Software Engineering-2 Cover Sheet

Project Title: ……………………………………………………………………………………………………………

Row Number (in PDF): …… Time: ………….

|  |  |  |  |
| --- | --- | --- | --- |
| # | Member Id (printed) | Member Name (printed  In Arabic) | Handwritten signature |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

# Project Requirements(grades)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | SRS |  | | SDD |  | | Validation |  | | OCL |  | | AOP |  | | Microservices |  | | Cloud |  | | 2  0 |

Table of Contents

[Project Requirements(grades) 1](#_Toc166018161)

[1.0 Introduction 4](#_Toc166018162)

[1.1 Purpose 4](#_Toc166018163)

[1.2 Scope of the project 4](#_Toc166018164)

[3.0 Software Design Document (SDD) 4](#_Toc166018165)

[3.0Functional Requirements 7](#_Toc166018166)

[3.1 User Registration 7](#_Toc166018167)

[3.2 User Login 8](#_Toc166018168)

[3.3 Product searching and categorization and filtering 8](#_Toc166018169)

[3.4 Payment processing 9](#_Toc166018170)

[3.5 Inventory Management 10](#_Toc166018171)

[3.6 Delivery Address 10](#_Toc166018172)

[4.0 Non-Functional Requirements 11](#_Toc166018173)

[5.0 Use Case Diagrams 14](#_Toc166018174)

[5.1 General Use Case Diagram 14](#_Toc166018175)

[**5.2 Detailed Use Case Diagrams** 15](#_Toc166018176)

[5.2.1 User Registration 15](#_Toc166018177)

[5.2.2 User Login 16](#_Toc166018178)

[5.2.3 Product searching and categorization and filtering 17](#_Toc166018179)

[5.2.4 Payment processing 20](#_Toc166018180)

[5.2.5 Inventory Management 23](#_Toc166018181)

[5.2.6 Delivery Address 26](#_Toc166018182)

[5.2.7 Wish list 28](#_Toc166018183)

[6.0 Activity Diagrams 30](#_Toc166018184)

[6.1 User Login & Registration 30](#_Toc166018185)

[6.2 General Activity Diagrams Diagram 31](#_Toc166018186)

[6.3 Payment processing 32](#_Toc166018187)

[7.0 Sequence Diagrams 33](#_Toc166018188)

[7.1 Login 33](#_Toc166018189)

[7.2 Registration 34](#_Toc166018190)

[7.3 General Sequence Diagram 35](#_Toc166018191)

[7.4 Payment processing 36](#_Toc166018192)

[8.0 Database Specification 37](#_Toc166018193)

[8.1 Logical ERD 37](#_Toc166018194)

[9.0 Class Diagram 38](#_Toc166018195)

[10.0 OCL 39](#_Toc166018196)

# **Introduction**

## 1.1 Purpose

The purpose of this document is to provide a detailed description of the Online Store application. The document will cover all aspects of the application, including its features, functionalities, and workflow. It is written in a user-friendly manner, with clear and concise explanations to facilitate understanding.

## 1.2 Scope of the project

The ecommerce online store app is designed to provide a platform for users to browse, purchase, a variety of products. The app will feature a user-friendly interface, allowing customers to easily search for products, add them to their cart, and make secure payments. Sellers will have the ability to create accounts, list products, manage inventory, Key features of the app.

# **3.0 Software Design Document (SDD)**

*1. Introduction:*

The online store application is designed to provide a platform for users to browse, search, and purchase products online. The application will include features such as user authentication, product catalog, shopping cart, payment processing, and order management.

*2. System Overview:*

The system will be developed using the Spring Boot framework, with a

MySQL database for data storage. The application will follow a micro services architecture, with separate services for user management, product catalog, shopping cart, payment processing, and order management.

*3. System Architecture:*

The system will consist of the following components:

User Management Service: Responsible for user authentication and management.

Product Catalog Service: Manages the product catalog, including product information and availability.

Shopping Cart Service: Manages the user's shopping cart and allows users to add, remove, and update items in the cart.

Payment Processing Service: Handles payment processing using third-party payment gateways.

Order Management Service: Manages user orders, including order placement, fulfillment, and tracking.

***4. Data Design:***

*4.1 Database Schema*

The database will consist of the following tables:

Users: Stores user information such as username, password, and email address.

Products: Stores product information such as name, description, price, and availability.

Shopping Cart: Stores the user's shopping cart items.

Orders: Stores order information such as order ID, user ID, and order status.

*4.2 Data Access Layer*

Data access will be implemented using Spring Data JPA, which provides easy integration with the MySQL database.

*5. User Interface Design:*

The user interface will be implemented using Spring Boot, React, MySQL Material-UI (MUI), Tailwind CSS, Redux, and React Router DOM with JSX (JavaScript XML) as the Template engine. The application will follow a responsive design to ensure compatibility with various devices and screen sizes.

*6. Security:*

The application will implement security best practices, including password hashing.

*7. Performance Considerations:*

To ensure optimal performance, the application will use caching for frequently accessed data and optimize database queries.

*8. Deployment:*

The application will be deployed to a cloud platform such Docker will be used to automate the deployment process.

*9. Conclusion:*

The online store application is designed to provide a secure, scalable, and user-friendly platform for users to shop online. The application leverages modern technologies and best practices to deliver a high-quality shopping experience.

# **3.0Functional Requirements**

## 3.1 User Registration

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **The system shall provide a registration feature that allows new users to create an account.** |
| **System Requirement(s)** | **Description** | * **The system shall allow users to enter their first name, last name, email address, password, and confirm password to register for a new account.** * **The system shall validate the user's information and prevent registration if any required field is missing or invalid.** * **The system shall check if the email address provided by the user is already registered in the system, and prevent registration if it is.** * **The system shall store the user's information in the database after successful registration.** |
| **Pre-condition(s)** | * **The user is not registered in the system.** * **The user has a valid email address and password.** |
| **Post-condition(s)** | * **The user's information is stored in the system upon successful registration.** |

## 3.2 User Login

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **The system shall provide a login feature that allows registered users to access their accounts.** |
| **System Requirement(s)** | **Description** | * **The system shall allow users to enter their email address and password to log in to their account.** * **The system shall validate the user's credentials against the stored user information in the database.** * **If the user's credentials are valid, the system shall grant access to the user's account.** * **If the user's credentials are invalid, the system shall prevent access to the user's account.** |
| **Pre-condition(s)** | * **The user is already registered in the system.** * **The user has a valid email address and password.** |
| **Post-condition(s)** | * **The user is granted access to their account upon successful login.** |

## 3.3 Product searching and categorization and filtering

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **Users should be able to search for products easily based on keywords, categories, or filters such as price range, brand, size, color, etc. They should also be able to navigate through different product categories to find items of interest.** |
| **System Requirement(s)** | **Description** | * **The system shall provide a search functionality that allows users to search for products based on keywords, categories, and various filters.** |
| **Pre-condition(s)** | * **The user is on the e-commerce app and intends to search for products** |
| **Post-condition(s)** | * **The search results are displayed to the user, showing relevant products based on the search query and selected filters** |

## 3.4 Payment processing

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **Users should be able to securely purchase products using various payment methods such as credit/debit cards, digital wallets, or other online payment systems. They should also receive confirmation of their order and payment status.** |
| **System Requirement(s)** | **Description** | * **The system shall provide a secure payment processing functionality that allows users to make purchases using credit/debit cards, digital wallets, or other online payment systems** |
| **Pre-condition(s)** | * **The user has selected products for purchase and has proceeded to the checkout process.** |
| **Post-condition(s)** | * **The payment is processed securely, and the user receives a confirmation of the order along with the payment status. The purchased items are reserved for the user.** |

## 

## 3.5 Inventory Management

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **Users should be able to view the availability of products in real-time, add items to their cart, and receive notifications if a product is out of stock. They should also be able to view the estimated delivery time if a product is not in stock** |
| **System Requirement(s)** | **Description** | * **The system shall manage the inventory of products, providing real-time updates on availability, allowing users to add items to their cart, and notifying users if a product is out of stock.** |
| **Pre-condition(s)** | * **The user is browsing the product catalog or has added items to their cart.** |
| **Post-condition(s)** | * **The system updates the inventory status after a user completes a purchase, ensuring that the available quantity is accurate. Users are notified if a product is out of stock and given the option to proceed with the purchase with a delayed delivery time or remove the item from their cart** |

## 3.6 Delivery Address

|  |  |  |
| --- | --- | --- |
| **User Requirement(s)** | **Overview** | * **Users should be able to add, edit, and save multiple delivery addresses to their account. They should also be able to select a saved address during the checkout process.** |
| **System Requirement(s)** | **Description** | * **The system shall allow users to manage their delivery addresses, including adding, editing, and saving multiple addresses to their account. Users should be able to select a saved address during the checkout process.** |
| **Pre-condition(s)** | * **The user is logged in to their account and is accessing the delivery address management section.** |
| **Post-condition(s)** | * **The user's selected delivery address is saved and can be used during the checkout process. Any changes made to the delivery address are reflected in the user's account.** |

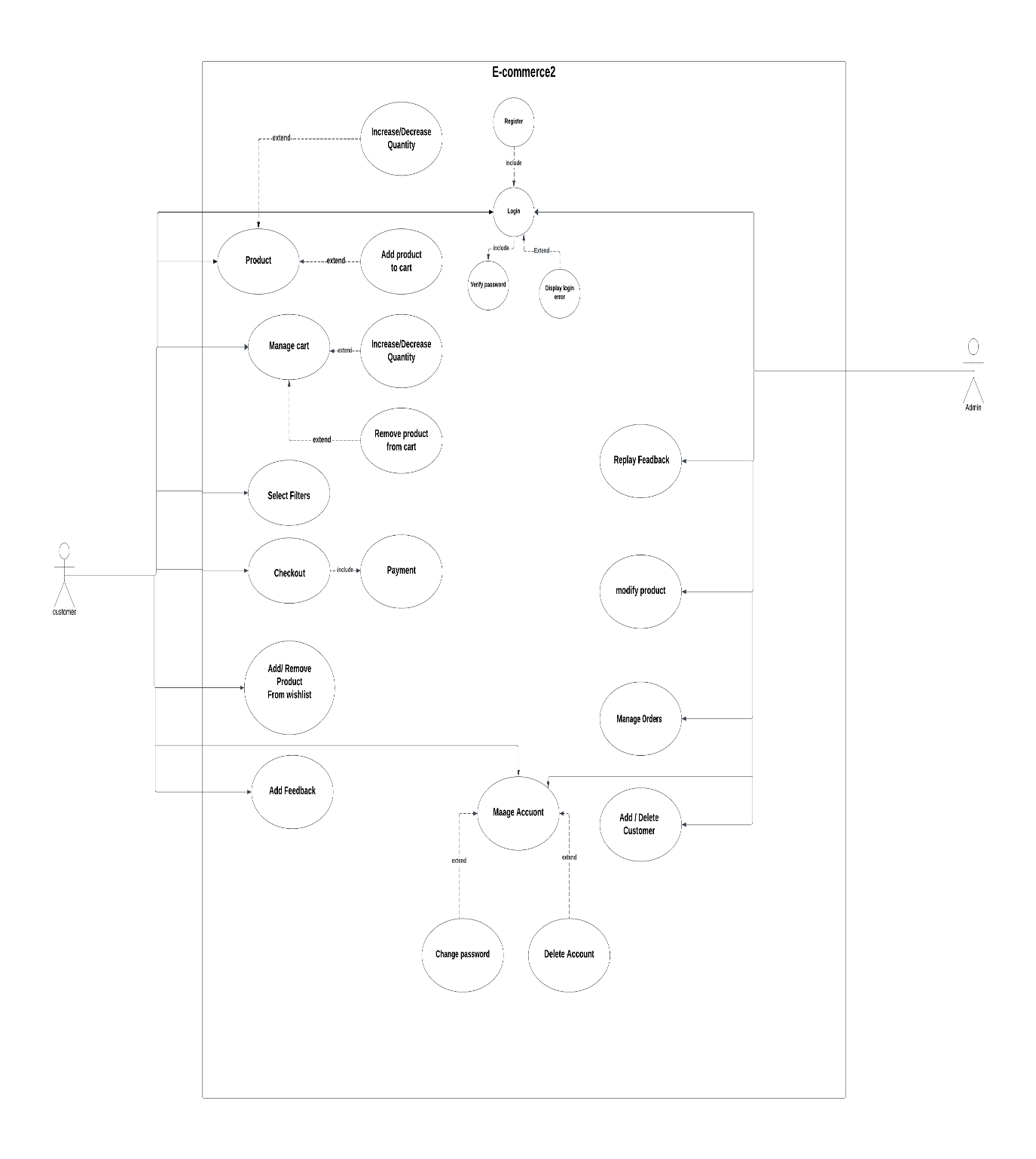
# **4.0 Non-Functional Requirements**

* ***Performance****:*
* The system must respond to user requests in less than 3 seconds.
* The system must be reliable, flexible, and able to handle multiple users simultaneously.
* ***Maintainability****:*
* The system must be easy to change/update.
* ***Serviceability****:*
* The system must be easy to maintain, monitor, and repair any problems that arise, as well as adding and removing users.
* ***Data Integrity****:*
* The system must ensure accuracy, completeness, and consistency of data for safety and security.
* ***Manageability****:*
* The system must be efficient and easy to monitor and maintain to keep the system performing, secure, and running smoothly.
* ***Interface****:*
* The system must have a good interface that is easy to understand.
* ***Look and Feel****:*
* The system should be painted in comfortable colors for the eyes.
* ***Safety and Security****:*
* The system must ensure that only authorized users can gain access and distinguish between authorized and non-authorized users.
* ***Availability****:*
* The system must be available 24 hours a day.
* ***Reliability***:
* The system must be reliable and not have data validation failures.

**Note**: These requirements are organized into different categories to improve readability and clarity.

# **5.0 Use Case Diagrams**

## 5.1 General Use Case Diagram



## **5.2 Detailed Use Case Diagrams**

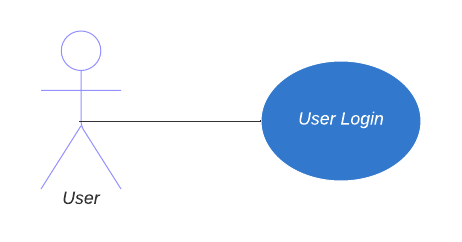
### 5.2.1 User Registration

Diagram

Description automatically generated

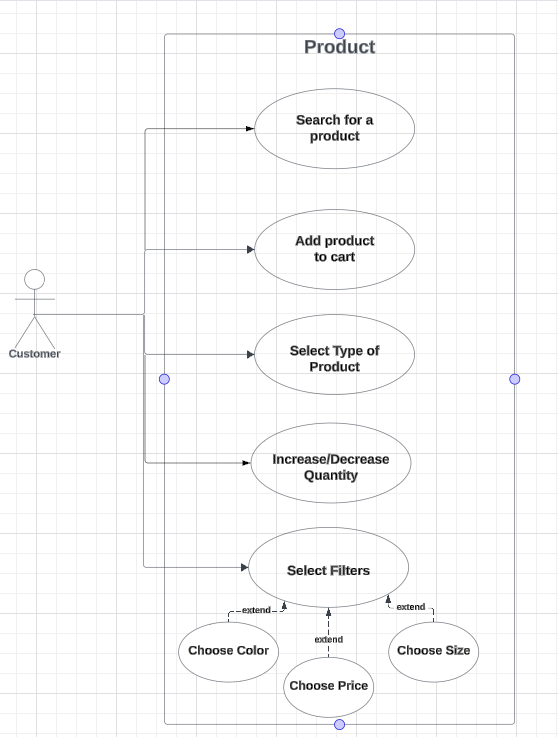
|  |  |
| --- | --- |
| **ID Number – Name** | **UC-1 – User Registration** |
| **Short Description** | **The user enters their first name, last name, email address, password, and confirm password to register for a new account.** |
| **Actor(s)** | **User** |
| **Standard Process**  **(Main Success Scenario)** | 1. **The user opens the site.** 2. **The user enters their first name, last name.** 3. **The user enters their email address and creates a password.** 4. **The user confirms their password.** 5. **The user clicks the "Register" button.** 6. **The site redirects the user to the login page.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | 1. **If the user enters an email address that is already registered, the site displays an error message and prompts the user to enter a different email address.** 2. **If the user does not confirm their password correctly, the site displays an error message and prompts the user to confirm their password again.** 3. **If the user enters incorrect or incomplete information, the site displays an error message and prompts the user to correct their information.** |

### 5.2.2 User Login



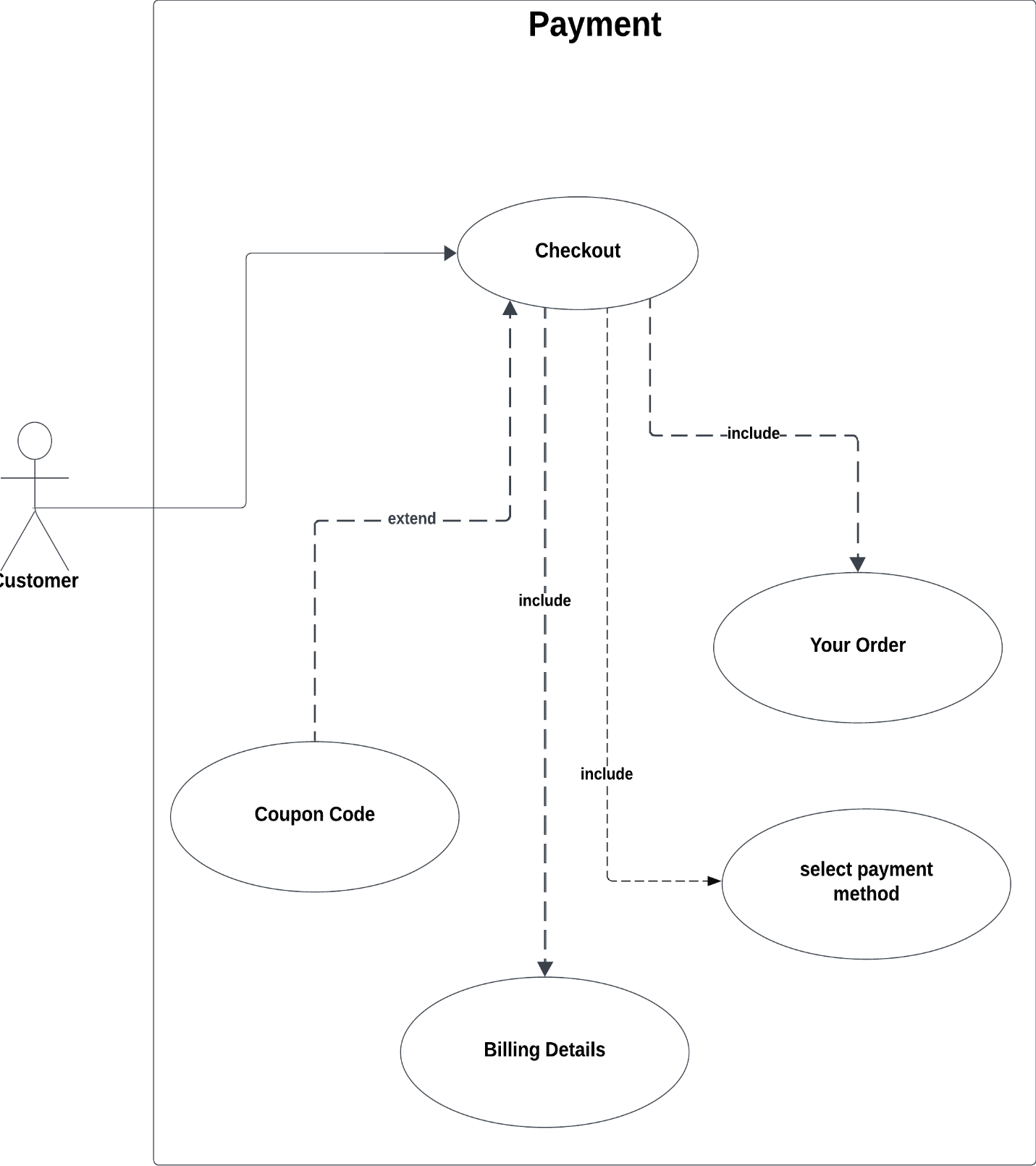
|  |  |
| --- | --- |
| **ID Number – Name** | **UC-2 – User Login** |
| **Short Description** | **The user enters their email address, and password to login.** |
| **Actor(s)** | **User** |
| **Standard Process**  **(Main Success Scenario)** | 1. **The user opens the site.** 2. **The user enters their email address, and password.** 3. **The user clicks the "Login" button.** 4. **The site verifies the information and grants access to the user's account.** 5. **The site redirects the user to the home page.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | 1. **If the user enters incorrect login information, the site displays an error message and prompts the user to re-enter their information.** |

### 5.2.3 Product searching and categorization and filtering



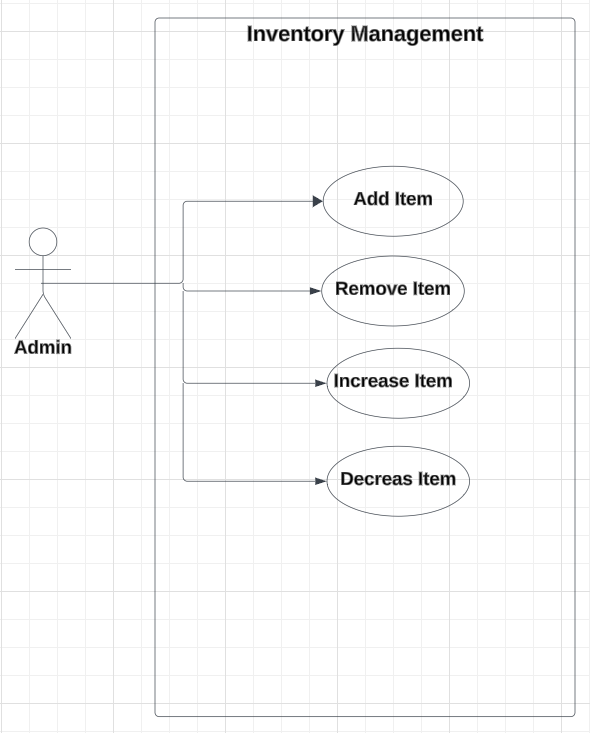
|  |  |
| --- | --- |
| **ID Number – Name** | **Product Selection and Viewing** |
| **Short Description** | **This feature enables users to browse, select, and view product details on the e-commerce website.** |
| **Actor(s)** | **User** |
| **Standard Process**  **(Main Success Scenario)** | * 1. **User navigates to the e-commerce website and enters the product browsing section. 2. User browses through various product categories or uses search functionality to find specific products. 3. User selects a product of interest to view its details. 4. Application displays the product details page, including images, description, price, available variations (if applicable), and other relevant information. 5. User reviews the product details and decides whether to proceed with the purchase. 6. If satisfied, user adds the product to the shopping cart. 7. Application confirms the addition of the product to the cart and provides options to continue shopping or proceed to checkout. 8. User may choose to continue shopping or proceed to checkout to complete the purchase.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | * + 1. **User encounters an out-of-stock product. 2. Application notifies the user about the out-of-stock status and provides options to receive notifications when the product is available again or to explore alternative products. 3. User may choose to wait for the product to become available, select an alternative product, or exit the website.  4. If the user selects an alternative product, the standard process resumes from step 3. 5. If the user decides to wait for the product, they may opt to receive notifications about its availability. 6. Once the product is restocked, user receives a notification from the application. 7. User revisits the product details page and proceeds with adding the product to the shopping cart or completing the purchase. 8. Application updates the order status accordingly and notifies the user of the successful addition of the product to the cart. 9. User may choose to continue shopping, review the cart, or proceed to checkout.** |

### 5.2.4 Payment processing



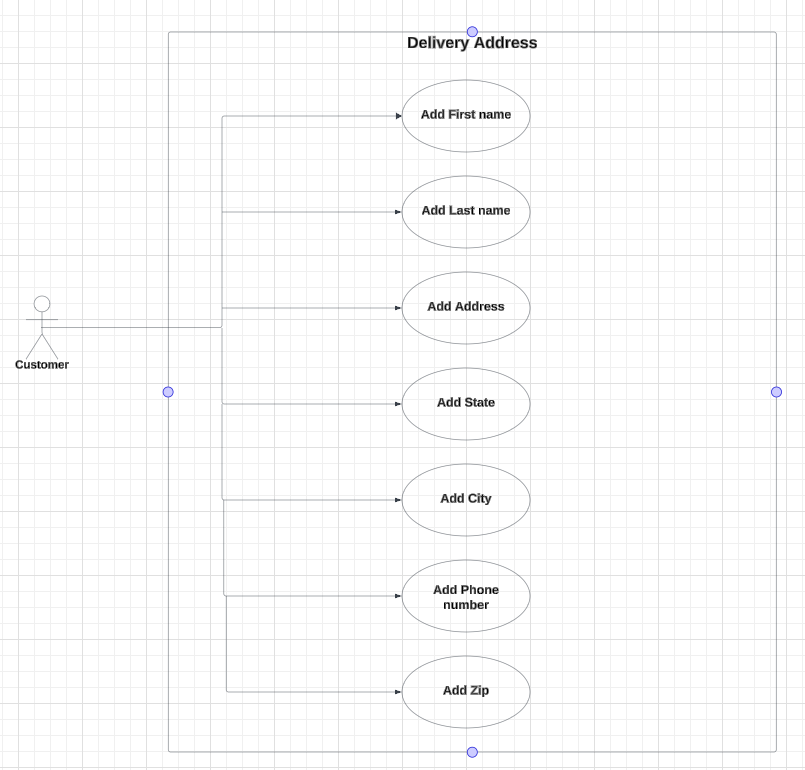
|  |  |
| --- | --- |
| **ID Number – Name** | **Payment process** |
| **Short Description** | **The feature enables users to make purchases on the e-commerce website by completing payment transactions securely.** |
| **Actor(s)** | **User, Payment Gateway** |
| **Standard Process**  **(Main Success Scenario)** | **1.User selects the desired product(s) and proceeds to checkout. 2.** **User enters shipping and billing information. 3. User selects a payment method (e.g., credit/debit card, PayPal, etc.). 4.** **Application redirects the user to the chosen payment gateway. 5. User enters payment details securely on the payment gateway's platform. 6. Payment gateway processes the transaction securely. 7. Application receives confirmation of successful payment from the payment gateway. 8. Application updates the order status to "Paid" and sends a confirmation email to the user. 9. User receives the confirmation email and completes the purchase. 10. User can choose to continue shopping, view order details, or exit the website.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | **1.User encounters an error during the payment process, such as declined payment or technical issues. 2. User is prompted to retry the payment or use an alternative payment method. 3. If the issue persists, user contacts customer support for assistance. 4. Customer support assists the user in resolving the payment issue, which may include troubleshooting, providing alternative payment options, or addressing technical glitches. 5. Once the payment issue is resolved, user completes the transaction as per the standard process. 6. Application updates the order status accordingly and notifies the user of the successful payment. 7. User can proceed with the purchase or choose to take other actions such as modifying the order or canceling it.** |

### 5.2.5 Inventory Management



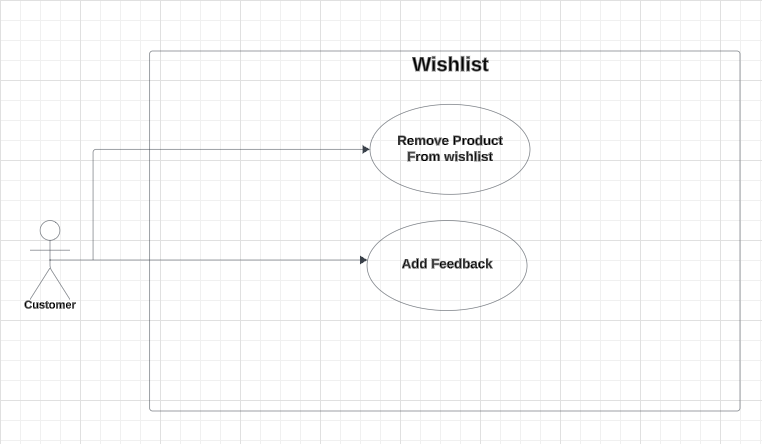
|  |  |
| --- | --- |
| **ID Number – Name** | **Inventory Management** |
| **Short Description** | **This feature enables the admin to manage the inventory of products on the e-commerce website by adding, removing, increasing, or decreasing the quantity of items available.** |
| **Actor(s)** | **Admin** |
| **Standard Process**  **(Main Success Scenario)** | **1. Admin accesses the inventory management section of the e-commerce website by logging into their admin account. 2. Admin selects one of the following actions: a. "Add Item" - to add a new product to the inventory. b. "Remove Item" - to remove an existing product from the inventory. c. "Increase Item" - to increase the quantity of an existing product in the inventory. d. "Decrease Item" - to decrease the quantity of an existing product in the inventory. 3. Admin provides the necessary details for the selected action: a. For "Add Item": product name, description, category, price, quantity, etc. b. For "Remove Item", "Increase Item", and "Decrease Item": product identifier (e.g., SKU or product ID) and quantity (if applicable). 4. Application processes the admin's request based on the selected action: a. For "Add Item": Application adds the new product to the inventory with the provided details. b. For "Remove Item": Application removes the specified product from the inventory. c. For "Increase Item": Application increases the quantity of the specified product in the inventory by the provided amount. d. For "Decrease Item": Application decreases the quantity of the specified product in the inventory by the provided amount, ensuring that the inventory quantity does not fall below zero. 5. Application updates the inventory database accordingly.  6. Admin receives a confirmation message indicating the successful completion of the inventory management action.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | **1. Admin attempts to perform an inventory management action on a non-existent or invalid product. 2. Application notifies the admin about the invalid product and prompts them to provide valid product details. 3. Admin revises the provided product details or selects a valid product and resubmits the request. 4. Application processes the revised request and updates the inventory accordingly. 5. Admin encounters an error while performing an inventory management action due to technical issues or database errors. 6. Application displays an error message informing the admin about the issue and suggests retrying later or contacting technical support for assistance. 7. Admin may choose to retry the inventory management action, use alternative methods, or seek assistance from technical support. 8. If the issue persists, admin contacts technical support for further assistance. 9. Technical support investigates the issue and resolves it, ensuring that the admin can perform inventory management actions without any further problems. 10. Once the issue is resolved, admin can continue managing the inventory as per the standard process.** |

### 5.2.6 Delivery Address



|  |  |
| --- | --- |
| **ID Number – Name** | **Delivery Address Management** |
| **Short Description** | **This feature enables users to manage delivery addresses for orders placed on the e-commerce website, ensuring accurate and convenient shipment of purchased items.** |
| **Actor(s)** | **User** |
| **Standard Process**  **(Main Success Scenario)** | **1. User navigates to the e-commerce website and logs into their account. 2. User accesses the account settings or a dedicated address management section. 3. User selects the option to add a new delivery address. 4. Application prompts the user to enter details such as recipient name, street address, city, state/province, postal/ZIP code, and country. 5. User fills in the required address information accurately. 6. Application validates the entered address to ensure correctness and completeness. 7. If the address is valid, user confirms the addition of the new delivery address. 8. Application saves the new delivery address to the user's address book. 9. User can view, edit, or delete saved delivery addresses as needed. 10. When placing an order, user selects a saved delivery address from their address book during the checkout process.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | **1. User attempts to add an incomplete or incorrect delivery address. 2. Application notifies the user about the missing or incorrect information and prompts them to provide the necessary details. 3. User revises the address information and submits the corrected details. 4. Application validates the revised address and saves it to the user's address book if it meets the validation criteria. 5. If the address remains invalid after revision, user may choose to correct it again or cancel the addition. 6. User encounters an error while saving the delivery address due to technical issues. 7. Application displays an error message informing the user about the issue and suggests retrying later or contacting customer support for assistance. 8. User may choose to retry saving the delivery address, use an alternative address, or exit the address management section. 9. If the issue persists, user contacts customer support for further assistance. 10. Customer support investigates the issue and resolves it, ensuring that the user can save delivery addresses without any further problems. 11. Once the issue is resolved, user can add, edit, or delete delivery addresses as per the standard process.** |

### 5.2.7 Wish list



|  |  |
| --- | --- |
| **ID Number – Name** | **wish list Management** |
| **Short Description** | **This feature allows users to create and manage a list of desired products, facilitating easy access and future purchase consideration.** |
| **Actor(s)** | **User** |
| **Standard Process**  **(Main Success Scenario)** | **1. User navigates to the e-commerce website and logs into their account. 2. User browses through various product categories or searches for specific products. 3. User finds a product they are interested in and selects the "Add to Wish list" option. 4. Application adds the selected product to the user's Wish list. 5. User can view and manage their Wish list from their account dashboard or a dedicated wish list section. 6. User may add multiple products to the wish list. User can review product details, such as images, descriptions, and prices, within the wish list. 7. If desired, user can remove a product from the wish list or move it to the shopping cart for purchase. 8. User can also share their wish list with others via email or social media. 9. User may revisit their wish list at any time to update or modify it.** |
| **Alternative Process(es) (Alternative/Other Scenario(s))** | **1. User attempts to add a product to the wish list without being logged in. 2. Application prompts the user to log in or create an account to use the wish list feature. 3. If the user chooses to log in or create an account, the standard process resumes from step 1. 4. If the user decides not to log in or create an account, they cannot add the product to the wish list. 5. User encounters a technical issue while trying to add a product to the wish list. 6. Application displays an error message informing the user about the issue and suggests retrying later or contacting customer support for assistance. 7. User may choose to retry adding the product to the wish list, explore other products, or exit the website. 8. If the issue persists, user contacts customer support for further assistance. 9. Customer support investigates the issue and resolves it, ensuring that the user can add products to their wish list without any further problems. 10. Once the issue is resolved, user can add products to their wish list as per the standard process.** |

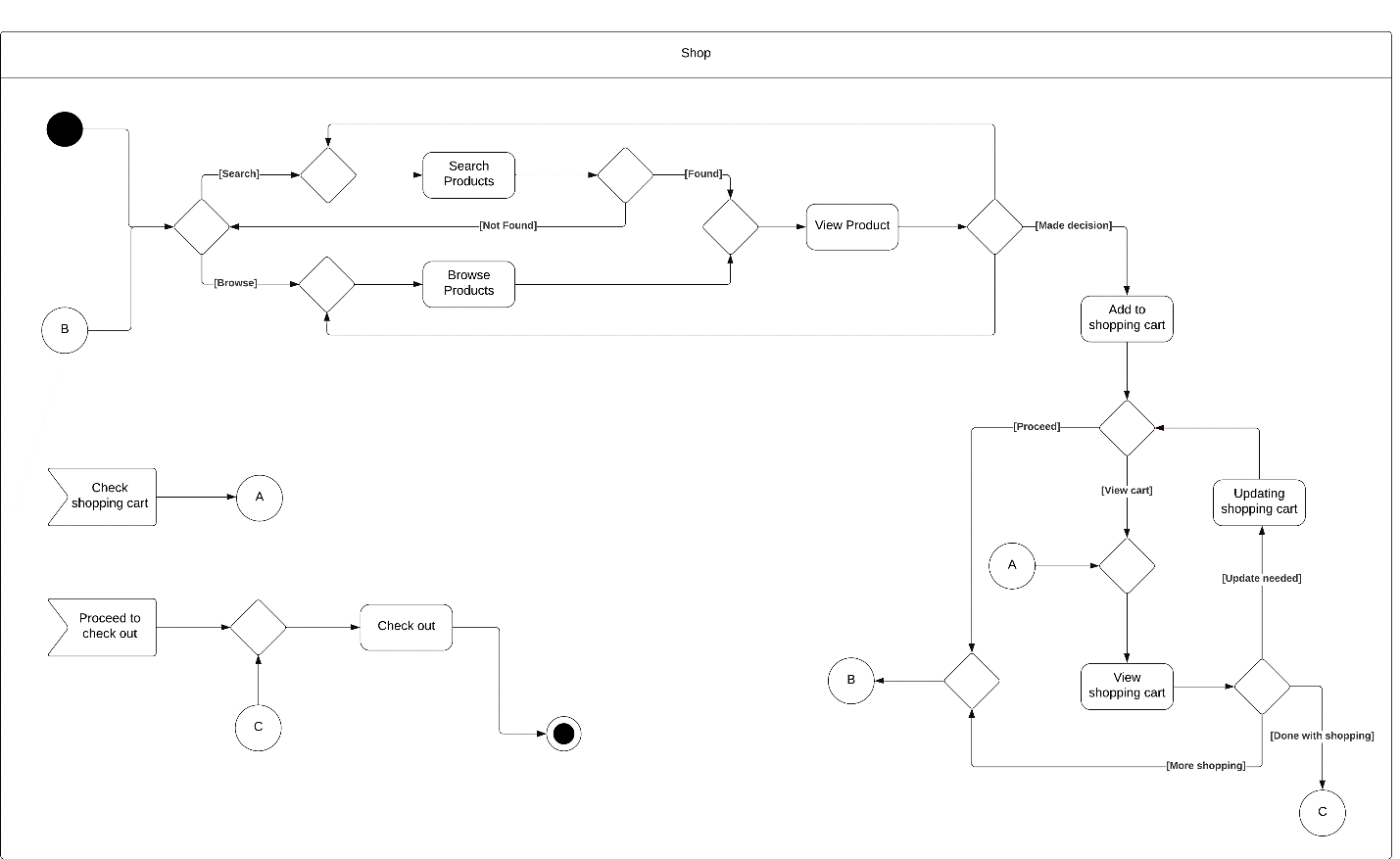
# **6.0 Activity Diagrams**

## 6.1 User Login & Registration

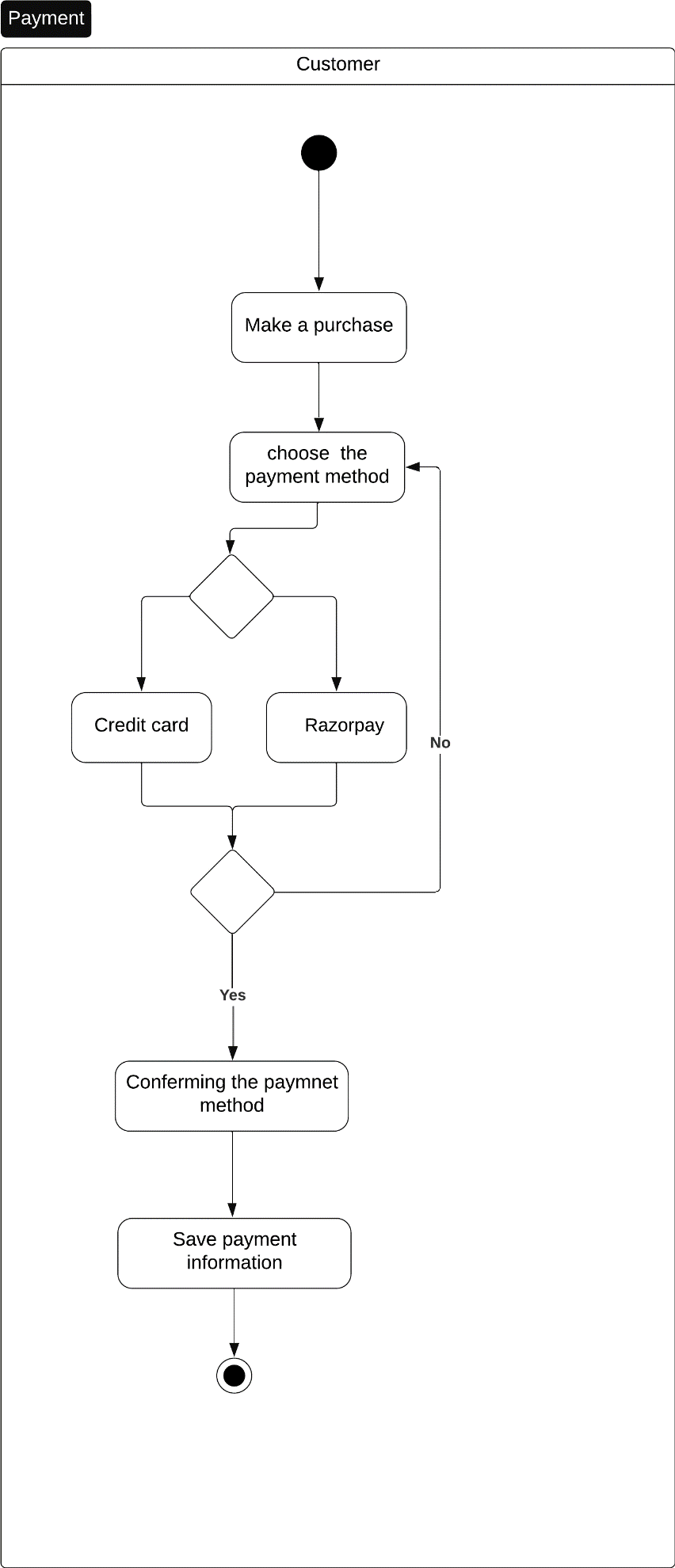
Diagram

Description automatically generated

## 6.2 General Activity Diagrams Diagram

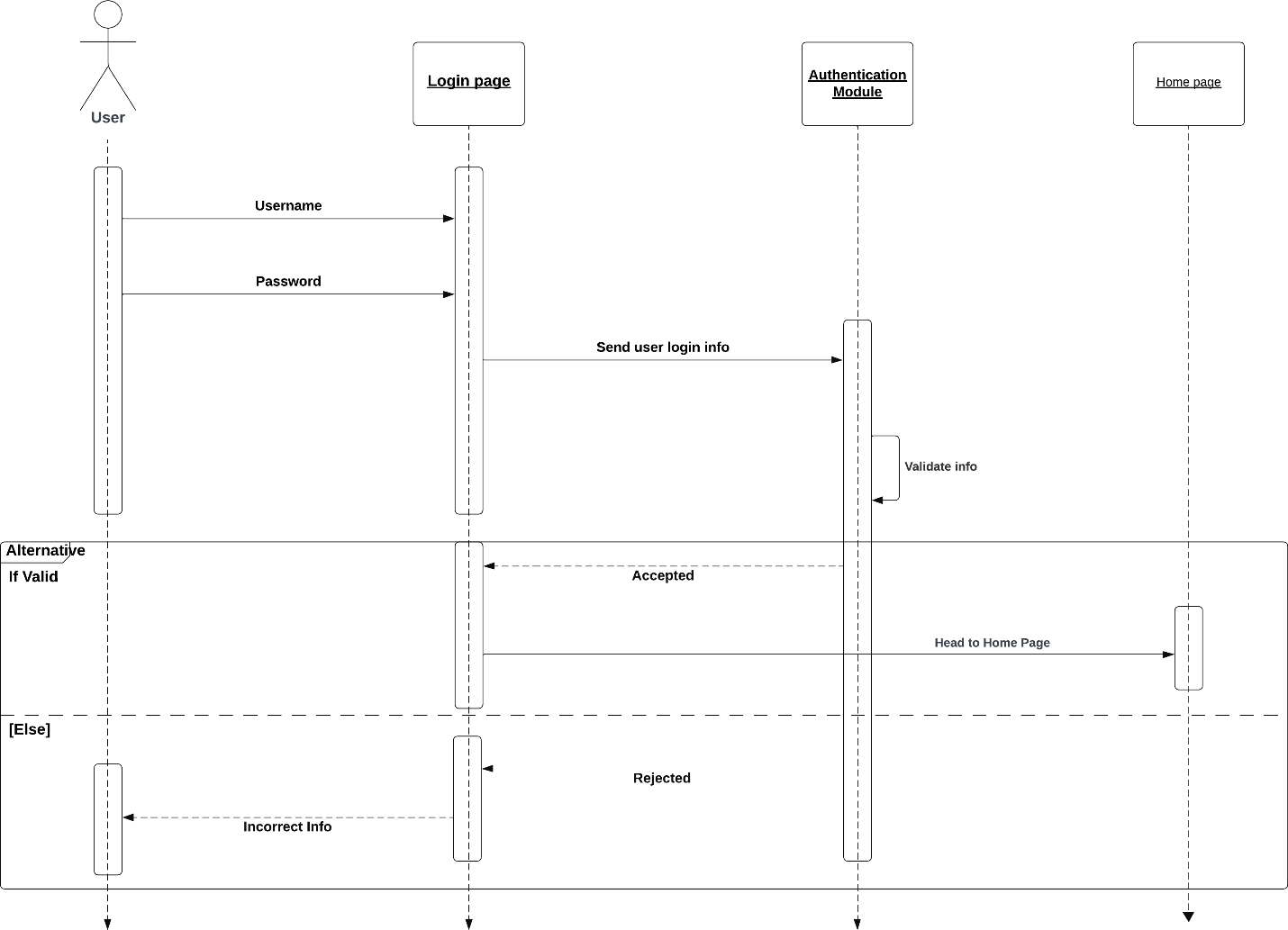


## 6.3 Payment processing

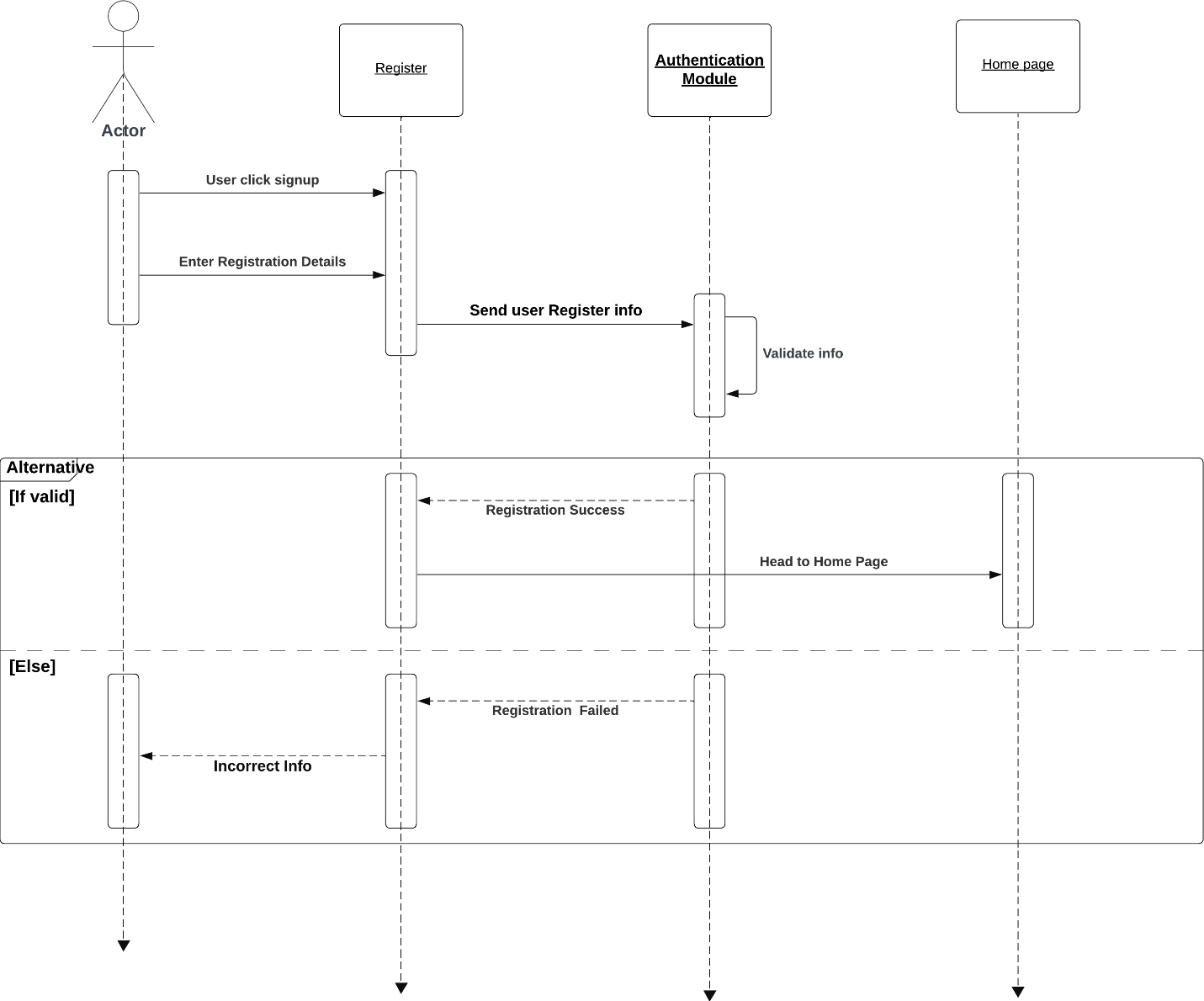


# **7.0 Sequence Diagrams**

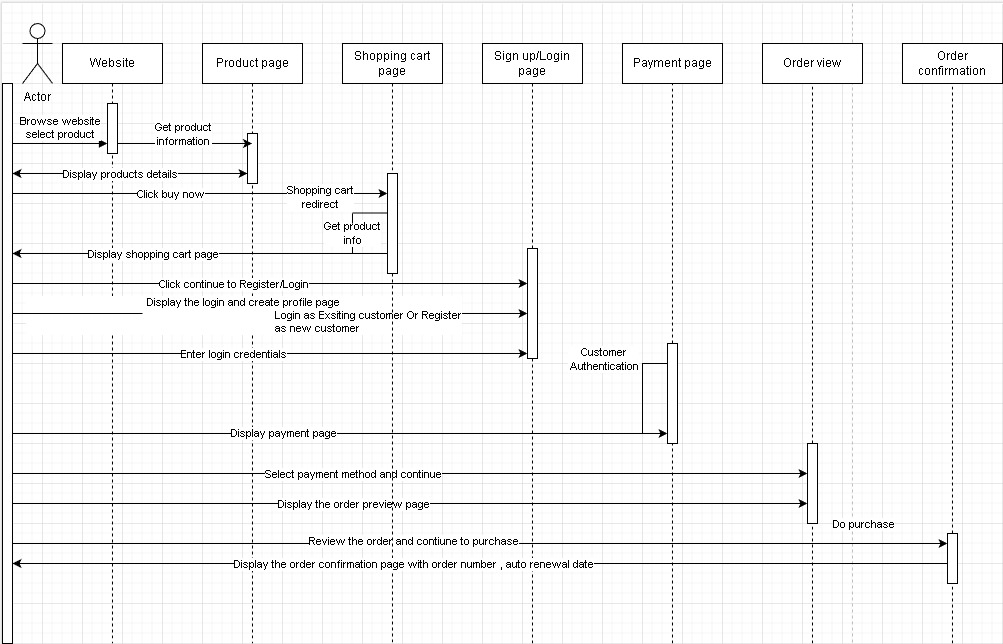
## 7.1 Login



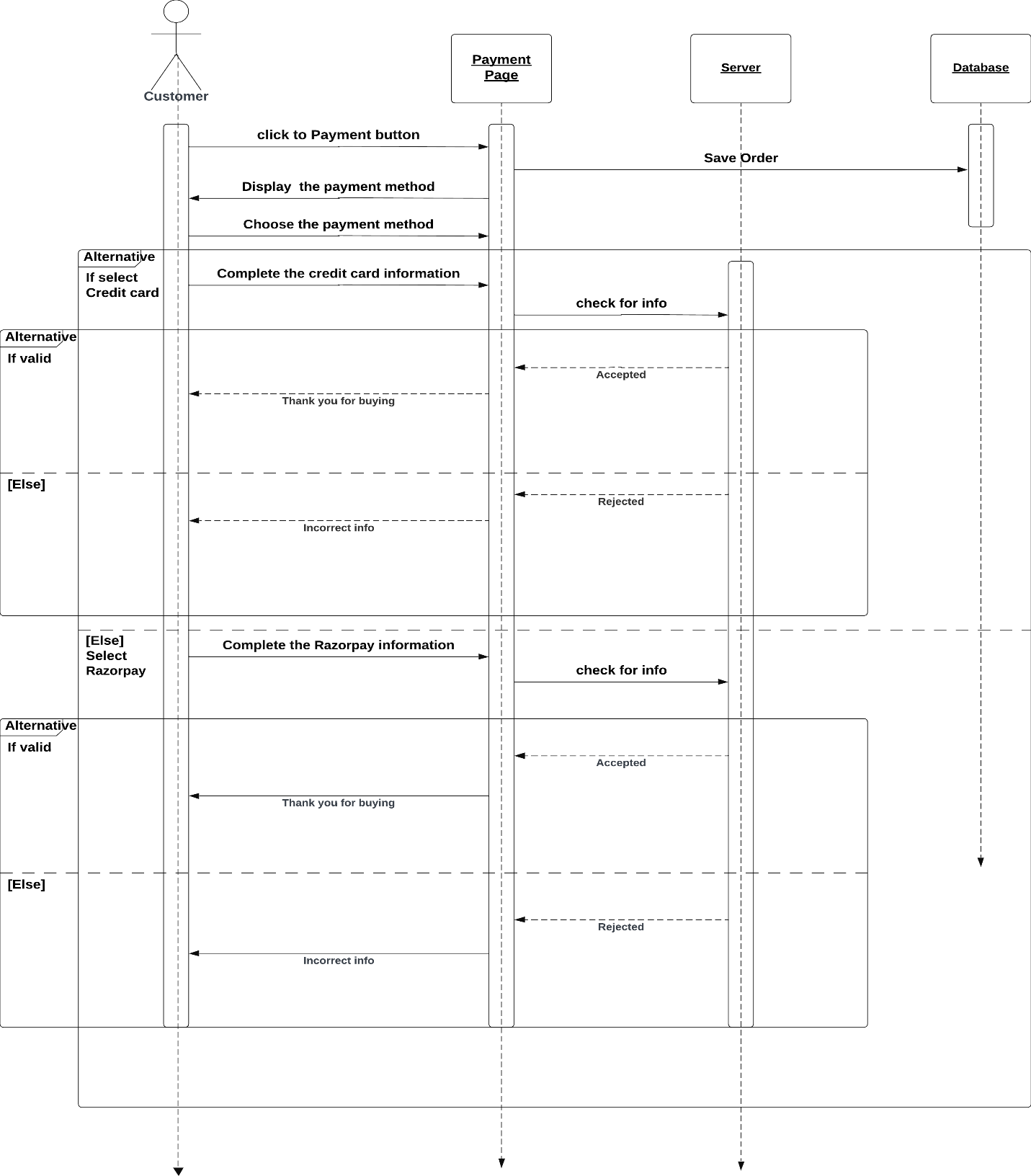
## 7.2 Registration



## 7.3 General Sequence Diagram

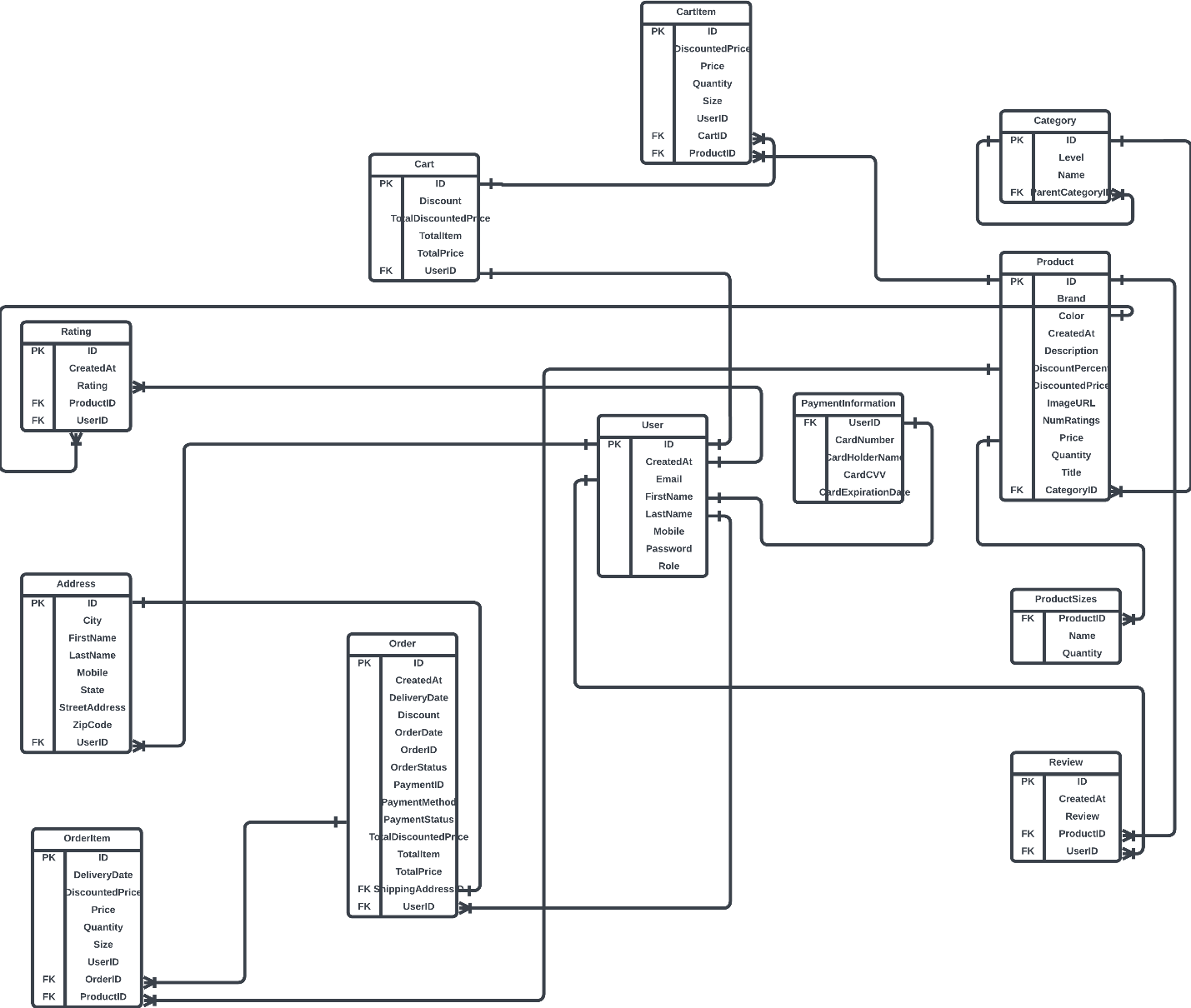


## 7.4 Payment processing

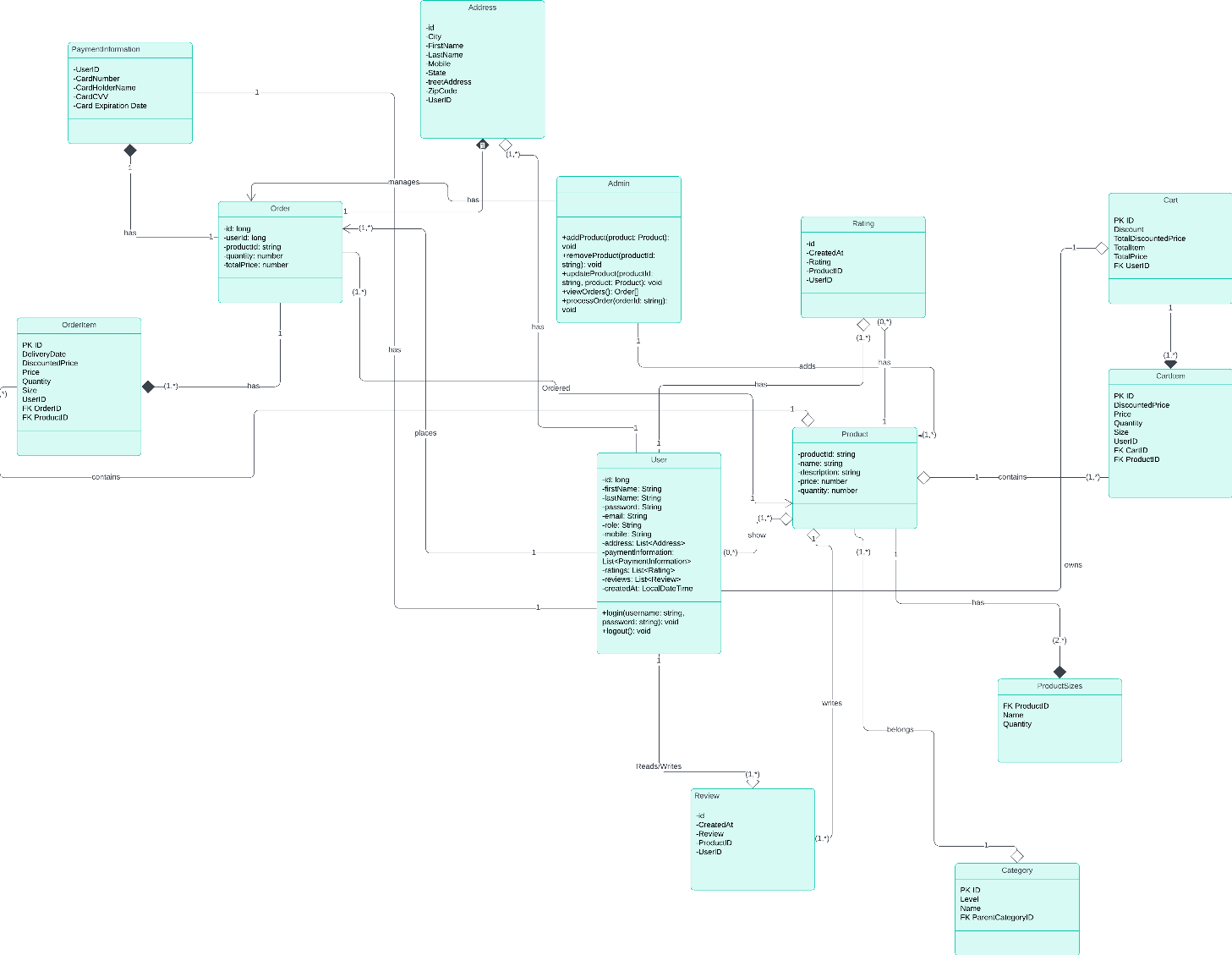


# **8.0 Database Specification**

## 8.1 Logical ERD



# **9.0 Class Diagram**



# **OCL**

-- Constraint: User must have a unique email

context User

inv: User.allInstances()->forAll(u1, u2 | u1 <> u2 implies u1.email <> u2.email)

-- Constraint: PaymentInformation must belong to an existing User

context PaymentInformation

inv: self.userID <> null

-- Constraint: CardExpirationDate must be in the future

context PaymentInformation

inv: self.CardExpirationDate > Date.today()

-- Constraint: Order must have a positive total price

context Order

inv: self.totalPrice > 0

-- Constraint: Rating must be between 1 and 5

context Rating

inv: self.rating >= 1 and self.rating <= 5

-- Constraint: Cart must have a valid total discounted price

context Cart

inv: self.totalDiscountedPrice >= 0

-- Constraint: OrderItem must have a positive quantity

context OrderItem

inv: self.quantity > 0

-- Constraint: Review must belong to an existing User and Product

context Review

inv: self.userID <> null and self.productID <> null

-- Constraint: Product must have a positive price

context Product

inv: self.price > 0

-- Constraint: Product must have a valid discount percent

context Product

inv: self.discountPercent >= 0 and self.discountPercent <= 100

-- Constraint: User's mobile number must be unique

context User

inv: User.allInstances()->forAll(u1, u2 | u1 <> u2 implies u1.mobile <> u2.mobile)

-- Constraint: Address must belong to an existing User

context Address

inv: self.userID <> null

-- Constraint: Cart must have a valid total price

context Cart

inv: self.totalPrice >= 0

-- Constraint: Order must have a valid total discounted price

context Order

inv: self.totalDiscountedPrice >= 0

-- Constraint: OrderItem's price must be greater than or equal to its discounted price

context OrderItem

inv: self.price >= self.discountedPrice

-- Constraint: Product's discounted price must be less than or equal to its regular price

context Product

inv: self.discountedPrice <= self.price

-- Constraint: Product's quantity must be greater than or equal to 0

context Product

inv: self.quantity >= 0

-- Constraint: Product's number of ratings must be greater than or equal to 0

context Product

inv: self.numRatings >= 0

-- Constraint: ProductSizes quantity must be greater than 0

context ProductSizes

inv: self.quantity > 0

-- Constraint: Review's product and user must exist

context Review

inv: self.productID <> null and self.userID <> null

-- Constraint: Review's rating must be between 1 and 5

context Review

inv: self.rating >= 1 and self.rating <= 5

-- Constraint: Category's level must be greater than 0

context Category

inv: self.level > 0

-- Constraint: Category's parent category must exist

context Category

inv: self.parentCategoryID <> null implies Category.allInstances()->exists(c | c.ID = self.parentCategoryID)

-- Constraint: Category's name must be unique within its parent category

context Category

inv: Category.allInstances()->forAll(c1, c2 | c1.parentCategoryID = c2.parentCategoryID implies c1 <> c2 implies c1.name <> c2.name)

-- Constraint: Each Order must have exactly one Customer

context Order

inv: self.customer <> null

context User

inv: self.cart->exist() = 1

context CartItem

inv: self.cart <> null

-- Constraint: Each Customer can place zero or more Orders

context Customer

inv: self.orders->notEmpty() or Order.allInstances()->exists(o | o.customer = self)

-- Constraint: Each Order must have at least one Product

context Order

inv: self.products->notEmpty()

-- Constraint: Each Product must be associated with at least one Order

context Product

inv: self.orders->notEmpty()

-- Constraint: Each Product must belong to exactly one Category

context Product

inv: self.category <> null

-- Constraint: Each Category can contain zero or more Products

context Category

inv: self.products->notEmpty() or Product.allInstances()->exists(p | p.category = self)

-- Constraint: Each Order must have a valid ShippingAddress

context Order

inv: self.shippingAddress <> null

-- Constraint: Each ShippingAddress must belong to exactly one Customer

context ShippingAddress

inv: self.customer <> null

-- Constraint: Each Customer can have zero or more ShippingAddresses

context Customer

inv: self.shippingAddresses->notEmpty() or ShippingAddress.allInstances()->exists(sa | sa.customer = self)

-- Constraint: Each Product can have zero or more Reviews

context Product

inv: self.reviews->notEmpty() or Review.allInstances()->exists(r | r.product = self)