**Veer Narmad South Gujarat University, Surat.**

**M.Sc. (Information Technology) Programme**

**Project Report**

**9 th Semester**

**M.Sc. (Information Technology)**

**5 Years Integrated Course**

**Year 2024-25**

**Basketeer**

**Guided By : Submitted By :**

**Mr. Pushpal Desai Karan Ramrakhyani(53)**

**(E23110018000670001)**

**Veer Narmad South Gujarat University, Surat.**

**Department of Information and Communication Technology**

**M.Sc. (I.T.) Programme**

**Certificate**

This is to certify that Mr./Miss. **Karan Ramrakhyani** Exam Seat Number: **53 (E23110018000670001)** and have satisfactorily completed his/her project work entitled **Basketeer** as a partial fulfillment of the requirements for **9th** ***Semester - M.Sc. (Information Technology)*** during the academic Year 2024-2025.

Date: 12/12/2023

Place : Surat

**Internal Project Guide Head of the Department**

**M.Sc.(I.T.) 9 th Semester Department of I.C.T.**

**Department of I.C.T. Veer Narmad South Gujarat University,**

**Veer Narmad South Gujarat University, Surat**

**Surat**

# **1. Proposed System**

## 

## **1.1 Scope**

The e-commerce platform aims to provide a seamless shopping experience for users while enabling administrators to manage products and orders efficiently. The platform includes user functionalities such as product browsing, filtering, adding to cart, placing orders, managing addresses, and reviewing products. It also provides admin-specific features like inventory management, order management, and product updates. The system leverages a microservices architecture with robust role-based access control to ensure secure and efficient operations.

## **1.2** **Objective**

The primary objective of the project is to build a scalable, modular, and user-friendly e-commerce platform using a microservices architecture. This system ensures:

* Easy navigation and purchase processes for users.
* Streamlined product and order management for administrators.
* Secure operations through token-based authentication and role-based authorization.
* High availability and scalability by leveraging Kubernetes for container orchestration.

## **1.3 Advantages**

**Scalability**: The microservices-based architecture enables independent scaling of services to handle high traffic and ensure performance under heavy loads.

**Security**: Role-based access control and token authentication provide a secure environment for users and administrators.

**Modularity**: Each service is designed to operate independently, allowing for easy maintenance and upgrades without disrupting the system.

**User Experience**: Features like product filtering, reviews, and streamlined cart management enhance customer satisfaction.

**Efficient Administration**: Admin-specific functionalities, such as inventory and order management, simplify backend operations.

**Future Proofing**: The modular design allows for easy integration of new features and services as the business evolves.

# **2. System Environment**

The e-commerce platform is designed to operate efficiently within the following system environment:

**1. Hardware Requirements**

* **Server**:
  + Minimum: 4 CPU cores, 8GB RAM, 100GB SSD
  + Recommended: 8 CPU cores, 16GB RAM, 250GB SSD
* **Client Devices**:
  + Any device with a modern web browser (desktop, tablet, smartphone).

**2. Software Requirements**

* **Backend**:
  + Java 17
  + Spring Boot 3.x
  + Docker 20.x for containerization
  + Kubernetes 1.25+ for container orchestration
* **Database**:
  + MySQL 8.x hosted in a Docker container
* **Frontend**:
  + React 18.x
  + Node.js 18.x
* **Development Tools**:
  + IntelliJ IDEA / VS Code for coding
  + Postman for API testing
  + Git for version control

**3. Middleware and Services**

* **Authentication and Authorization**:
  + JSON Web Tokens (JWT) for secure access
  + Role-based access control for managing permissions
* **APIs**:
  + RESTful APIs exposed by microservices
* **Payment Integration**:
  + PayPal SDK for payment processing

**4. Deployment Environment**

* **Hosting**:
  + Kubernetes Cluster (cloud-based or on-premise)
* **Load Balancer**:
  + NGINX or cloud-provider-specific services (e.g., AWS ALB, Azure Load Balancer)

**5. Networking**

* **Protocol**:
  + HTTPS for secure communication
* **Ports**:
  + Configurable ports for each microservice, with reverse proxy mapping via API Gateway

**6. Operating System**

* **Server**:
  + Linux (Ubuntu 20.04+ or CentOS 7+)
* **Client**:
  + Any OS capable of running a modern web browser (e.g., Windows, macOS, Android, iOS).

This environment ensures optimal performance, scalability, and security for the e-commerce platform.

# **3. System Planning**

Effective planning is crucial for the successful development and deployment of the e-commerce platform. This section outlines the feasibility analysis and the software engineering model adopted for the project.

## **3.1 Feasibility Study**

The feasibility study evaluates the practicality of implementing the e-commerce platform from technical, operational, and economic perspectives.

**3.1.1 Technical Feasibility**

* **Infrastructure**:  
  The platform utilizes modern tools such as Spring Boot, React, Docker, and Kubernetes, which are widely supported and scalable.
* **Expertise**:  
  The development team possesses expertise in Java, REST APIs, and frontend technologies, ensuring smooth development and deployment.
* **Integration**:  
  The system can seamlessly integrate with PayPal for payments and JWT for secure user authentication.

3.1.2 Operational Feasibility

* The platform supports essential e-commerce functionalities such as product browsing, cart management, order placement, and reviews, meeting user expectations.
* Admin functionalities, like product management and order handling, streamline operations, making it user-friendly for administrators.
* Robust role-based access control ensures that users and admins interact with appropriate services.

**3.1.3 Economic Feasibility**

* **Cost Analysis**:  
  The project minimizes cost through open-source technologies and containerization, reducing infrastructure and licensing expenses.
* **Return on Investment**:  
  The system is scalable and designed to accommodate future business growth, ensuring long-term value.

**3.1.4 Schedule Feasibility**

* The project timeline, from September to December, included all phases from design to deployment, demonstrating that the schedule was achievable.

## **3.2 Software Engineering Model**

The **Agile Development Model** was chosen for this project due to its flexibility, iterative approach, and adaptability to changes.

**3.2.1 Key Features of the Agile Model**

1. **Incremental Development**:  
   Each feature (e.g., user registration, product search, cart management) was developed in small, manageable increments.
2. **Continuous Feedback**:  
   Regular feedback cycles ensured alignment with requirements and timely identification of issues.
3. **Collaboration**:  
   Frequent interactions among developers, testers, and stakeholders ensured the project met its goals.
4. **Testing and Quality Assurance**:  
   Testing was performed at the end of each sprint to ensure that all modules met functionality and performance standards.

**3.2.2 Implementation Phases**

1. **Requirement Analysis**:  
   Gathered user stories and prioritized tasks based on business needs.
2. **Design and Planning**:  
   Created architectural diagrams, class designs, and database schemas.
3. **Development**:  
   Iteratively developed backend microservices, frontend components, and integrated APIs.
4. **Testing**:  
   Conducted unit testing, integration testing, and system testing to ensure system reliability.
5. **Deployment**:  
   Deployed services on a Kubernetes cluster for scalability and fault tolerance.

By adopting the Agile Model, the project remained dynamic, addressed evolving requirements, and achieved a high-quality deliverable.

# **4. System Analysis and Modeling**

This section provides a detailed analysis of the system requirements, focusing on functional and non-functional requirements essential for the development and operation of the e-commerce platform.

## **4.1 Software Requirement Specification (SRS)**

**4.1.1 Functional Requirements**

The functional requirements outline the core features and operations of the e-commerce platform.

1. **User Management**:
   * Users can register and log in using valid credentials.
   * Admins and users can log out securely.
   * Role-based access control ensures that only authorized actions are permitted for each role.
2. **Product Management**:
   * Users can browse, search, and filter products by categories, brands, price, and other criteria.
   * Admins can add, edit, and delete products.
   * Product stock is updated dynamically upon order placement or manual adjustments by admins.
3. **Cart Management**:
   * Users can add, update, and remove products in their cart.
   * The system validates inventory availability before adding products to the cart.
4. **Order Management**:
   * Users can place orders for items in their cart.
   * The system validates stock availability, user details, and address before processing orders.
   * Admins can update the status of orders.
5. **Payment Integration**:
   * PayPal integration ensures secure payment processing for orders.
6. **Address Management**:
   * Users can add, edit, and delete up to three addresses associated with their account.
7. **Product Reviews**:
   * Users can submit reviews and ratings for purchased products.
   * Reviews are linked to the product and the user who submitted them.

**4.1.2 Non-Functional Requirements**

The non-functional requirements ensure that the system performs efficiently, securely, and reliably under various conditions.

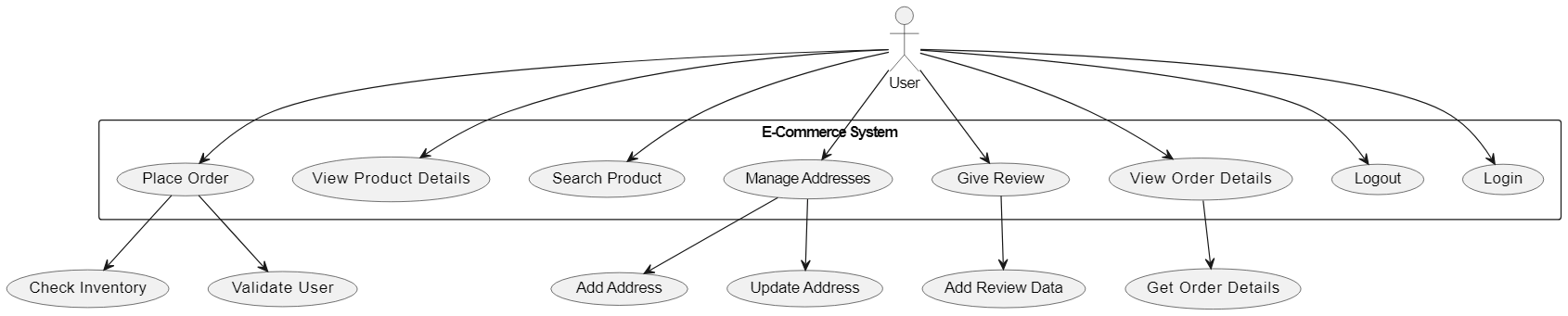
1. **Scalability**:
   * The system is designed using a microservices architecture and Kubernetes to support increased user loads and future expansion.
2. **Security**:
   * All API requests require valid JWT tokens for authentication.
   * Role-based access control restricts unauthorized actions.
   * User passwords are securely hashed before storage.
3. **Reliability**:
   * The system must achieve 99.9% uptime, supported by containerization and clustering.
4. **Maintainability**:
   * Modular design ensures that services can be updated independently without affecting the entire system.
5. **Usability**:
   * The UI must be intuitive and responsive across devices, ensuring a smooth user experience.
6. **Integration**:
   * Seamless integration with third-party systems like PayPal for payments and Docker/Kubernetes for deployment.

By adhering to these functional and non-functional requirements, the e-commerce platform provides a robust, user-friendly, and secure system that meets user and business needs.

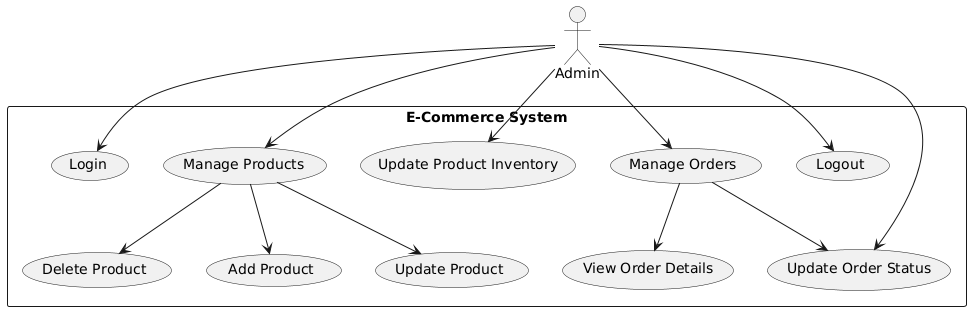
## **4.2 UML Diagrams**

### **4.2.1 Use Case Diagram**

**User Side**

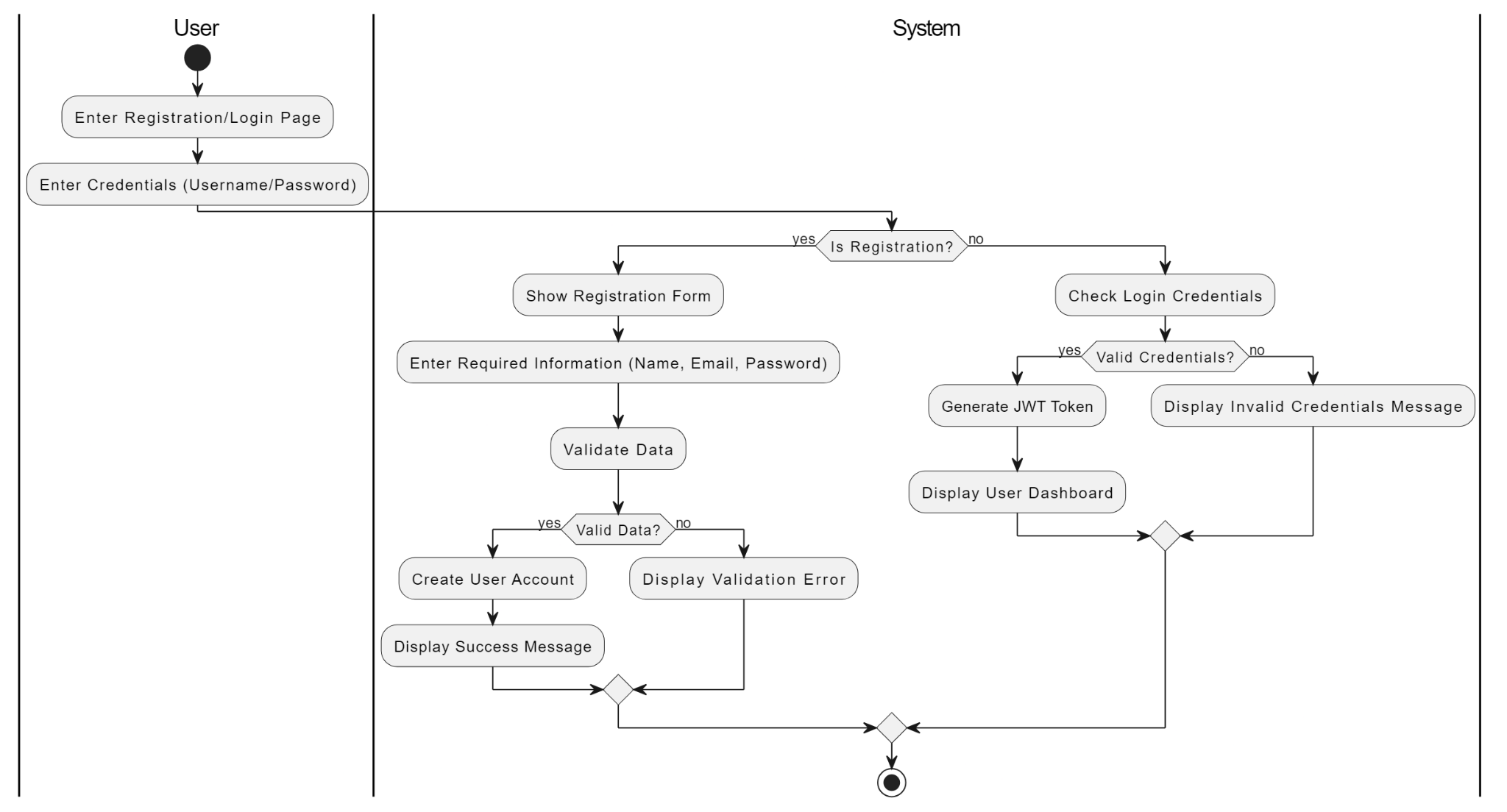
****

**Admin Side**

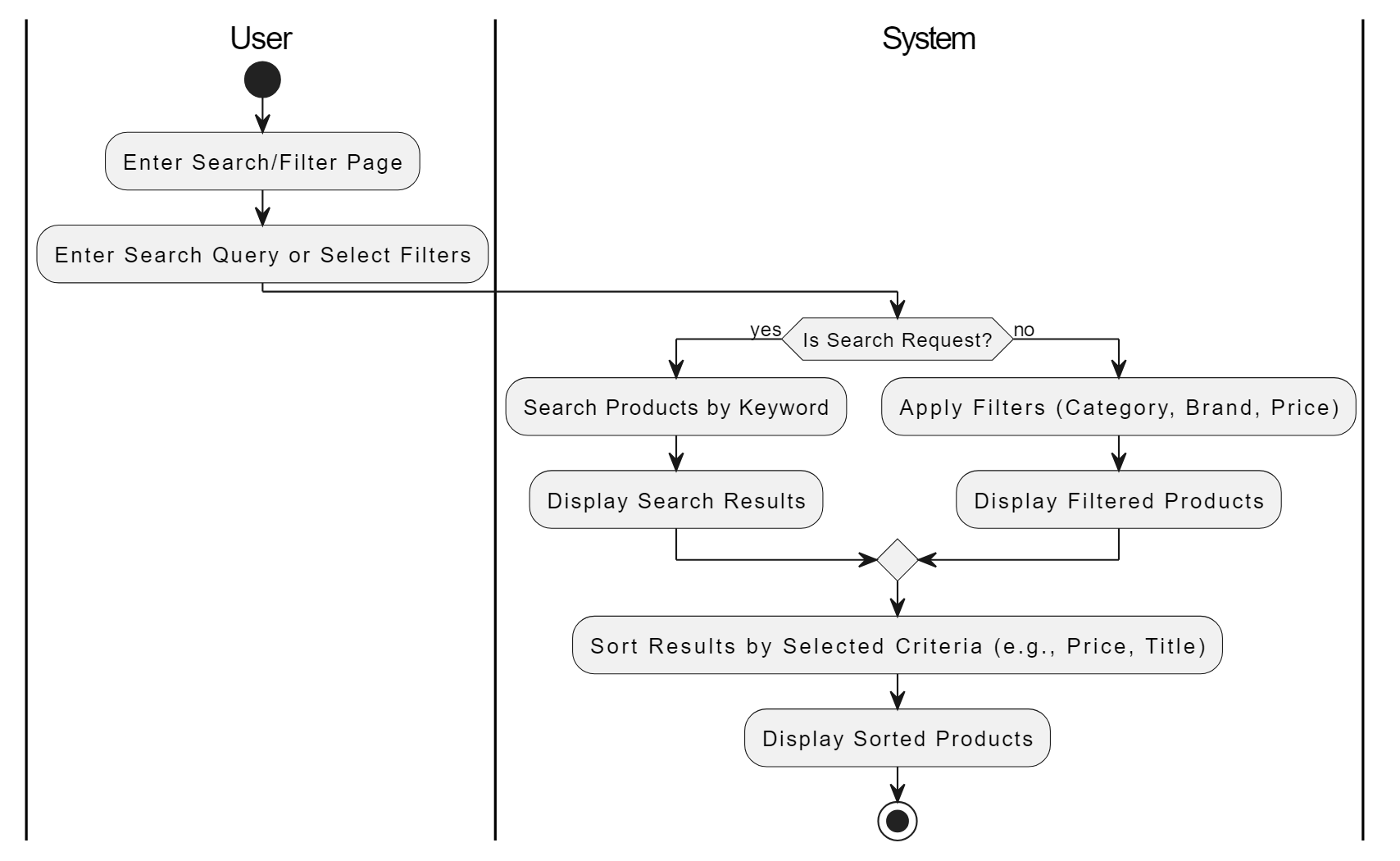
****

### **4.2.2 Activity Diagrams**

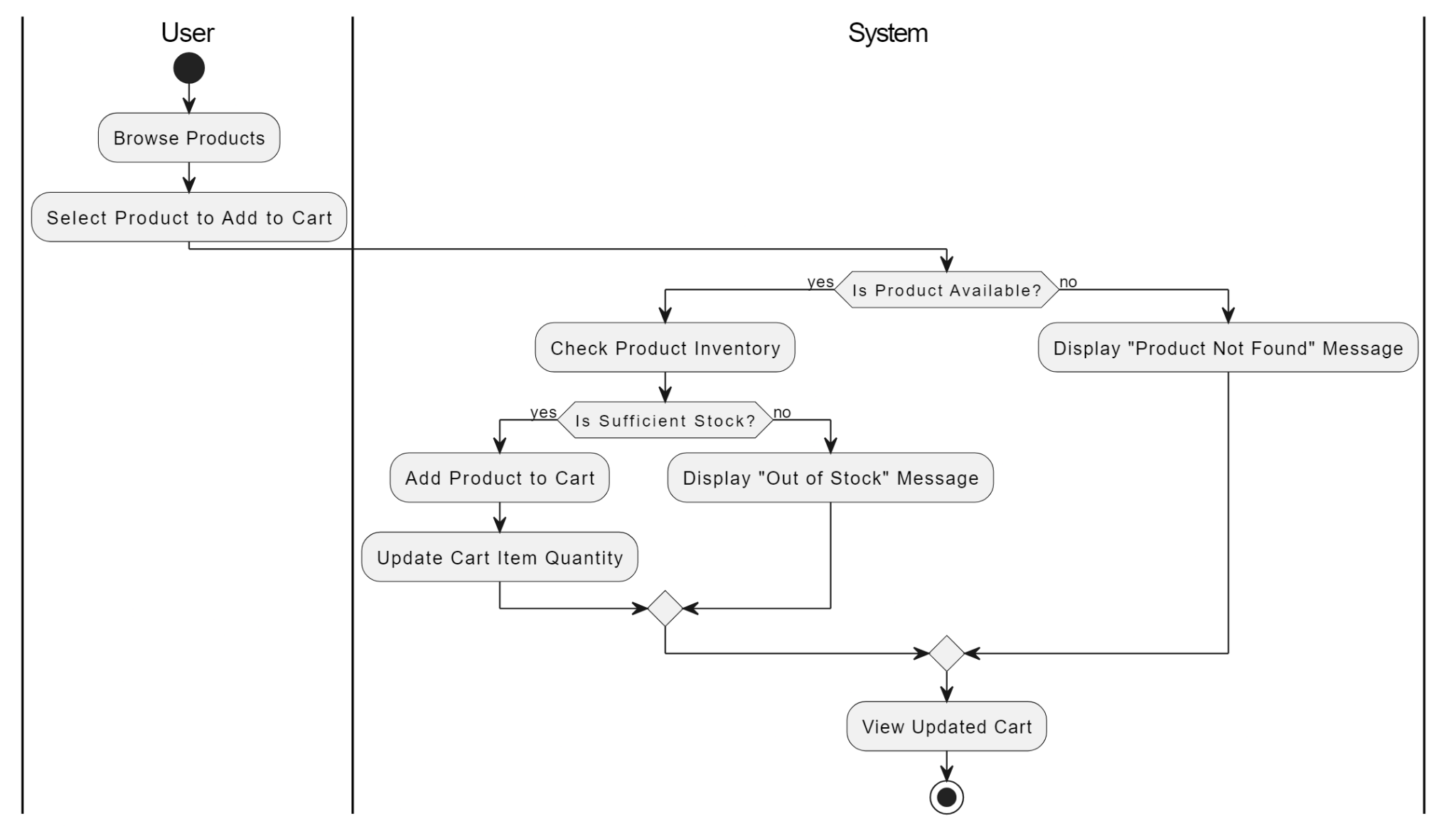
**User Registration and Login**



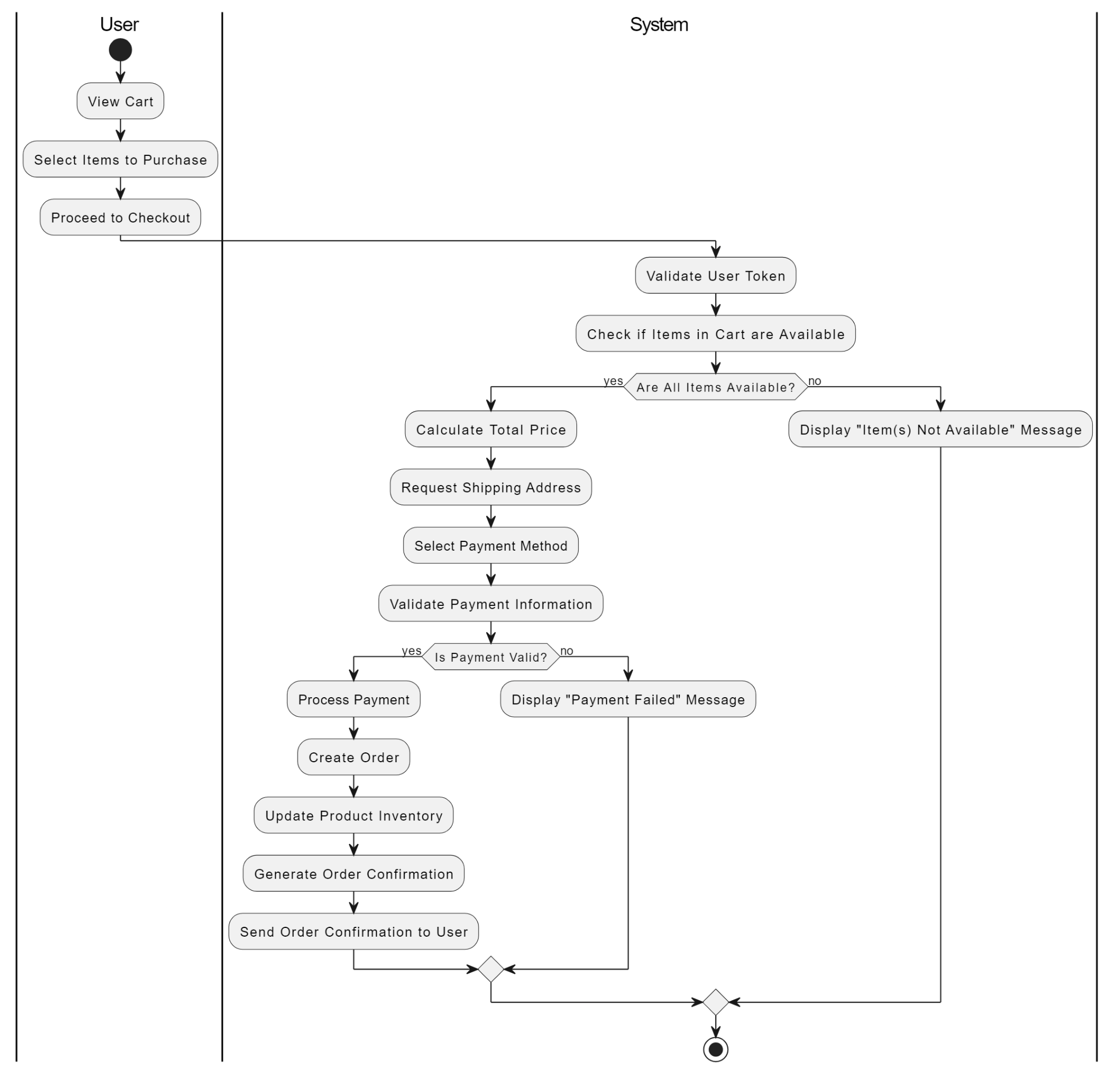
**Product Search and Filtering**

****

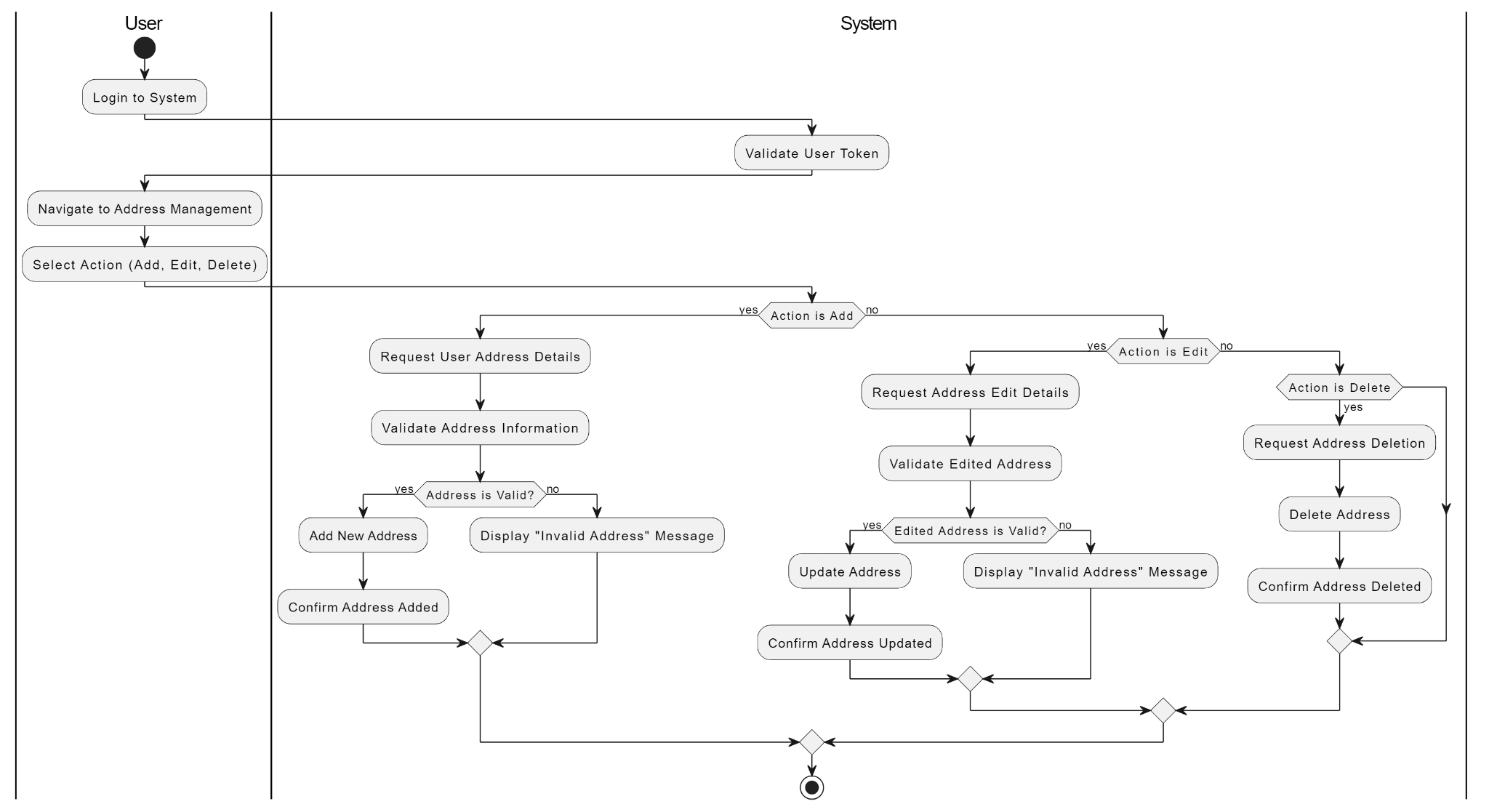
**Adding Products to Cart**

****

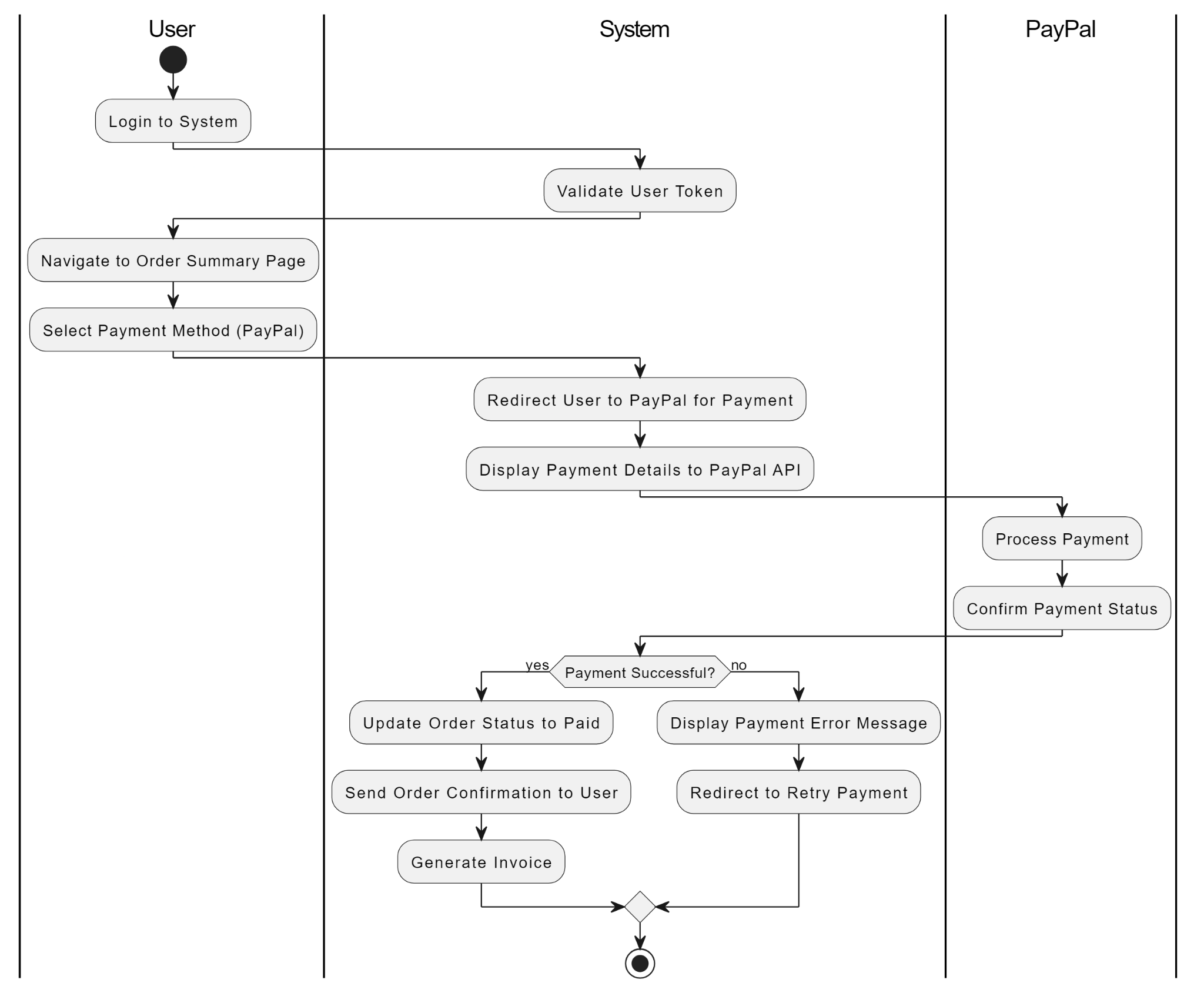
**Placing an Order**

****

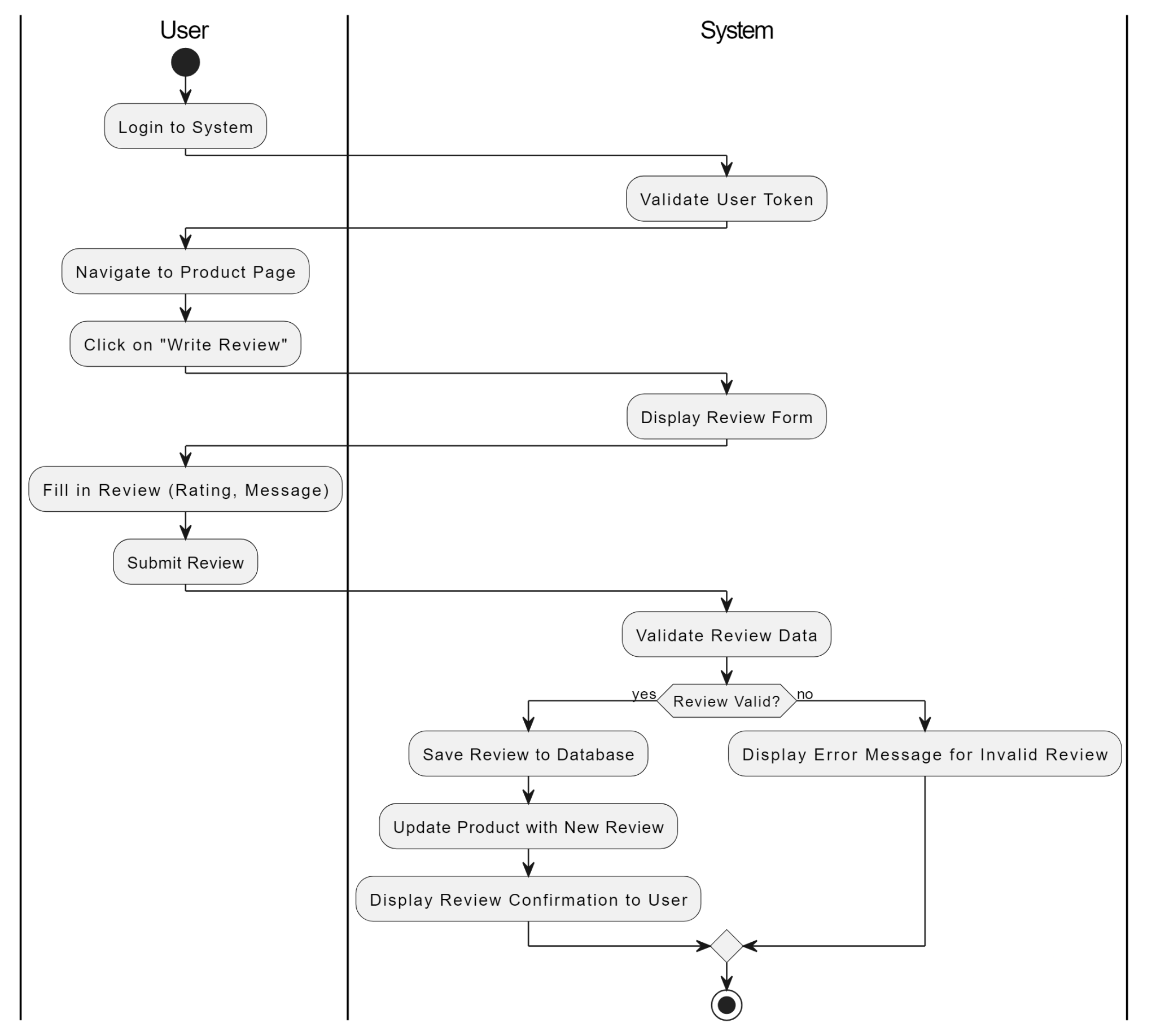
**Address Management**

****

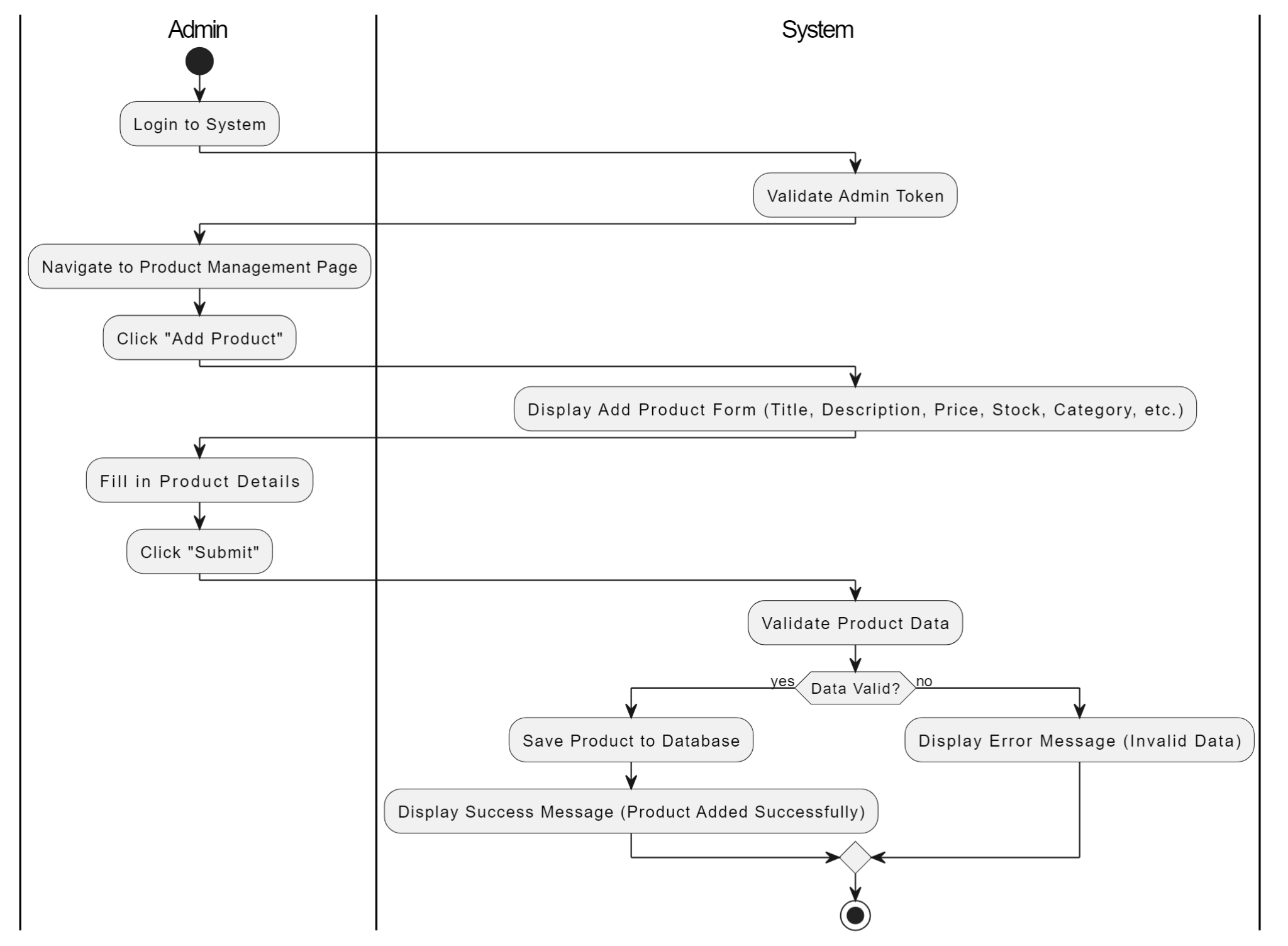
**Order Payment (via PayPal)**

****

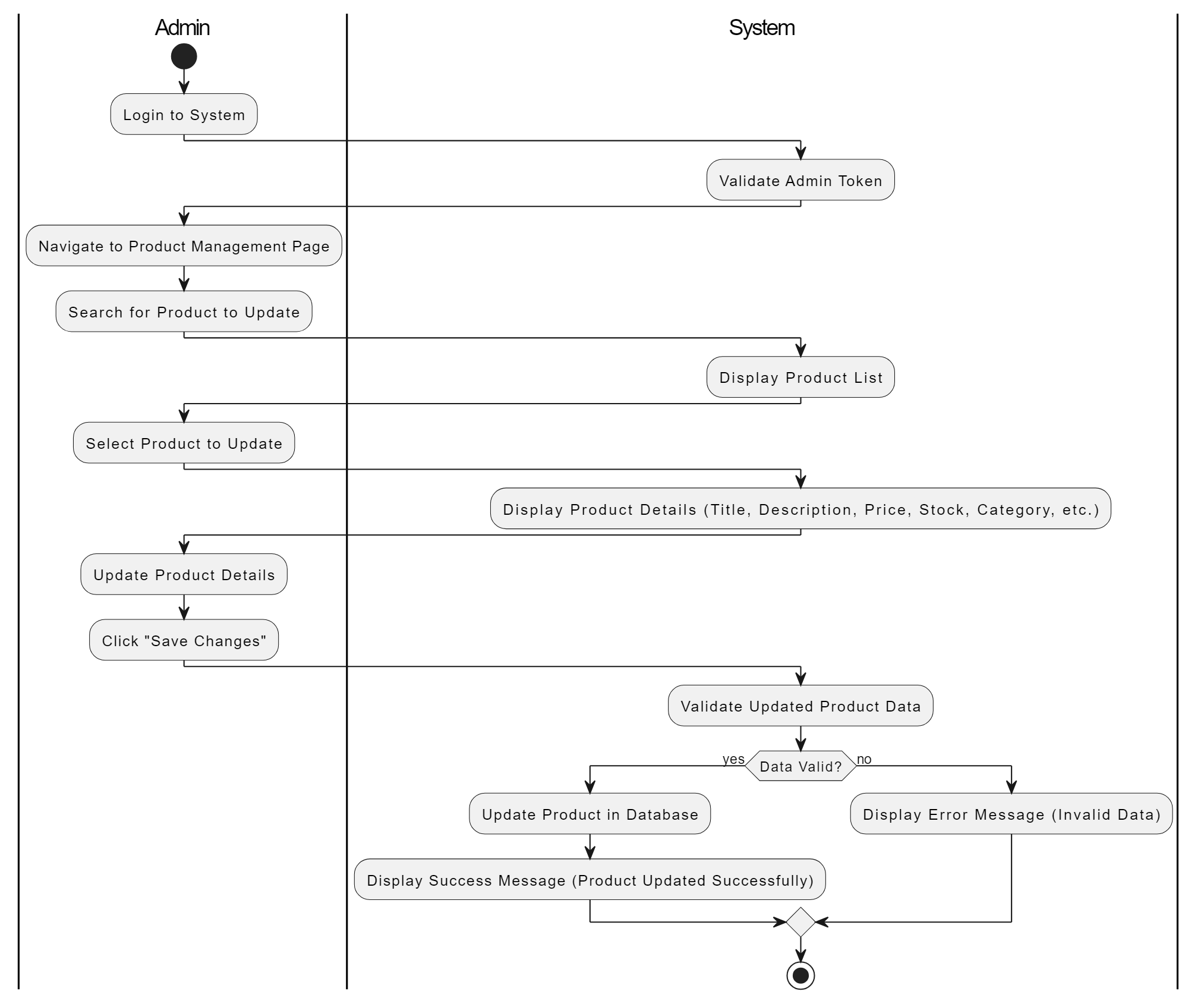
**Product Review**

****

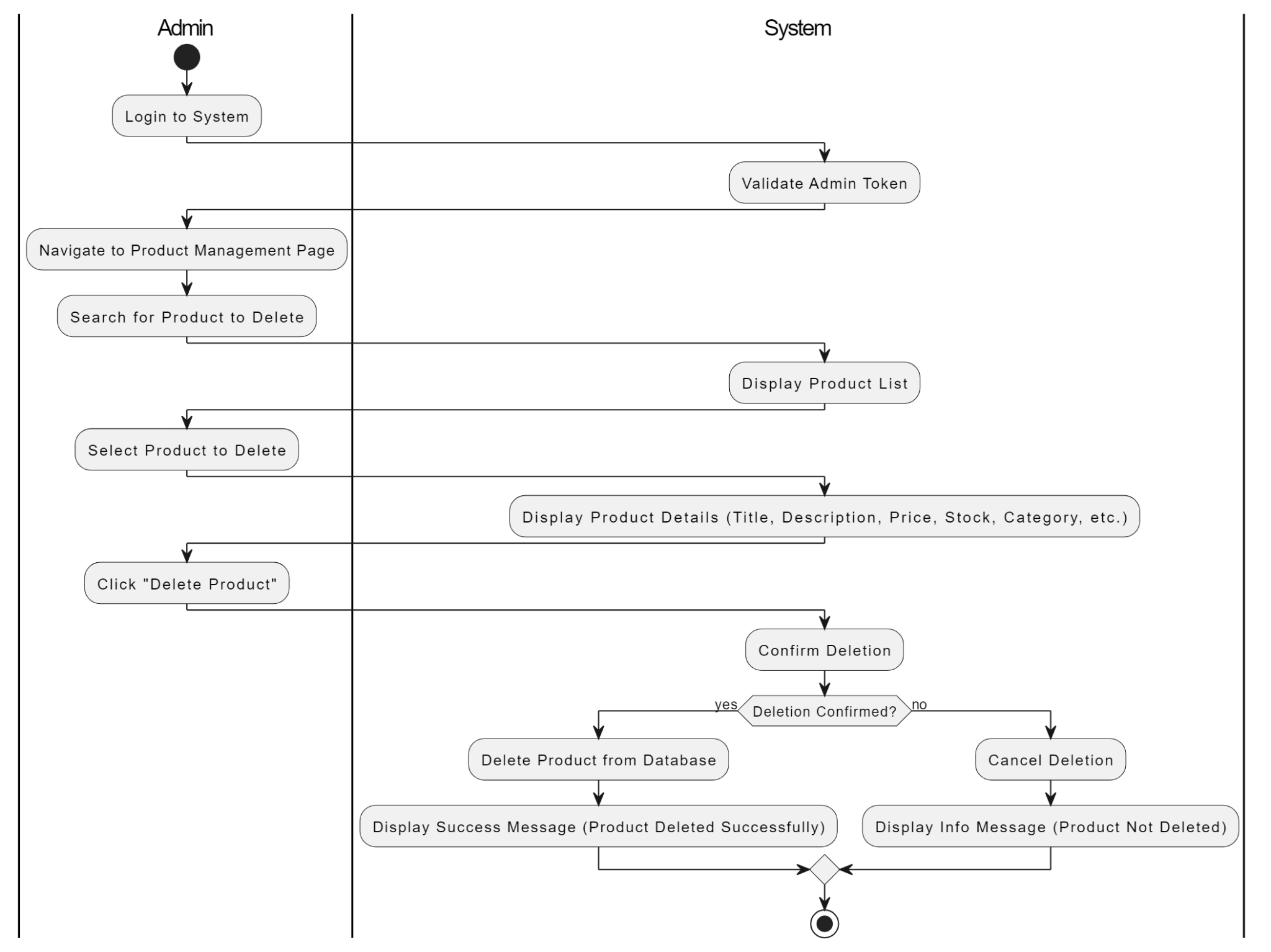
**Admin Product Management (Add Products)**

****

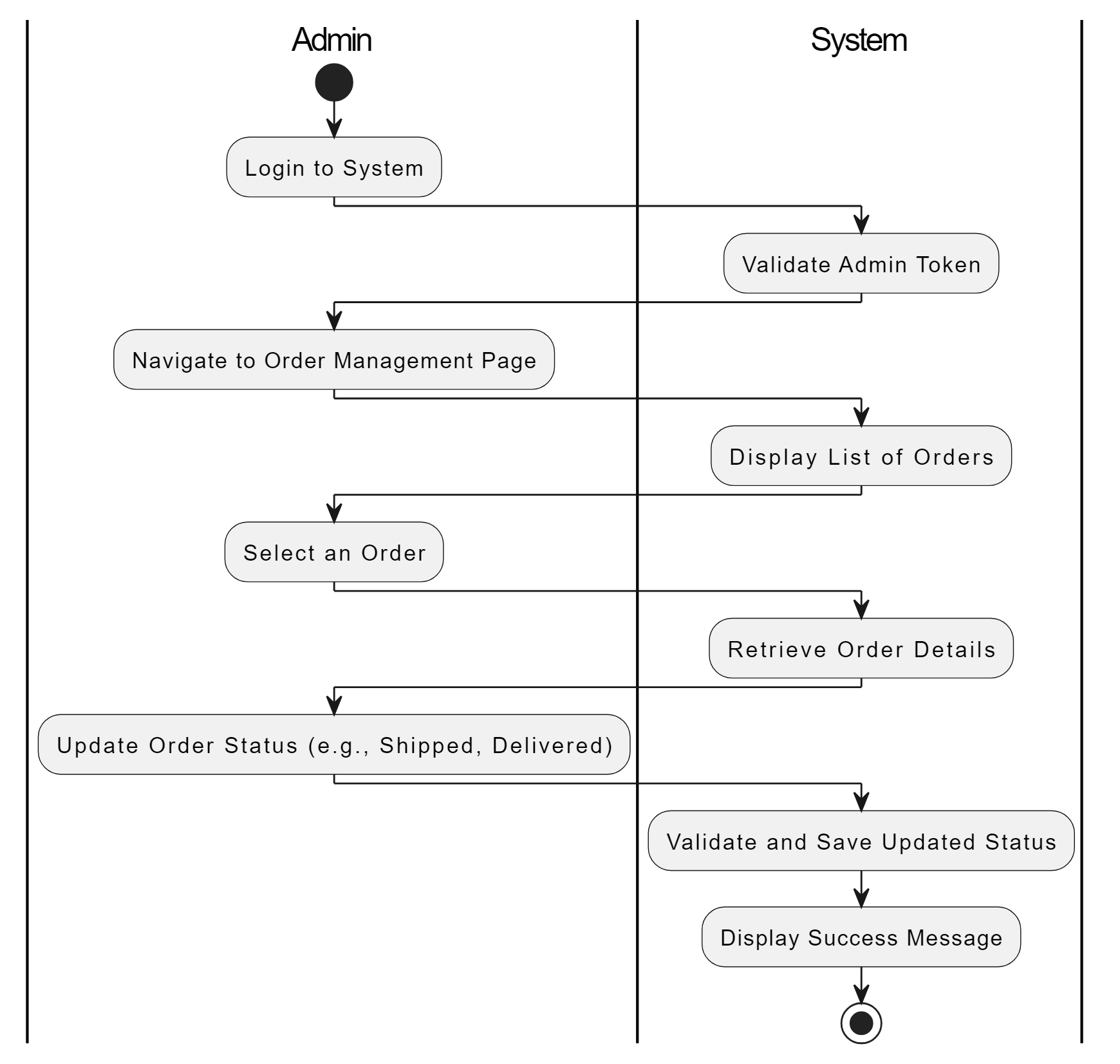
**Admin Product Management (Updaate Products)**

****

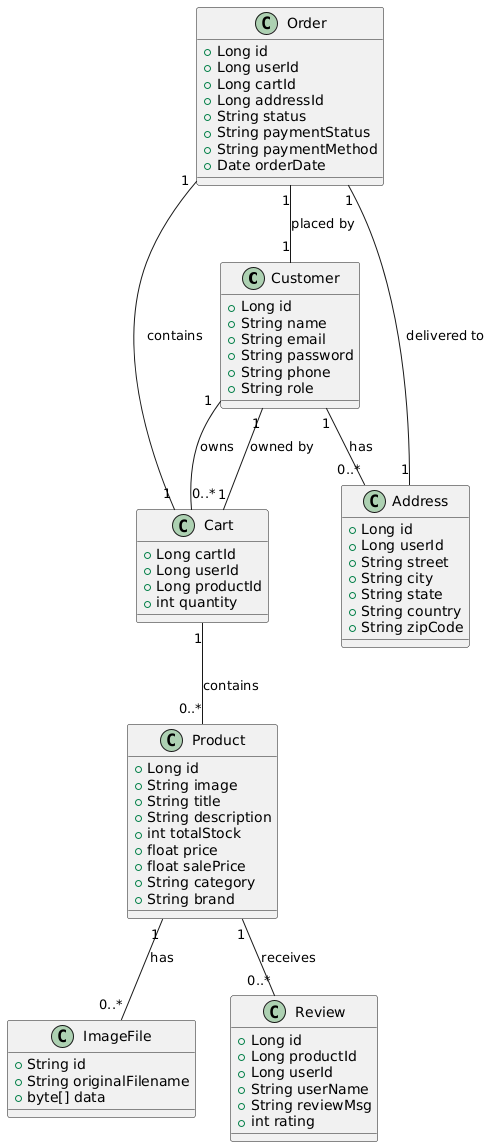
**Admin Product Management (Delete Products)**

****

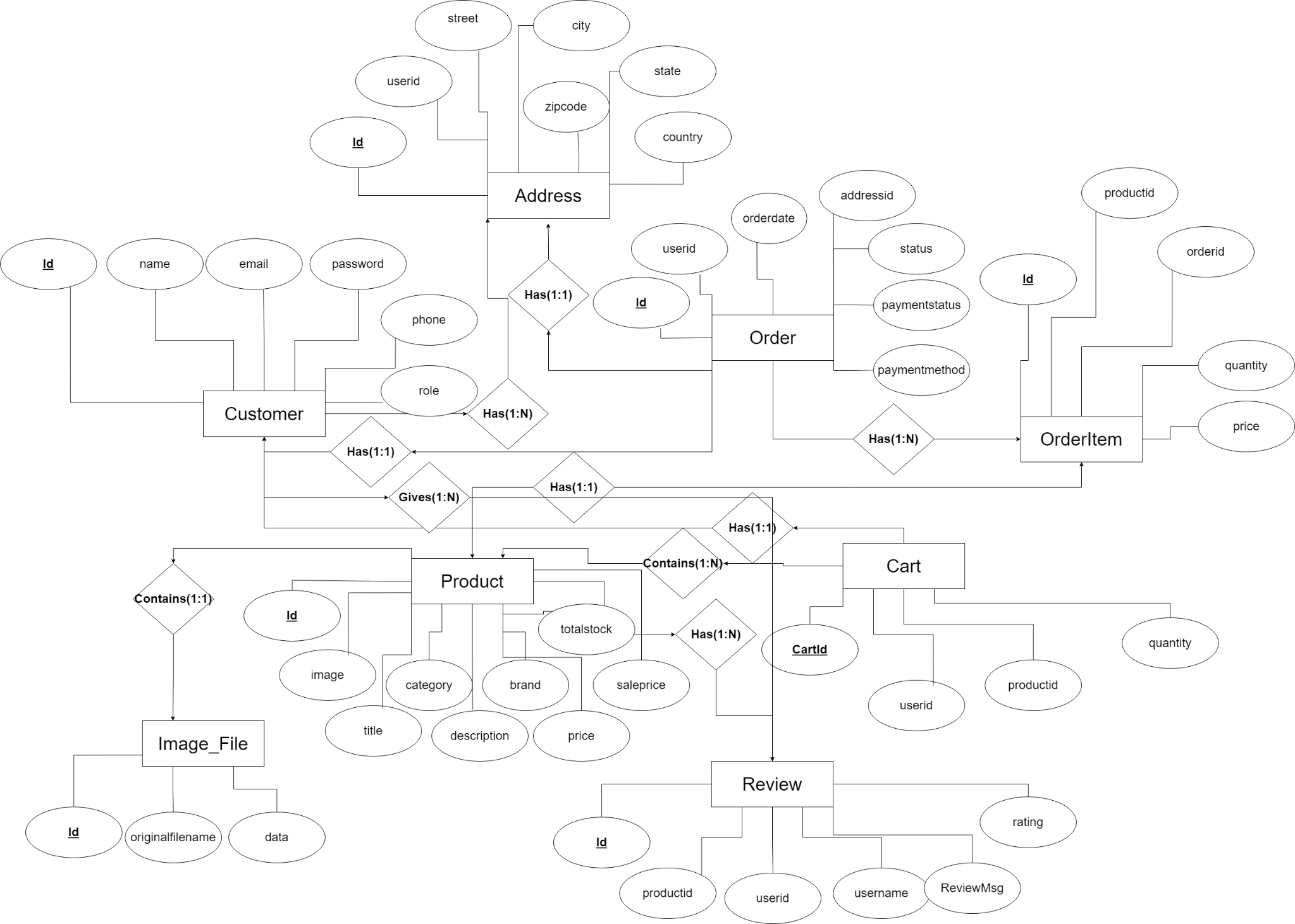
**Admin Order Management**

****

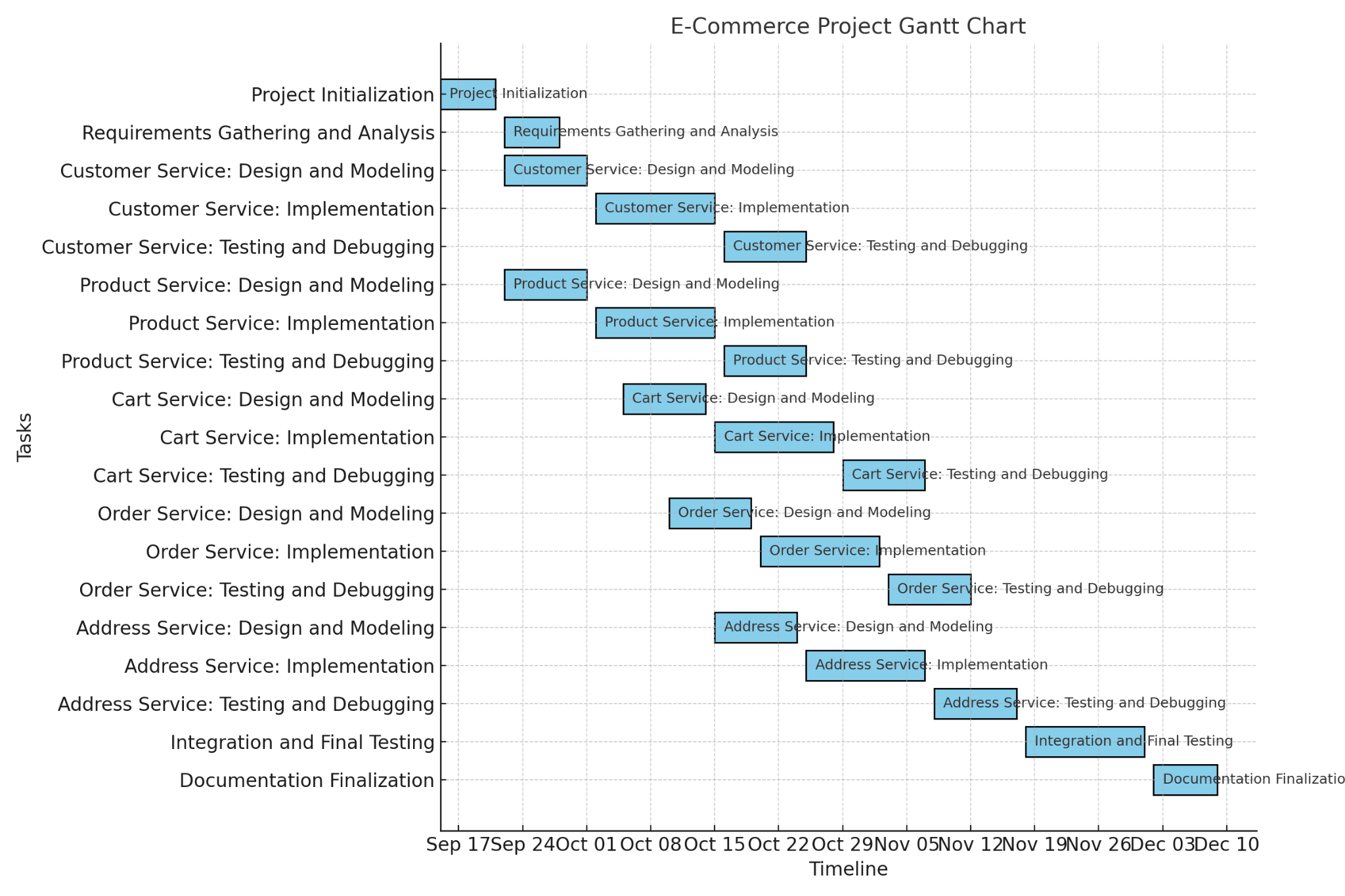
### **4.2.3 Class Diagram**



## **4.3 E-R Diagram**

****

## **4.4 Gantt Chart**



# **5. Software design**

## 

## **5.1 Database Design**

**Table: Product  
Data:** Details about the products in the e-commerce system  
**Where & How Used:** To store information about the products sold on the platform  
**Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| id | Product ID | Long | Primary Key | - |
| image | Image URL or file path | String | Not Null | - |
| title | Product Title | String | Not Null | - |
| description | Product Description | String | Not Null | - |
| totalStock | Total stock available | Integer | Not Null | - |
| price | Product Price | Float | Not Null | - |
| salePrice | Sale Price | Float | Not Null | - |
| category | Product Category | String | Not Null | - |
| brand | Product Brand | String | Not Null | - |

**Table: Cart  
Data: Cart information for users  
Where & How Used: To store user cart details, associating products with the user  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Cart ID** | **Long** | **Primary Key** | **-** |
| **userId** | **User ID (linked to Customer)** | **Long** | **Foreign Key** | **Customer(id)** |
| **productId** | **Product ID (linked to Product)** | **Long** | **Foreign Key** | **Product(id)** |
| **quantity** | **Quantity of Product** | **Integer** | **Not Null** | **-** |

**Table: Address  
Data: User address information  
Where & How Used: To store customer shipping addresses  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Address ID** | **Long** | **Primary Key** | **-** |
| **userId** | **User ID (linked to Customer)** | **Long** | **Foreign Key** | **Customer(id)** |
| **addressLine1** | **Address Line 1** | **String** | **Not Null** | **-** |
| **addressLine2** | **Address Line 2 (optional)** | **String** | **-** | **-** |
| **city** | **City** | **String** | **Not Null** | **-** |
| **state** | **State** | **String** | **Not Null** | **-** |
| **zipCode** | **Zip Code** | **String** | **Not Null** | **-** |
| **country** | **Country** | **String** | **Not Null** | **-** |

**Table: Order  
Data: Information about customer orders  
Where & How Used: To store customer orders with product details  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Order ID** | **Long** | **Primary Key** | **-** |
| **customerId** | **Customer ID (linked to Customer)** | **Long** | **Foreign Key** | **Customer(id)** |
| **totalAmount** | **Total Order Amount** | **Float** | **Not Null** | **-** |
| **status** | **Order Status (Pending/Completed)** | **String** | **Not Null** | **-** |
| **shippingAddressId** | **Shipping Address (linked to Address)** | **Long** | **Foreign Key** | **Address(id)** |
| **paymentStatus** | **Payment Status (Paid/Unpaid)** | **String** | **Not Null** | **-** |
| **orderDate** | **Order Date** | **Date** | **Not Null** | **-** |

**Table: ProductReview  
Data: Reviews given by users for products  
Where & How Used: To store reviews related to products  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Review ID** | **Long** | **Primary Key** | **-** |
| **productId** | **Product ID (linked to Product)** | **Long** | **Foreign Key** | **Product(id)** |
| **userId** | **User ID (linked to Customer)** | **Long** | **Foreign Key** | **Customer(id)** |
| **rating** | **Product Rating (1 to 5)** | **Integer** | **Not Null** | **-** |
| **reviewMessage** | **Review Message** | **String** | **Not Null** | **-** |
| **reviewDate** | **Review Date** | **Date** | **Not Null** | **-** |

**Table: Product  
Data: Details about the products available in the store  
Where & How Used: To store and manage product information  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Product ID** | **Long** | **Primary Key** | **-** |
| **title** | **Product Title** | **String** | **Not Null** | **-** |
| **description** | **Product Description** | **String** | **Not Null** | **-** |
| **price** | **Product Price** | **Float** | **Not Null** | **-** |
| **salePrice** | **Product Sale Price** | **Float** | **-** | **-** |
| **totalStock** | **Total Stock Available** | **Integer** | **Not Null** | **-** |
| **category** | **Product Category** | **String** | **Not Null** | **-** |
| **brand** | **Product Brand** | **String** | **Not Null** | **-** |

**Table: ImageFile  
Data: Product images associated with the products  
Where & How Used: To store product image data  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Image ID** | **Long** | **Primary Key** | **-** |
| **productId** | **Product ID (linked to Product)** | **Long** | **Foreign Key** | **Product(id)** |
| **originalFilename** | **Original Filename** | **String** | **Not Null** | **-** |
| **imageData** | **Image Data (in byte format)** | **Byte[]** | **Not Null** | **-** |

**Table: OrderItem  
Data: Items in an order (linking products to orders)  
Where & How Used: To store each item in an order and associate it with the product and order  
Columns:**

| **Element Name** | **Element Description** | **Data Types** | **Constraint** | **Reference** |
| --- | --- | --- | --- | --- |
| **id** | **Order Item ID** | **Long** | **Primary Key** | **-** |
| **orderId** | **Order ID (linked to Order)** | **Long** | **Foreign Key** | **Order(id)** |
| **productId** | **Product ID (linked to Product)** | **Long** | **Foreign Key** | **Product(id)** |
| **quantity** | **Quantity Ordered** | **Integer** | **Not Null** | **-** |
| **price** | **Price of the Product at the time of the order** | **Float** | **Not Null** | **-** |