản phẩm đã có trong giỏ |
| remove\_item() | Xóa một sản phẩm ra khỏi giỏ hàng |
| get\_cart\_items() | Lấy danh sách tất cả sản phẩm trong giỏ hàng của một người dùng |
| clear\_cart() | Xóa toàn bộ giỏ hàng của người dùng |

Review Service



**1. Overview**

**Review Service** chịu trách nhiệm xử lý các đánh giá của người dùng đối với sản phẩm. Dịch vụ này giúp người dùng phản hồi về chất lượng sản phẩm sau khi mua hàng, từ đó tạo sự tin cậy và hỗ trợ những người dùng khác khi chọn mua.

**Chức năng chính:**

* Người dùng viết đánh giá cho sản phẩm.
* Người dùng có thể chỉnh sửa hoặc xóa đánh giá của mình.
* Hệ thống hiển thị danh sách đánh giá theo sản phẩm.
* Hệ thống tính trung bình điểm đánh giá cho từng sản phẩm.

**2. Models**

**2.1 Class: Review**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | AutoField | Khóa chính tự tăng cho mỗi đánh giá |
| user\_id | IntegerField | ID người dùng viết đánh giá |
| product\_id | CharField(255) | ID của sản phẩm được đánh giá |
| rating | IntegerField | Điểm đánh giá (thường từ 1 đến 5) |
| comment | TextField | Nội dung đánh giá |
| created\_at | DateTimeField | Thời điểm đánh giá được tạo |
| updated\_at | DateTimeField | Thời điểm đánh giá được cập nhật lần cuối |

**3. Methods (Views)**

| **Method** | **Description** |
| --- | --- |
| create\_review() | Cho phép người dùng tạo đánh giá mới cho một sản phẩm |
| update\_review() | Cập nhật nội dung và điểm đánh giá của người dùng |
| delete\_review() | Xóa đánh giá của người dùng |
| get\_reviews\_by\_product() | Lấy danh sách đánh giá theo product\_id |
| get\_average\_rating() | Tính trung bình điểm đánh giá cho một sản phẩm |

Payment service

**1. Overview**

**Payment Service** chịu trách nhiệm xử lý các giao dịch thanh toán trong hệ thống thương mại điện tử. Dịch vụ này tiếp nhận yêu cầu thanh toán từ người dùng, cập nhật trạng thái thanh toán của đơn hàng và lưu trữ thông tin liên quan đến các giao dịch.

**Chức năng chính:**

* Tạo yêu cầu thanh toán cho đơn hàng.
* Cập nhật trạng thái thanh toán.
* Truy xuất thông tin thanh toán theo người dùng hoặc đơn hàng.
* Kết nối với bên thứ ba (cổng thanh toán, ví điện tử…).

**2. Models**

**2.1 Class: Payment**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | AutoField | Khóa chính |
| user\_id | IntegerField | ID người dùng thực hiện thanh toán |
| order\_id | IntegerField | ID đơn hàng tương ứng |
| payment\_method | CharField(50) | Phương thức thanh toán (COD, PayPal, VNPAY...) |
| amount | DecimalField | Số tiền thanh toán |
| status | CharField(20) | Trạng thái thanh toán (pending, success, failed) |
| created\_at | DateTimeField | Thời điểm tạo thanh toán |
| updated\_at | DateTimeField | Thời điểm cập nhật thông tin thanh toán gần nhất |

**3. Methods (Views)**

| **Method** | **Description** |
| --- | --- |
| create\_payment() | Tạo yêu cầu thanh toán cho một đơn hàng |
| update\_payment\_status() | Cập nhật trạng thái thanh toán (thường sau callback từ cổng thanh toán) |
| get\_payment\_by\_user() | Lấy tất cả thanh toán theo user\_id |
| get\_payment\_by\_order() | Lấy thông tin thanh toán theo order\_id |

**Class Diagram (ASCII)**

+------------------------------+

| Payment |

+------------------------------+

| - id: AutoField |

| - user\_id: Integer |

| - order\_id: Integer |

| - payment\_method: CharField |

| - amount: DecimalField |

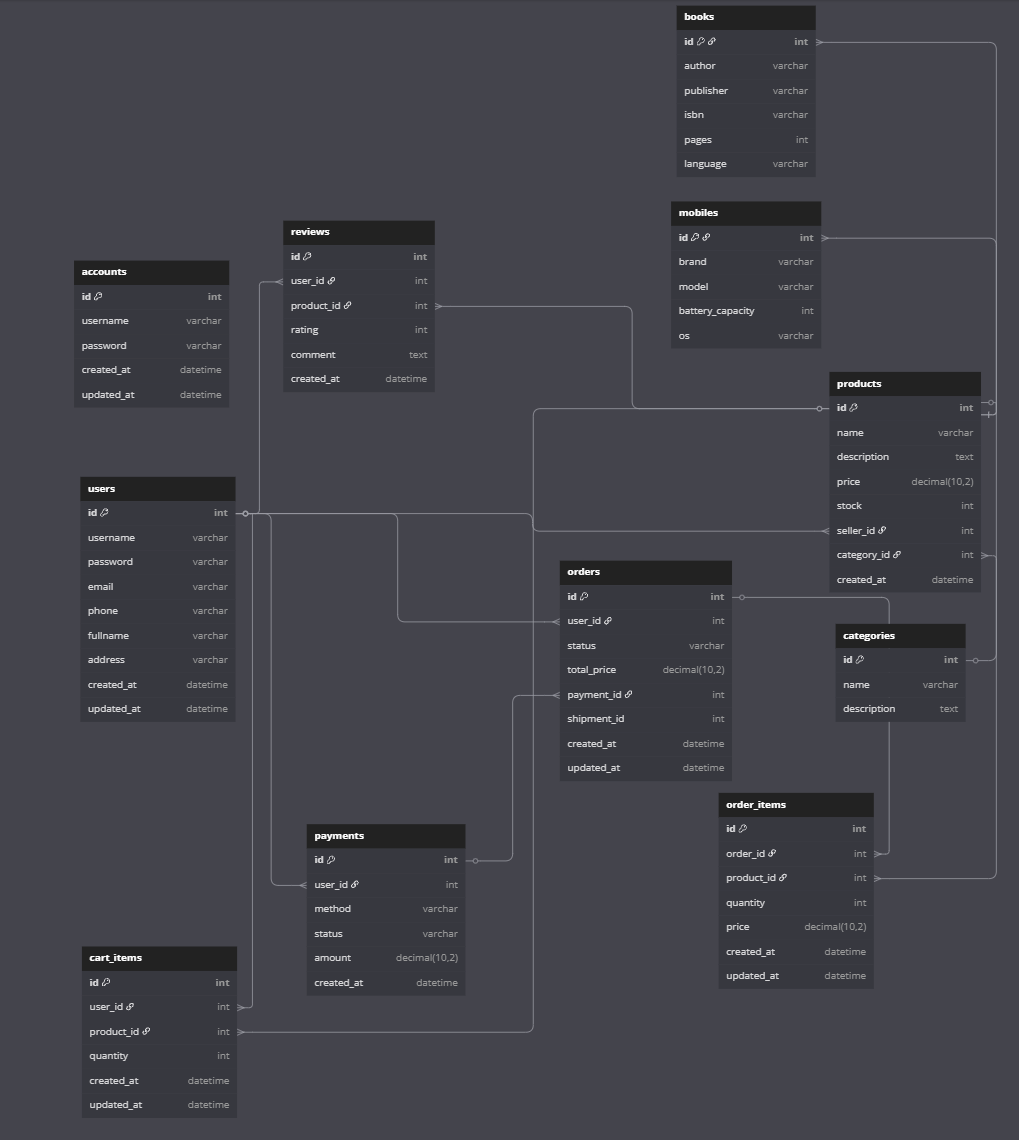
| - status: CharField |

| - created\_at: DateTimeField |

| - updated\_at: DateTimeField |

+------------------------------+

Database



**2.3 DESIGN API**

**1. User Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /users/register | Register new user |
| POST | /users/login | Authenticate and login |
| GET | /users/<id> | Get user profile |
| PUT | /users/<id> | Update user info |
| DELETE | /users/<id> | Delete user account |

**2. Product Service (JTI)**

**🟠 Base Product**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /products/ | List all products |
| POST | /products/ | Create product |
| GET | /products/<id> | Product detail |
| PUT | /products/<id> | Update product |
| DELETE | /products/<id> | Delete product |

**🟠 Categories**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /categories/ | List categories |
| POST | /categories/ | Create category |
| GET | /categories/<id> | Category detail |
| PUT | /categories/<id> | Update category |
| DELETE | /categories/<id> | Delete category |

**🟠 JTI: Book**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /products/books/ | List books |
| POST | /products/books/ | Create book product |
| GET | /products/books/<id> | Book detail |
| PUT | /products/books/<id> | Update book |
| DELETE | /products/books/<id> | Delete book |

**🟠 JTI: Mobile**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /products/mobiles/ | List mobiles |
| POST | /products/mobiles/ | Create mobile product |
| GET | /products/mobiles/<id> | Mobile detail |
| PUT | /products/mobiles/<id> | Update mobile |
| DELETE | /products/mobiles/<id> | Delete mobile |

**3. Order Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /orders/ | List all orders |
| POST | /orders/ | Create new order |
| GET | /orders/<id> | Get order detail |
| PUT | /orders/<id> | Update order (status, etc.) |
| DELETE | /orders/<id> | Cancel order |

**4. Cart Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /cart/items/ | View cart items of current user |
| POST | /cart/items/ | Add item to cart |
| PUT | /cart/items/<id> | Update quantity in cart |
| DELETE | /cart/items/<id> | Remove item from cart |

**5. Review Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /reviews/ | List all reviews |
| POST | /reviews/ | Create new review |
| GET | /reviews/<id> | Review detail |
| PUT | /reviews/<id> | Update review |
| DELETE | /reviews/<id> | Delete review |

**6. Payment Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /payments/ | List payments |
| POST | /payments/ | Initiate new payment |
| GET | /payments/<id> | Payment detail |
| PUT | /payments/<id> | Update payment status |
| DELETE | /payments/<id> | Cancel payment |

**7. Notification Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /notifications/send | Send notification (email/SMS) |
| GET | /notifications/user/<id> | Get user’s notification history |
| PUT | /notifications/<id>/read | Mark notification as read |

**8. Shipping Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /shipping/ | List shipping records |
| POST | /shipping/ | Create shipping for order |
| GET | /shipping/<id> | Shipping detail |
| PUT | /shipping/<id> | Update shipping status |

**2.4 Conclusion**

Trong giai đoạn thiết kế này, hệ thống đã được phân rã thành các **microservice riêng biệt**, mỗi service phụ trách một **chức năng độc lập**, có khả năng mở rộng và tái sử dụng cao. Cụ thể:

* **User Service** quản lý thông tin người dùng, xác thực và phân quyền.
* **Product Service** ứng dụng mô hình **JTI (Joined Table Inheritance)** để hỗ trợ nhiều loại sản phẩm như *Book*, *Mobile*, v.v.
* **Order Service** chịu trách nhiệm xử lý các đơn hàng và liên kết chặt chẽ với **Payment** và **Shipping**.
* **Cart Service** giữ trạng thái tạm thời trước khi tạo đơn hàng.
* **Review Service** hỗ trợ người dùng đánh giá sản phẩm, giúp cải thiện trải nghiệm mua sắm.
* **Payment Service** quản lý quá trình thanh toán, có thể mở rộng ra tích hợp cổng thanh toán bên thứ ba.
* **Notification Service** xử lý các thông báo, email và SMS đến người dùng.
* **Shipping Service** quản lý quá trình vận chuyển và theo dõi trạng thái đơn hàng.

Về phần thiết kế:

* Các **lớp (models)** được xây dựng dựa trên nguyên tắc *Single Responsibility*, dễ bảo trì và mở rộng.
* API của từng service được thiết kế **RESTful**, rõ ràng, trực quan, giúp các service giao tiếp hiệu quả và dễ dàng tích hợp với frontend hoặc các client khác.
* Sự tách biệt rõ ràng giữa các service giúp hệ thống có thể triển khai theo kiến trúc **scalable & fault-tolerant**, sẵn sàng phục vụ khối lượng lớn người dùng.