项目文档

# Functional Requirement

# 1. Functional Requirements  
  
## 1.1 Customer Registration Function   
Function ID: FR-01   
Description: Allow new customers to create an account by providing their personal information. The system must validate input data and send a confirmation email.   
Input: Name, Email, Password, Address, PhoneNumber   
Output: New Customer entity, Confirmation email, Login page redirect or confirmation page   
  
## 1.2 Customer Login Function   
Function ID: FR-02   
Description: Allow registered customers to log in using their email and password. The system must verify credentials and initiate a session.   
Input: Email, Password   
Output: Active session, Dashboard or intended page redirect, Login event log   
  
## 1.3 Customer Logout Function   
Function ID: FR-03   
Description: Allow customers to terminate their active session securely. The system must log the logout event.   
Input: Logout request   
Output: Session termination, Home page or login page redirect, Logout event log   
  
## 1.4 View Product Details Function   
Function ID: FR-04   
Description: Allow customers to view detailed information about a product, including name, description, price, availability, and images.   
Input: Product ID   
Output: Product details display, View event log, API-related data if configured   
  
## 1.5 Search Products Function   
Function ID: FR-05   
Description: Allow customers to search for products using keywords, categories, or other attributes. The system must return relevant results.   
Input: Search query (e.g., product name, keyword, category)   
Output: List of matching Product entities, Search event log, Plugin-enhanced results if configured   
  
## 1.6 Add Product to Cart Function   
Function ID: FR-06   
Description: Allow customers to add a product to their shopping cart. The system must update the cart and log the action.   
Input: Product ID, Quantity   
Output: Updated ShoppingCart entity, Cart display update, Add event log, Plugin trigger if configured   
  
## 1.7 Remove Product from Cart Function   
Function ID: FR-07   
Description: Allow customers to remove a product from their cart. The system must update the cart and log the action.   
Input: Product ID or CartItem ID   
Output: Updated ShoppingCart entity, Cart display update, Remove event log, Plugin trigger if configured   
  
## 1.8 Place Order Function   
Function ID: FR-08   
Description: Allow customers to complete the checkout process by selecting a payment method and confirming the order. The system must create an Order entity, process payment, and send a confirmation email.   
Input: Cart items, Payment method selection, Shipping address, Customer information   
Output: New Order entity, Updated Product inventory, Confirmation email, Order event log   
  
## 1.9 View Order History Function   
Function ID: FR-09   
Description: Allow customers to view a list of their previous orders with summary details. The system must retrieve Order entities and log the access.   
Input: Customer ID   
Output: List of Order entities, Order history access log, Order details if selected   
  
## 1.10 Cancel Order Function   
Function ID: FR-10   
Description: Allow customers to cancel eligible orders. The system must update the Order status and initiate a refund if necessary.   
Input: Order ID   
Output: Updated Order status to "Cancelled", Refund initiated if paid, Cancellation event log   
  
## 1.11 Update Order Information Function   
Function ID: FR-11   
Description: Allow customers to modify eligible order details such as product quantity or shipping address. The system must validate changes and update the Order entity.   
Input: Order ID, Updated information (e.g., quantity, address)   
Output: Updated Order entity, Order update event log, Plugin trigger if configured   
  
## 1.12 View Order Confirmation Email Function   
Function ID: FR-12   
Description: Allow customers to view the content of their order confirmation email. The system must retrieve and display the email content.   
Input: Order ID or Email ID   
Output: Email content display, Email view event log, Order details if linked   
  
## 1.13 Administrator Login Function   
Function ID: FR-13   
Description: Allow administrators to log in with their credentials. The system must verify the username and password and create an admin session.   
Input: Username, Password   
Output: Active admin session, Admin dashboard redirect, Login event log   
  
## 1.14 Administrator Logout Function   
Function ID: FR-14   
Description: Allow administrators to securely log out of the system. The system must terminate the admin session and log the event.   
Input: Logout request   
Output: Session termination, Admin login page or main site redirect, Logout event log   
  
## 1.15 Add Product Function   
Function ID: FR-15   
Description: Allow administrators to add new products to the system. The system must validate input data and store the new Product entity.   
Input: Product name, Description, Price, Category, Image, Stock quantity   
Output: New Product entity, Updated product list, Product addition event log   
  
## 1.16 Update Product Information Function   
Function ID: FR-16   
Description: Allow administrators to modify existing product details. The system must validate the input and update the Product entity in the database.   
Input: Product ID, Updated product details (e.g., name, price, description)   
Output: Updated Product entity, Updated product list, Product update event log   
  
## 1.17 Delete Product Function   
Function ID: FR-17   
Description: Allow administrators to delete an existing product. The system must confirm the deletion and update the inventory accordingly.   
Input: Product ID   
Output: Deleted Product entity, Updated inventory, Product deletion event log   
  
## 1.18 View Product Inventory Function   
Function ID: FR-18   
Description: Allow administrators to view the current inventory levels of all products. The system must retrieve and display the inventory data.   
Input: Inventory filter/sort criteria (e.g., category, status)   
Output: Product inventory list, Inventory view event log, Plugin sync if configured   
  
## 1.19 View Order Details Function   
Function ID: FR-19   
Description: Allow customers and administrators to view detailed information about a specific order. The system must retrieve the Order entity and associated data.   
Input: Order ID   
Output: Order details display, Order view event log, Plugin-related data if available   
  
## 1.20 Plugin Registration Function   
Function ID: FR-20   
Description: Allow administrators to register new plugins. The system must validate the plugin and store the Plugin entity.   
Input: Plugin name, Version, Description, Author, Plugin file or API endpoint   
Output: New Plugin entity, Updated plugin list, Plugin registration event log   
  
## 1.21 Plugin Configuration Function   
Function ID: FR-21   
Description: Allow administrators to configure plugin settings. The system must update the PluginConfiguration entity and log the action.   
Input: Plugin ID, Configuration name, Configuration value   
Output: Updated PluginConfiguration entity, Plugin update if applicable, Configuration event log   
  
## 1.22 Plugin Deletion Function   
Function ID: FR-22   
Description: Allow administrators to delete an existing plugin. The system must confirm the deletion and log the event.   
Input: Plugin ID   
Output: Deleted Plugin entity, Updated plugin list, Plugin deletion event log   
  
## 1.23 Manage Payment Methods Function   
Function ID: FR-23   
Description: Allow customers to view, add, update, or delete their saved payment methods. The system must manage the Payment entity and synchronize with payment gateways if configured.   
Input: Customer ID, Payment method details (e.g., card info, gateway settings)   
Output: Updated Payment entity, Payment method list, Payment method event log   
  
## 1.24 View Payment Details Function   
Function ID: FR-24   
Description: Allow customers to view detailed information about their payment transactions. The system must retrieve and display the Payment entity.   
Input: Payment ID or Order ID   
Output: Payment details display, Payment view event log, Plugin-related data if available   
  
## 1.25 View Order Status Function   
Function ID: FR-25   
Description: Allow customers to view the current status of their orders. The system must retrieve the OrderStatus entity and display it along with any related tracking information.   
Input: Order ID   
Output: Order status display, Status event log, Plugin-related data if available   
  
## 1.26 Access API Documentation Function   
Function ID: FR-26   
Description: Allow administrators and developers to access the API documentation for plugins. The system must retrieve and display the documentation from the Documentation module.   
Input: Plugin ID or Documentation ID   
Output: API documentation display, Access event log, Plugin-related data if configured   
  
## 1.27 Update Plugin Documentation Function   
Function ID: FR-27   
Description: Allow administrators or developers to update the API documentation for a plugin. The system must validate the updated content and store it in the Documentation module.   
Input: Plugin ID, Updated documentation content   
Output: Updated Documentation entity, Documentation update event log, Plugin update if required   
  
## 1.28 Manage Payment Method Function   
Function ID: FR-28   
Description: Allow administrators to add, update, or delete payment methods available in the system. The system must manage the PaymentMethod entity and synchronize with payment gateways if configured.   
Input: Payment method ID or configuration details   
Output: Updated PaymentMethod entity, Payment method list, Plugin sync if configured, Event log

# External Description

# 2. External Interfaces  
  
## 2.1 User Interface Output  
  
The system interacts with users through a series of well-defined user interfaces that support customer and administrator actions. These interfaces are primarily web-based and designed to be intuitive and accessible on various devices, including desktops, tablets, and mobile phones. The following are the key user interfaces and their interactions:  
  
### 2.1.1 Customer Account Creation Interface   
\*\*Description:\*\* A form-based interface where new customers can input their personal information (Name, Email, Password, Address, PhoneNumber) to create an account.   
\*\*Interaction:\*\* The system validates the input data and sends a confirmation email. After successful registration, the user is redirected to the login page or a confirmation page.  
  
### 2.1.2 Customer Login Interface   
\*\*Description:\*\* A login form where registered customers can enter their Email and Password to authenticate.   
\*\*Interaction:\*\* The system verifies the credentials and initiates a session. If successful, the user is redirected to the Dashboard or the intended page.  
  
### 2.1.3 Customer Logout Interface   
\*\*Description:\*\* A button or link within the customer interface that allows users to terminate their active session.   
\*\*Interaction:\*\* The system logs the logout event and redirects the user to the Home page or the login page.  
  
### 2.1.4 Product Details Interface   
\*\*Description:\*\* A display interface for showing detailed product information (Name, Description, Price, Availability, Images).   
\*\*Interaction:\*\* The system retrieves product data based on the Product ID and displays it to the user. A View event is logged for analytics.  
  
### 2.1.5 Product Search Interface   
\*\*Description:\*\* A search bar or form where users can enter keywords, categories, or attributes to find products.   
\*\*Interaction:\*\* The system processes the search query and displays a list of matching products. A Search event is logged for tracking and analytics.  
  
### 2.1.6 Shopping Cart Interface   
\*\*Description:\*\* A user interface for managing the shopping cart, including adding and removing products.   
\*\*Interaction:\*\* The system allows users to add a Product ID and Quantity to the cart. It updates the cart display and logs the Add event. Similarly, the system allows users to remove products by Product ID or CartItem ID, updating the cart and logging the Remove event.  
  
### 2.1.7 Order Placement Interface   
\*\*Description:\*\* A checkout interface where customers can confirm their order, select a payment method, and enter a shipping address.   
\*\*Interaction:\*\* The system processes the cart items, payment method selection, shipping address, and customer information to create a new Order entity. It updates the product inventory, logs the Order event, and sends a confirmation email.  
  
### 2.1.8 Order History Interface   
\*\*Description:\*\* A list view interface for displaying the customer's previous orders with summary details.   
\*\*Interaction:\*\* The system retrieves a list of Order entities based on the Customer ID and displays them. It logs the Order history access event.  
  
### 2.1.9 Order Cancellation Interface   
\*\*Description:\*\* An interface where customers can cancel eligible orders.   
\*\*Interaction:\*\* The system updates the Order status to "Cancelled" and initiates a refund if applicable. It logs the Cancellation event.  
  
### 2.1.10 Order Information Update Interface   
\*\*Description:\*\* A form where customers can modify eligible order details such as product quantity or shipping address.   
\*\*Interaction:\*\* The system validates the changes and updates the Order entity accordingly. It logs the Order update event and triggers a Plugin if configured.  
  
### 2.1.11 Order Confirmation Email Interface   
\*\*Description:\*\* A display interface for showing the content of the order confirmation email.   
\*\*Interaction:\*\* The system retrieves and displays the email content based on the Order ID or Email ID. It logs the Email view event and may show Order details if linked.  
  
### 2.1.12 Administrator Login Interface   
\*\*Description:\*\* A login form for administrators to input their Username and Password for authentication.   
\*\*Interaction:\*\* The system verifies the credentials and creates an admin session. If successful, the user is redirected to the Admin Dashboard.  
  
### 2.1.13 Administrator Logout Interface   
\*\*Description:\*\* A button or link for administrators to terminate their active session.   
\*\*Interaction:\*\* The system logs the logout event and redirects the user to the Admin login page or the main site.  
  
### 2.1.14 Product Management Interface   
\*\*Description:\*\* An interface for administrators to add, update, or delete products.   
\*\*Interaction:\*\* The system validates the input data and updates the Product entity in the database. It logs the Product addition, update, or deletion events and updates the product list.  
  
### 2.1.15 Plugin Management Interface   
\*\*Description:\*\* An interface for administrators to register, configure, or delete plugins.   
\*\*Interaction:\*\* The system validates the plugin and its configuration and updates the Plugin and PluginConfiguration entities accordingly. It logs the Plugin registration, configuration, or deletion events and updates the plugin list.  
  
### 2.1.16 Payment Method Management Interface   
\*\*Description:\*\* An interface for customers and administrators to manage payment methods (view, add, update, delete).   
\*\*Interaction:\*\* The system processes the Payment method details and synchronizes with payment gateways if configured. It updates the Payment or PaymentMethod entities and logs the event.  
  
### 2.1.17 Order Status Interface   
\*\*Description:\*\* An interface for customers to view the current status of their orders.   
\*\*Interaction:\*\* The system retrieves the OrderStatus entity and displays it along with any related tracking information. It logs the Status event and may include Plugin-related data if available.  
  
### 2.1.18 API Documentation Interface   
\*\*Description:\*\* An interface for administrators and developers to access or update API documentation for plugins.   
\*\*Interaction:\*\* The system retrieves or updates the Documentation entity based on the Plugin ID or Documentation ID. It logs the Access or Update event and may include Plugin-related data if configured.  
  
---  
  
## 2.2 Hardware Interface Output  
  
The system does not have direct hardware dependencies or require specialized hardware for its core functionalities. All interactions with hardware are indirect and occur through standard software interfaces or user interfaces. However, the following hardware-related considerations are relevant:  
  
### 2.2.1 Mobile Device Compatibility   
\*\*Description:\*\* The system supports interactions via mobile devices.   
\*\*Interaction:\*\* The user interfaces are responsive and optimized for touch-based navigation. No direct hardware interface is required, but the system must function correctly on various mobile hardware platforms (iOS, Android).  
  
### 2.2.2 Point of Sale (POS) Integration (Optional)   
\*\*Description:\*\* If the system is integrated with a physical store, hardware interfaces may be required for POS devices (e.g., barcode scanners, payment terminals).   
\*\*Interaction:\*\* The system may communicate with POS hardware via serial or USB connections to process product scans or payment transactions. This is not a core requirement and is considered optional.  
  
---  
  
## 2.3 Software Interface Output  
  
The system interacts with various software components, including databases, third-party APIs, and internal modules. These interfaces are essential for storing, retrieving, and processing data. The following are the key software interfaces:  
  
### 2.3.1 Customer Database   
\*\*Description:\*\* A database storing customer entities (Name, Email, Password, Address, PhoneNumber).   
\*\*Interaction:\*\* The system reads and writes customer data to this database during registration, login, and other user-related functions. It supports queries by Customer ID or Email.  
  
### 2.3.2 Product Database   
\*\*Description:\*\* A database storing product entities (Name, Description, Price, Category, Image, Stock Quantity).   
\*\*Interaction:\*\* The system retrieves product data based on Product ID for display and updates inventory levels after order placement or product deletion.  
  
### 2.3.3 Order Database   
\*\*Description:\*\* A database storing order entities (Cart items, Payment method, Shipping address, Customer information).   
\*\*Interaction:\*\* The system creates new orders, retrieves order details, and updates order status (e.g., to "Cancelled"). It also logs order events for tracking and reporting.  
  
### 2.3.4 Plugin Database   
\*\*Description:\*\* A database storing plugin entities (Plugin name, Version, Description, Author, Plugin file or API endpoint) and their configurations.   
\*\*Interaction:\*\* The system reads and writes plugin data to manage plugin registration, configuration, and deletion. It also retrieves and updates the PluginConfiguration entity.  
  
### 2.3.5 Payment Gateway Integration   
\*\*Description:\*\* Integration with third-party payment gateways (e.g., PayPal, Stripe, Alipay).   
\*\*Interaction:\*\* The system communicates with payment gateways to process payments and manage refunds. It may update the PaymentMethod entity and synchronize with gateway systems.  
  
### 2.3.6 Email Service Interface   
\*\*Description:\*\* An external email service for sending and retrieving confirmation and transactional emails.   
\*\*Interaction:\*\* The system sends confirmation emails to customers after registration and order placement. It also retrieves email content for display in the Order Confirmation Email interface.  
  
### 2.3.7 API Communication Interface   
\*\*Description:\*\* A set of APIs for interacting with plugins and external systems.   
\*\*Interaction:\*\* The system communicates with plugins via APIs for enhanced functionality (e.g., search, cart updates, order tracking). It retrieves data from the Documentation module for API documentation.  
  
---  
  
## 2.4 Communication Interface Output  
  
The system communicates with external entities via various protocols and methods, including email and API calls. These communication interfaces are used to exchange data with users, payment gateways, and plugins.  
  
### 2.4.1 Email Notification Interface   
\*\*Description:\*\* A communication interface for sending confirmation and transactional emails to users.   
\*\*Interaction:\*\* The system sends emails to customers for account confirmation and order confirmation. It uses an external email service and logs the email events (e.g., send, view).  
  
### 2.4.2 Web Browsing Interface   
\*\*Description:\*\* The system is accessed via web browsers (e.g., Chrome, Firefox, Safari).   
\*\*Interaction:\*\* Users and administrators interact with the system through HTTP/HTTPS protocols. All web-based interfaces are designed for compatibility with modern browsers.  
  
### 2.4.3 Plugin API Communication Interface   
\*\*Description:\*\* A RESTful or GraphQL API for communicating with registered plugins.   
\*\*Interaction:\*\* The system sends and receives data to and from plugins for functionalities such as enhanced search, cart updates, order tracking, and payment synchronization. Inputs and outputs are defined in the PluginConfiguration and Plugin entities.  
  
### 2.4.4 Payment Gateway API Communication Interface   
\*\*Description:\*\* An API for interacting with payment gateway services (e.g., Stripe, PayPal).   
\*\*Interaction:\*\* The system sends payment requests and receives transaction responses from payment gateways. It updates the Payment and PaymentMethod entities based on the API responses.  
  
---  
  
## 2.5 Summary of External Data Sources and Their Interfaces  
  
The following table summarizes the external data sources referenced in the functional requirements and the corresponding interfaces defined in this section:  
  
| \*\*External Data Source\*\* | \*\*Interface Type\*\* | \*\*Description\*\* |  
|----------------------------------------|----------------------------|----------------------------------------------------------------------------------|  
| Customer Database | Software Interface | Stores and retrieves customer entities for registration and login. |  
| Product Database | Software Interface | Stores and retrieves product entities for display and inventory management. |  
| Order Database | Software Interface | Stores and retrieves order entities for order placement and cancellation. |  
| Plugin Database | Software Interface | Stores and retrieves plugin entities and configurations for plugin management. |  
| Payment Gateway | Software Interface / API | Processes payments and refunds via third-party payment services. |  
| Email Service | Communication Interface | Sends and retrieves confirmation and transactional emails. |  
| API Documentation Module | Software Interface | Stores and retrieves API documentation for plugins. |  
| Plugin API (for enhanced functionality) | Communication Interface / API | Enables interaction with plugins for additional features. |  
  
All external data sources are consistently mapped to the appropriate interfaces, ensuring that developers and stakeholders can clearly understand the system's dependencies and interactions.

# Use Case

Use Case Name:   
Customer Registration   
  
Use Case ID: UC-01   
  
Actors:   
- Customer   
- Administrator   
- Email System   
- Plugin System   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer does not have an existing account in the system.   
3. The Administrator has configured the necessary plugins and email templates.   
  
Postconditions:   
1. A new customer account is successfully created in the system.   
2. The customer receives a confirmation email.   
3. The customer is logged in or redirected to a login page.   
  
Main Flow:   
1. The customer navigates to the registration page.   
2. The system displays the registration form with required fields (e.g., name, email, password, address).   
3. The customer fills in the form and submits it.   
4. The system validates the input data (e.g., checks for valid email format, password strength).   
5. The system checks if the email is already registered.   
6. If the email is unique, the system creates a new Customer entity in the database.   
7. The system triggers the Email Plugin to send a confirmation email to the customer.   
8. The system logs the event in the Documentation module for audit purposes.   
9. The system redirects the customer to a confirmation page or login page.   
  
Alternative Flow:   
1. If the system detects an invalid input (e.g., missing field, incorrect format), it displays an error message and prompts the customer to correct the information.   
2. If the email is already registered, the system displays an error message indicating that the email is in use and prompts the customer to enter a different email address.   
3. If the Email Plugin fails to send the confirmation email, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the customer cancels the registration process at any step, the system terminates the use case and redirects them to the home page.  
  
Use Case Name:   
Customer Login   
  
Use Case ID: UC-02   
  
Actors:   
- Customer   
- Email System   
- Plugin System   
- Administrator   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer has already registered and has a valid account.   
3. The customer has not logged in yet in the current session.   
4. The Administrator has configured the necessary plugins and authentication settings.   
  
Postconditions:   
1. The customer is successfully authenticated and logged into the system.   
2. The system logs the login event in the Documentation module for audit purposes.   
3. The customer is redirected to the dashboard or previously accessed page.   
  
Main Flow:   
1. The customer navigates to the login page.   
2. The system displays the login form with fields for email and password.   
3. The customer enters their registered email and password.   
4. The customer submits the login request.   
5. The system verifies the email and password against the Customer entity in the database.   
6. If the credentials are valid, the system creates a session for the customer.   
7. The system logs the login event in the Documentation module.   
8. The system redirects the customer to the dashboard or the intended page.   
  
Alternative Flow:   
1. If the system detects an invalid email format, it displays an error message and prompts the customer to correct it.   
2. If the password is incorrect, the system displays an error message and allows the customer to re-enter the password.   
3. If the email is not registered, the system displays an error message and prompts the customer to enter a valid email or register.   
4. If the login attempt fails multiple times, the system may trigger a Plugin (e.g., account lockout or CAPTCHA) for security purposes.   
5. If the Plugin System fails to handle a security check, the system logs the error and displays a message to the customer, offering to retry or contact support.   
6. If the customer cancels the login process, the system terminates the use case and redirects them to the home page.  
  
Use Case Name:   
Customer Logout   
  
Use Case ID: UC-03   
  
Actors:   
- Customer   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is currently logged in.   
3. The session management and security plugins are configured and active.   
  
Postconditions:   
1. The customer is successfully logged out of the system.   
2. The session is terminated.   
3. The logout event is logged in the Documentation module for audit purposes.   
  
Main Flow:   
1. The customer navigates to the account settings or logout option in the system.   
2. The system displays a confirmation prompt for the logout action.   
3. The customer confirms the logout request.   
4. The system terminates the active session for the Customer entity.   
5. The system triggers the Plugin System to perform any session cleanup or security checks.   
6. The system logs the logout event in the Documentation module.   
7. The system redirects the customer to the home page or login page.   
  
Alternative Flow:   
1. If the customer cancels the logout confirmation, the system terminates the use case and returns them to the previous page.   
2. If the Plugin System fails to perform a cleanup action, the system logs the error and continues with the logout process.   
3. If the session termination fails, the system displays an error message and prompts the customer to try again or contact support.  
  
Use Case Name:   
View Product Details   
  
Use Case ID: UC-04   
  
Actors:   
- Customer   
- Product System   
- Administrator   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is either logged in or browsing as a guest.   
3. The Product entity exists in the system and is available for viewing.   
4. The Administrator has configured product display settings and enabled the necessary plugins.   
  
Postconditions:   
1. The customer views the detailed information of the selected product.   
2. The system logs the view event in the Documentation module for audit and analytics purposes.   
3. The customer may be redirected to a related page (e.g., add to cart, product reviews).   
  
Main Flow:   
1. The customer browses the product catalog or searches for a product.   
2. The customer selects a specific product from the list.   
3. The system retrieves the Product entity from the database based on the selected ID.   
4. The system displays the product details (e.g., name, description, price, images, availability).   
5. The system logs the action in the Documentation module.   
6. The system may load additional data via API, if configured.   
7. The customer is redirected to the product details page or can take further actions (e.g., add to cart).   
  
Alternative Flow:   
1. If the system cannot find the selected Product entity, it displays an error message and redirects the customer to the product catalog.   
2. If the API call fails to load additional product data, the system displays a warning message but continues to show the available product details.   
3. If the product is out of stock, the system displays a message indicating unavailability and may suggest similar products.   
4. If the Plugin System fails to load a configured feature (e.g., reviews or ratings), the system logs the error and displays a simplified version of the product details.   
5. If the customer cancels or navigates away from the product details page, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Search Products   
  
Use Case ID: UC-05   
  
Actors:   
- Customer   
- Product System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is either logged in or browsing as a guest.   
3. The Product System is operational and contains at least one Product entity.   
4. The Administrator has configured the search functionality and any associated plugins.   
  
Postconditions:   
1. The customer receives a list of relevant products based on their search criteria.   
2. The system logs the search event in the Documentation module for audit and analytics purposes.   
3. The customer can choose to view a product's details or add it to the cart.   
  
Main Flow:   
1. The customer navigates to the search page or uses the search bar on the homepage.   
2. The system displays a search input field.   
3. The customer enters a search query (e.g., product name, keyword, or category).   
4. The system processes the search query using the Product System.   
5. The system retrieves and displays a list of matching Product entities.   
6. The system logs the search event in the Documentation module.   
7. The customer selects a product from the search results to view its details or take other actions.   
  
Alternative Flow:   
1. If no products match the search query, the system displays a message indicating that no results were found and suggests refining the search.   
2. If the search query is incomplete or invalid, the system displays an error message and prompts the customer to correct it.   
3. If the Product System fails to retrieve data, the system logs the error and displays a message to the customer, offering to retry the search or contact support.   
4. If the Plugin System fails to enhance the search results (e.g., with filters or recommendations), the system logs the error and displays the search results without the plugin's enhancements.   
5. If the customer cancels the search or navigates away, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Add Product to Cart   
  
Use Case ID: UC-06   
  
Actors:   
- Customer   
- Product System   
- Order System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is either logged in or browsing as a guest.   
3. The selected Product entity exists in the system and is available for purchase.   
4. The Order System is operational and capable of managing cart items.   
5. The Administrator has configured the necessary plugins and cart functionality.   
  
Postconditions:   
1. The selected Product is added to the customer's cart.   
2. The system updates the cart in the Order entity.   
3. The system logs the action in the Documentation module for audit purposes.   
4. The customer is informed that the product was successfully added to the cart.   
  
Main Flow:   
1. The customer views a Product entity in the catalog or on the product details page.   
2. The system displays an "Add to Cart" button or option.   
3. The customer clicks the "Add to Cart" button.   
4. The system retrieves the product details from the Product entity.   
5. The system adds the product to the customer's Order (cart) entity.   
6. The system updates the cart display to reflect the new item.   
7. The system logs the action in the Documentation module.   
8. The system may trigger a Plugin (e.g., inventory update or recommendation engine).   
9. The system displays a confirmation message that the product was added to the cart.   
  
Alternative Flow:   
1. If the product is out of stock, the system displays a message and does not add the product to the cart.   
2. If the customer attempts to add the same product multiple times, the system updates the quantity instead of creating a new cart item.   
3. If the Order System fails to update the cart, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the Plugin System fails to execute an associated action (e.g., inventory sync), the system logs the error but continues with the cart update.   
5. If the customer cancels the action or navigates away, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Remove Product from Cart   
  
Use Case ID: UC-07   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is either logged in or browsing as a guest.   
3. The customer has at least one Product in their cart (Order entity).   
4. The Order System is operational and capable of managing cart items.   
5. The Administrator has configured the necessary plugins and cart functionality.   
  
Postconditions:   
1. The selected Product is successfully removed from the customer's cart.   
2. The system updates the cart in the Order entity.   
3. The system logs the action in the Documentation module for audit purposes.   
4. The customer is informed that the product was successfully removed from the cart.   
  
Main Flow:   
1. The customer views their cart on the Order details page.   
2. The system displays a list of products currently in the cart.   
3. The customer selects a product and clicks the "Remove from Cart" option.   
4. The system confirms the cart item to be removed by referencing the Order entity.   
5. The system removes the selected Product from the cart.   
6. The system updates the cart display to reflect the change.   
7. The system logs the removal event in the Documentation module.   
8. The system may trigger a Plugin for inventory update or other post-removal actions.   
9. The system displays a confirmation message that the product was removed from the cart.   
  
Alternative Flow:   
1. If the system cannot find the selected Product in the cart, it displays an error message and prompts the customer to verify the cart contents.   
2. If the customer cancels the removal action, the system terminates the use case and returns them to the cart view.   
3. If the Order System fails to update the cart, the system logs the error and displays a message to the customer, offering to retry the action or contact support.   
4. If the Plugin System fails to execute an associated action (e.g., inventory sync), the system logs the error but continues with the cart update.   
5. If the customer navigates away from the cart page, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Place Order   
  
Use Case ID: UC-08   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Payment System   
- Plugin System   
- Administrator   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in or has a guest session active.   
3. The customer has at least one Product in their cart (Order entity).   
4. The Payment System is operational and configured with supported payment methods.   
5. The Administrator has enabled the necessary plugins and configured order processing settings.   
  
Postconditions:   
1. A new Order entity is created and stored in the system.   
2. The Payment System processes the payment and updates the Order status.   
3. The system logs the order placement event in the Documentation module.   
4. The customer receives a confirmation message or email.   
5. The Plugin System may be triggered for additional order-related actions (e.g., inventory update, shipping calculation).   
  
Main Flow:   
1. The customer navigates to the checkout page.   
2. The system displays the items in the cart, along with total price and available payment methods.   
3. The customer selects a payment method and confirms the order details.   
4. The system validates the cart contents (e.g., product availability, pricing).   
5. The system initiates a transaction with the Payment System.   
6. The Payment System processes the payment and returns a success status.   
7. The system creates a new Order entity with the selected Products, customer information, and payment details.   
8. The system updates the Product inventory via the Product System.   
9. The system logs the order in the Documentation module.   
10. The system triggers the Email Plugin to send a confirmation email to the customer.   
11. The customer is redirected to an order confirmation page.   
  
Alternative Flow:   
1. If the cart is empty, the system displays an error message and redirects the customer to the product catalog.   
2. If the system detects an unavailable product in the cart, it displays a warning message and allows the customer to remove or replace the item.   
3. If the Payment System fails to process the transaction, the system logs the error and displays a message to the customer, offering to retry or cancel the order.   
4. If the Payment System returns an invalid or declined payment, the system displays an error message and prompts the customer to choose a different payment method.   
5. If the Email Plugin fails to send the confirmation email, the system logs the error and displays a message to the customer, offering to retry or contact support.   
6. If the Plugin System fails to update inventory or perform a related action, the system logs the error but continues with the order creation.   
7. If the customer cancels the order at any step, the system terminates the use case and returns them to the cart or home page.  
  
Use Case Name:   
View Order History   
  
Use Case ID: UC-09   
  
Actors:   
- Customer   
- Order System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The Order System contains at least one Order entity associated with the customer.   
4. The Administrator has configured the order history display and enabled relevant plugins.   
  
Postconditions:   
1. The customer views a list of their previous orders.   
2. The system logs the access to the order history in the Documentation module for audit purposes.   
3. The customer can select an order to view detailed information.   
  
Main Flow:   
1. The customer navigates to the "Order History" section in their account dashboard.   
2. The system retrieves the Order entities associated with the customer from the database.   
3. The system displays a list of orders, including order ID, date, status, total amount, and summary of products.   
4. The system logs the access to the order history in the Documentation module.   
5. The customer selects an order to view more details.   
6. The system loads and displays the full details of the selected Order (e.g., products, payment method, shipping address).   
7. The system may trigger a Plugin to display additional information (e.g., tracking details if applicable).   
8. The customer is informed that the selected order details are available.   
  
Alternative Flow:   
1. If the system finds no Order entities for the customer, it displays a message indicating that no order history is available.   
2. If the Plugin System fails to load additional details (e.g., tracking information), the system logs the error and displays the available order information.   
3. If the Order System fails to retrieve order data, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the customer cancels or navigates away from the order history page, the system terminates the use case and returns them to the dashboard or home page.  
  
Use Case Name:   
Cancel Order   
  
Use Case ID: UC-10   
  
Actors:   
- Customer   
- Order System   
- Payment System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The customer has an existing Order entity with a status that allows cancellation (e.g., pending, processing).   
4. The Payment System is operational and capable of handling cancellations (e.g., refunding payments).   
5. The Administrator has configured the necessary plugins and cancellation rules.   
  
Postconditions:   
1. The selected Order is successfully marked as cancelled in the system.   
2. The Payment System updates the payment status if applicable (e.g., initiates a refund).   
3. The system logs the cancellation event in the Documentation module for audit purposes.   
4. The customer is informed that the order has been cancelled.   
  
Main Flow:   
1. The customer navigates to the "Order History" or "My Orders" section.   
2. The system displays a list of orders, including those eligible for cancellation.   
3. The customer selects an order and clicks the "Cancel Order" option.   
4. The system confirms the eligibility of the selected Order for cancellation.   
5. The system prompts the customer for confirmation of the cancellation.   
6. The customer confirms the cancellation request.   
7. The system updates the Order status to "Cancelled" in the Order entity.   
8. If the order has been paid, the system triggers the Payment System to initiate a refund.   
9. The system logs the cancellation in the Documentation module.   
10. The system may trigger a Plugin to notify the Administrator or update inventory.   
11. The system displays a confirmation message that the order has been successfully cancelled.   
  
Alternative Flow:   
1. If the selected Order is not eligible for cancellation (e.g., already shipped), the system displays an error message and informs the customer of the reason.   
2. If the customer cancels the cancellation confirmation, the system terminates the use case and returns them to the order details page.   
3. If the Payment System fails to process a refund, the system logs the error and displays a message to the customer, offering to retry the action or contact support.   
4. If the Plugin System fails to execute a related action (e.g., notification to Administrator), the system logs the error but continues with the cancellation process.   
5. If the Order System fails to update the status, the system logs the error and displays a message to the customer, offering to retry the action or contact support.   
6. If the customer navigates away from the cancellation page, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Update Order Information   
  
Use Case ID: UC-11   
  
Actors:   
- Customer   
- Order System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The customer has an existing Order entity in the system with a status that allows modification (e.g., pending).   
4. The Order System is operational and capable of updating order details.   
5. The Administrator has configured the necessary plugins and order modification rules.   
  
Postconditions:   
1. The selected Order entity is successfully updated with the new information.   
2. The system logs the update event in the Documentation module for audit purposes.   
3. The customer is informed that the order has been successfully updated.   
4. The Plugin System may be triggered for related actions (e.g., inventory sync, notification).   
  
Main Flow:   
1. The customer navigates to the "Order History" or "My Orders" section.   
2. The system displays a list of orders, including those eligible for updates.   
3. The customer selects an order and clicks the "Edit Order" option.   
4. The system displays a form with editable fields (e.g., product quantity, shipping address).   
5. The customer modifies the desired information and submits the form.   
6. The system validates the updated data (e.g., product availability, valid address format).   
7. The system updates the relevant fields in the Order entity in the database.   
8. The system logs the update event in the Documentation module.   
9. The system may trigger the Plugin System to perform related actions (e.g., inventory update).   
10. The system displays a confirmation message that the order has been updated.   
  
Alternative Flow:   
1. If the selected Order is not eligible for updates (e.g., already shipped or completed), the system displays an error message and informs the customer of the reason.   
2. If the system detects invalid or missing information in the form (e.g., incorrect address format), it displays an error message and prompts the customer to correct the data.   
3. If the updated product quantity exceeds available stock, the system displays a warning and allows the customer to adjust the quantity.   
4. If the Order System fails to update the Order entity, the system logs the error and displays a message to the customer, offering to retry or contact support.   
5. If the Plugin System fails to execute a related action (e.g., inventory sync), the system logs the error but continues with the order update.   
6. If the customer cancels the update process at any step, the system terminates the use case and returns them to the order details page.  
  
Use Case Name:   
View Order Confirmation Email   
  
Use Case ID: UC-12   
  
Actors:   
- Customer   
- Email Plugin   
- Order System   
- Administrator   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer has placed an Order (UC-08) and the system has sent a confirmation email.   
3. The Email Plugin is configured and operational.   
4. The Administrator has set up the email templates and notification settings.   
5. The Documentation Module is active and logs email-related events.   
  
Postconditions:   
1. The customer is able to view the content of the order confirmation email.   
2. The system logs the email viewing event in the Documentation module.   
3. The customer can take further actions based on the email content (e.g., check order status, contact support).   
  
Main Flow:   
1. The customer navigates to their "Order History" or "Notifications" section.   
2. The system displays a list of past orders, including the order confirmation email as an associated document.   
3. The customer selects the order confirmation email to view it.   
4. The system retrieves the email content from the Order entity or Email Plugin storage.   
5. The system displays the email content to the customer (e.g., order summary, payment details, delivery information).   
6. The system logs the email view event in the Documentation module.   
7. The customer is informed that the order confirmation email is available for viewing.   
  
Alternative Flow:   
1. If the system cannot find the order confirmation email, it displays an error message and offers to resend the email.   
2. If the Email Plugin fails to retrieve the email content, the system logs the error and displays a message to the customer, offering to retry or contact support.   
3. If the Documentation Module fails to log the event, the system logs the error but continues to display the email content to the customer.   
4. If the customer cancels or navigates away from the email view, the system terminates the use case and returns them to the order details page.  
  
Use Case Name:   
Administrator Login   
  
Use Case ID: UC-13   
  
Actors:   
- Administrator   
- Authentication System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator has not logged in yet in the current session.   
3. The Administrator has a valid account in the system.   
4. The Authentication System is operational and configured.   
5. The Plugin System is active and configured for login-related actions.   
6. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The Administrator is successfully authenticated and logged into the system.   
2. The system logs the login event in the Documentation module for audit purposes.   
3. The Administrator is redirected to the admin dashboard.   
  
Main Flow:   
1. The Administrator navigates to the admin login page.   
2. The system displays the login form with fields for username and password.   
3. The Administrator enters their username and password.   
4. The Administrator submits the login request.   
5. The system verifies the credentials against the Administrator entity in the database.   
6. If the credentials are valid, the system creates an admin session.   
7. The system logs the login event in the Documentation module.   
8. The system redirects the Administrator to the admin dashboard.   
  
Alternative Flow:   
1. If the system detects an invalid username format, it displays an error message and prompts the Administrator to correct it.   
2. If the password is incorrect, the system displays an error message and allows the Administrator to re-enter the password.   
3. If the username is not registered, the system displays an error message and prompts the Administrator to enter a valid username or contact support.   
4. If the login attempt fails multiple times, the system may trigger a Plugin (e.g., account lockout or CAPTCHA) for security purposes.   
5. If the Plugin System fails to handle a security check, the system logs the error and displays a message to the Administrator, offering to retry or contact support.   
6. If the Administrator cancels the login process, the system terminates the use case and redirects them to the admin home page or main site.  
  
Use Case Name:   
Administrator Logout   
  
Use Case ID: UC-14   
  
Actors:   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is currently logged in.   
3. The session management and security plugins are configured and active.   
  
Postconditions:   
1. The Administrator is successfully logged out of the system.   
2. The admin session is terminated.   
3. The logout event is logged in the Documentation module for audit purposes.   
  
Main Flow:   
1. The Administrator navigates to the account settings or logout option in the admin dashboard.   
2. The system displays a confirmation prompt for the logout action.   
3. The Administrator confirms the logout request.   
4. The system terminates the active session for the Administrator entity.   
5. The system triggers the Plugin System to perform any session cleanup or security checks.   
6. The system logs the logout event in the Documentation module.   
7. The system redirects the Administrator to the admin login page or main site.   
  
Alternative Flow:   
1. If the Administrator cancels the logout confirmation, the system terminates the use case and returns them to the dashboard.   
2. If the Plugin System fails to perform a cleanup action, the system logs the error and continues with the logout process.   
3. If the session termination fails, the system displays an error message and prompts the Administrator to try again or contact support.  
  
Use Case Name:   
Add Product   
  
Use Case ID: UC-04   
  
Actors:   
- Administrator   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Product System is operational and capable of storing new Product entities.   
4. The Plugin System is configured and active for actions such as image processing or inventory sync.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. A new Product entity is successfully created and stored in the system.   
2. The system logs the product addition in the Documentation module for audit purposes.   
3. The Plugin System may be triggered for related actions (e.g., image optimization).   
4. The Administrator is informed that the product was successfully added.   
  
Main Flow:   
1. The Administrator navigates to the "Add Product" section in the admin dashboard.   
2. The system displays a form with required fields (e.g., product name, price, description, category, image upload).   
3. The Administrator fills in the product details and uploads any necessary media.   
4. The system validates the input data (e.g., checks for valid price format, required fields).   
5. The system triggers the Plugin System to process the uploaded media (e.g., image resizing).   
6. The system creates a new Product entity in the database with the provided information.   
7. The system logs the action in the Documentation module.   
8. The system displays a confirmation message that the product was successfully added.   
9. The Administrator is redirected to the product management page or the dashboard.   
  
Alternative Flow:   
1. If the system detects invalid input (e.g., missing fields, incorrect price format), it displays an error message and prompts the Administrator to correct the data.   
2. If the Plugin System fails to process media (e.g., image upload error), the system logs the error and continues with the product addition without the media.   
3. If the Product System fails to save the new Product entity, the system logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
4. If the Administrator cancels the action or navigates away from the form, the system terminates the use case and returns them to the product management page.  
  
Use Case Name:   
Update Product Information   
  
Use Case ID: UC-15   
  
Actors:   
- Administrator   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Product System contains at least one Product entity that can be updated.   
4. The Plugin System is configured and active for actions such as image processing or inventory sync.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The selected Product entity is successfully updated with the new information.   
2. The system logs the update event in the Documentation module for audit purposes.   
3. The Plugin System may be triggered for related actions (e.g., image optimization, inventory update).   
4. The Administrator is informed that the product was successfully updated.   
  
Main Flow:   
1. The Administrator navigates to the "Product Management" section in the admin dashboard.   
2. The system displays a list of existing Product entities.   
3. The Administrator selects a specific product to edit and clicks the "Edit Product" option.   
4. The system displays a form with editable fields (e.g., product name, price, description, category, image upload).   
5. The Administrator modifies the desired information and submits the form.   
6. The system validates the updated data (e.g., checks for valid price format, required fields).   
7. The system updates the relevant fields in the Product entity in the database.   
8. The system triggers the Plugin System to process any uploaded media or perform related actions (e.g., image resizing).   
9. The system logs the update event in the Documentation module.   
10. The system displays a confirmation message that the product was successfully updated.   
11. The Administrator is redirected to the product management page or the dashboard.   
  
Alternative Flow:   
1. If the system detects invalid input (e.g., missing fields, incorrect price format), it displays an error message and prompts the Administrator to correct the data.   
2. If the Plugin System fails to process media (e.g., image upload error), the system logs the error and continues with the product update without the media.   
3. If the Product System fails to update the Product entity, the system logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
4. If the Administrator cancels the update process at any step, the system terminates the use case and returns them to the product management page.  
  
Use Case Name:   
Delete Product   
  
Use Case ID: UC-16   
  
Actors:   
- Administrator   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Product System contains at least one Product entity that can be deleted.   
4. The Plugin System is configured and active for actions such as inventory sync or deletion confirmation.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The selected Product entity is successfully deleted from the system.   
2. The system logs the deletion event in the Documentation module for audit purposes.   
3. The Plugin System may be triggered for related actions (e.g., inventory update, notification).   
4. The Administrator is informed that the product was successfully deleted.   
  
Main Flow:   
1. The Administrator navigates to the "Product Management" section in the admin dashboard.   
2. The system displays a list of existing Product entities.   
3. The Administrator selects a specific product and clicks the "Delete Product" option.   
4. The system confirms the product to be deleted by referencing the Product entity in the database.   
5. The system prompts the Administrator for confirmation of the deletion.   
6. The Administrator confirms the deletion request.   
7. The system removes the selected Product entity from the database.   
8. The system updates the inventory via the Product System if necessary.   
9. The system logs the deletion event in the Documentation module.   
10. The system may trigger the Plugin System for related actions (e.g., inventory sync, notification).   
11. The system displays a confirmation message that the product was successfully deleted.   
12. The Administrator is redirected to the product management page or the dashboard.   
  
Alternative Flow:   
1. If the system cannot find the selected Product entity, it displays an error message and offers to return to the product list.   
2. If the Administrator cancels the deletion confirmation, the system terminates the use case and returns them to the product details page.   
3. If the Product System fails to update inventory, the system logs the error but continues with the deletion.   
4. If the Plugin System fails to execute a related action (e.g., inventory sync), the system logs the error but continues with the deletion process.   
5. If the system fails to delete the Product entity, it logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
6. If the Administrator navigates away from the deletion page, the system terminates the use case and returns them to the product management page.  
  
Use Case Name:   
View Product Inventory   
  
Use Case ID: UC-17   
  
Actors:   
- Administrator   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Product System is operational and contains at least one Product entity.   
4. The Plugin System is configured for inventory-related actions.   
5. The Documentation Module is active for logging events.   
  
Postconditions:   
1. The Administrator views a list of all products and their current inventory levels.   
2. The system logs the inventory view event in the Documentation module.   
3. The Plugin System may be triggered to update inventory or provide additional data.   
4. The Administrator is informed of the current inventory status.   
  
Main Flow:   
1. The Administrator navigates to the "Product Inventory" section in the admin dashboard.   
2. The system retrieves all Product entities from the database, including inventory levels.   
3. The system displays the product inventory in a tabular or list format, including product name, quantity, and status.   
4. The system logs the inventory view event in the Documentation module.   
5. The Administrator can filter or sort the inventory by category, status, or quantity.   
6. The system may trigger the Plugin System to sync inventory data with external sources if configured.   
7. The Administrator is informed of the current inventory status and can proceed to manage products.   
  
Alternative Flow:   
1. If the system cannot retrieve Product entities, it displays an error message and prompts the Administrator to retry or contact support.   
2. If the Plugin System fails to sync inventory data, the system logs the error but continues to display the current inventory.   
3. If the inventory data is outdated, the system displays a warning message and suggests manual updates.   
4. If the Administrator cancels or navigates away from the inventory page, the system terminates the use case and returns them to the dashboard.  
  
Use Case Name:   
View Order Details   
  
Use Case ID: UC-18   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The customer has at least one Order entity in the system.   
4. The Order System is operational and can retrieve order details.   
5. The Plugin System is configured for features such as order tracking or notifications.   
6. The Documentation Module is active for logging events.   
  
Postconditions:   
1. The customer views detailed information about a specific order.   
2. The system logs the access to the order details in the Documentation module.   
3. The Plugin System may be triggered to provide additional functionality (e.g., tracking information).   
4. The customer is informed of the order status and other relevant details.   
  
Main Flow:   
1. The customer navigates to the "Order History" or "My Orders" section.   
2. The system displays a list of orders with details such as order ID, date, and status.   
3. The customer selects an order from the list to view its details.   
4. The system retrieves the selected Order entity from the database.   
5. The system displays detailed information about the order, including products, prices, payment method, and shipping address.   
6. The system logs the access event in the Documentation module.   
7. The system may load additional information via the Plugin System (e.g., tracking status if available).   
8. The customer is informed that the order details are now available for review.   
  
Alternative Flow:   
1. If the system cannot find the selected Order entity, it displays an error message and offers to return to the order list.   
2. If the Plugin System fails to load additional data (e.g., tracking information), the system logs the error and continues to display the basic order details.   
3. If the Order System fails to retrieve the order, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the customer cancels or navigates away from the order details page, the system terminates the use case and returns them to the order history page.  
  
Use Case Name:   
Process Order   
  
Use Case ID: UC-19   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Payment System   
- Plugin System   
- Administrator   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in or has a guest session active.   
3. The customer has at least one Product in their cart (Order entity).   
4. The Payment System is configured with valid payment methods.   
5. The Administrator has enabled necessary plugins and configured order processing settings.   
6. The Product System confirms product availability for the items in the cart.   
  
Postconditions:   
1. The selected products are successfully processed and the Order entity is created.   
2. The Payment System confirms the transaction and updates the Order status.   
3. The system logs the order processing event in the Documentation module.   
4. The Plugin System may be triggered for related actions (e.g., inventory update, shipping notification).   
5. The customer receives a confirmation message or email.   
  
Main Flow:   
1. The customer navigates to the checkout page after reviewing the cart.   
2. The system displays the cart items, total price, and available payment methods.   
3. The customer selects a payment method and confirms the order.   
4. The system validates the cart items for availability and pricing.   
5. The system initiates the payment transaction through the Payment System.   
6. The Payment System processes the payment and returns a success status.   
7. The system creates a new Order entity with the customer details, product list, and payment information.   
8. The system updates the Product inventory using the Product System.   
9. The system logs the order processing event in the Documentation module.   
10. The system triggers the Email Plugin to send an order confirmation email to the customer.   
11. The customer is redirected to an order confirmation page.   
  
Alternative Flow:   
1. If the cart is empty, the system displays an error message and redirects the customer to the product catalog.   
2. If the system detects an out-of-stock product in the cart, it displays a warning and allows the customer to remove or replace the item.   
3. If the Payment System fails to process the transaction, the system logs the error and displays a message to the customer, offering to retry or cancel the order.   
4. If the payment is declined, the system displays an error message and prompts the customer to choose a different payment method.   
5. If the Email Plugin fails to send the confirmation email, the system logs the error and displays a message to the customer, offering to resend or contact support.   
6. If the Plugin System fails to update inventory or perform a related action, the system logs the error but continues with the order creation.   
7. If the customer cancels the order at any step, the system terminates the use case and returns them to the cart or home page.  
  
Use Case Name:   
Manage Payment Methods   
  
Use Case ID: UC-20   
  
Actors:   
- Customer   
- Payment System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The Payment System is operational and configured with supported payment methods.   
4. The Administrator has enabled and configured the necessary plugins for payment method management.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The customer can view, add, update, or delete their saved payment methods.   
2. The system updates the Payment entity in the database accordingly.   
3. The system logs all changes in the Documentation module for audit purposes.   
4. The Plugin System may be triggered for additional actions (e.g., payment gateway sync).   
  
Main Flow:   
1. The customer navigates to the "Payment Methods" section in their account settings.   
2. The system retrieves the Payment entities associated with the customer from the database.   
3. The system displays a list of saved payment methods (e.g., credit card, PayPal, etc.).   
4. The customer selects an option to add a new payment method.   
5. The system displays a form for entering payment details (e.g., card number, expiration date, name on card).   
6. The customer fills in the form and submits it.   
7. The system validates the payment information and checks for format correctness.   
8. The system saves the new Payment entity in the database.   
9. The system logs the addition of the payment method in the Documentation module.   
10. The system may trigger the Plugin System to synchronize with external payment gateways.   
11. The system displays a confirmation message that the payment method was successfully added.   
  
Alternative Flow:   
1. If the system detects invalid or missing payment information, it displays an error message and prompts the customer to correct the data.   
2. If the Payment System fails to validate the payment method, the system logs the error and displays a message to the customer, offering to retry or contact support.   
3. If the Plugin System fails to sync with the payment gateway, the system logs the error but continues to store the payment method locally.   
4. If the customer chooses to update an existing payment method, the system displays an editable form, validates the changes, and updates the Payment entity in the database.   
5. If the customer chooses to delete a payment method, the system prompts for confirmation, removes the Payment entity from the database, and logs the deletion.   
6. If the system cannot retrieve the Payment entities, it displays an error message and offers to retry or contact support.   
7. If the customer cancels the action or navigates away from the payment method page, the system terminates the use case and returns them to the account settings or home page.  
  
Use Case Name:   
View Payment Details   
  
Use Case ID: UC-21   
  
Actors:   
- Customer   
- Payment System   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The customer has at least one Payment entity associated with their account.   
4. The Payment System is operational and can retrieve payment information.   
5. The Plugin System is configured for features such as payment gateway integration or data enrichment.   
6. The Documentation Module is active for logging events.   
  
Postconditions:   
1. The customer views detailed information about a specific payment.   
2. The system logs the access to the payment details in the Documentation module.   
3. The Plugin System may be triggered to provide additional data (e.g., transaction status from a third-party gateway).   
4. The customer is informed of the payment method and transaction status.   
  
Main Flow:   
1. The customer navigates to the "Payment History" or "Payment Methods" section in their account.   
2. The system retrieves the relevant Payment entities from the database.   
3. The system displays a list of payments, including payment method, transaction status, amount, and date.   
4. The customer selects a specific payment to view its details.   
5. The system retrieves and displays detailed information about the selected Payment (e.g., transaction ID, payment date, status, and associated order).   
6. The system logs the access to the payment details in the Documentation module.   
7. The system may trigger the Plugin System to fetch additional transaction details from a payment gateway.   
8. The customer is informed that the payment details are available for review.   
  
Alternative Flow:   
1. If the system cannot find the selected Payment entity, it displays an error message and offers to return to the payment list.   
2. If the Plugin System fails to fetch additional data (e.g., gateway response), the system logs the error and continues to display the basic payment information.   
3. If the Payment System fails to retrieve the payment, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the customer cancels or navigates away from the payment details page, the system terminates the use case and returns them to the payment history or account settings.  
  
Use Case Name:   
Plugin Registration   
  
Use Case ID: UC-22   
  
Actors:   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Plugin System is operational and capable of accepting new plugins.   
4. The Documentation Module is active for logging events.   
5. The Administrator has access to plugin configuration settings.   
  
Postconditions:   
1. The selected Plugin is successfully registered and activated in the system.   
2. The system logs the plugin registration event in the Documentation module for audit purposes.   
3. The Plugin System is updated to include the new plugin functionality.   
4. The Administrator is informed that the plugin was successfully registered.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Management" section in the admin dashboard.   
2. The system displays a list of available plugins and an option to register a new plugin.   
3. The Administrator selects the "Register Plugin" option.   
4. The system displays a form for entering plugin details (e.g., name, version, configuration settings).   
5. The Administrator fills in the form and uploads the plugin file or provides the API endpoint.   
6. The system validates the plugin file or endpoint for compatibility and required metadata.   
7. The system registers the Plugin entity in the Plugin System.   
8. The system activates the plugin if applicable.   
9. The system logs the registration event in the Documentation module.   
10. The system displays a confirmation message that the plugin was successfully registered.   
11. The Administrator is redirected to the plugin management page.   
  
Alternative Flow:   
1. If the system detects an invalid or missing plugin file or endpoint, it displays an error message and prompts the Administrator to correct the data.   
2. If the Plugin System fails to register the plugin, the system logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
3. If the plugin is already registered, the system displays a message indicating the conflict and prompts the Administrator to choose a different plugin or version.   
4. If the Administrator cancels the registration process at any step, the system terminates the use case and returns them to the plugin management page.   
5. If the Documentation Module fails to log the event, the system logs the error but continues with the plugin registration.  
  
Use Case Name:   
Plugin Configuration   
  
Use Case ID: UC-23   
  
Actors:   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Plugin System is operational and contains at least one registered plugin.   
4. The Administrator has appropriate permissions to configure plugins.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The selected Plugin is successfully configured with the specified settings.   
2. The system logs the plugin configuration event in the Documentation module.   
3. The Plugin System updates the plugin behavior based on the new configuration.   
4. The Administrator is informed that the plugin configuration was successful.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Configuration" section in the admin dashboard.   
2. The system displays a list of registered Plugins with their current configuration status.   
3. The Administrator selects a specific Plugin for configuration.   
4. The system displays a configuration form with relevant settings (e.g., API keys, activation status, parameters).   
5. The Administrator modifies the plugin settings as needed and submits the form.   
6. The system validates the input data and checks for required configuration fields.   
7. The system updates the Plugin entity in the Plugin System with the new configuration.   
8. The system logs the configuration event in the Documentation module.   
9. The system displays a confirmation message that the plugin configuration was successful.   
10. The Administrator is redirected to the plugin management or dashboard page.   
  
Alternative Flow:   
1. If the system detects invalid or missing configuration data, it displays an error message and prompts the Administrator to correct the input.   
2. If the Plugin System fails to update the plugin configuration, the system logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
3. If the plugin is not registered, the system displays an error message and suggests registering the plugin first.   
4. If the Administrator cancels the configuration process at any step, the system terminates the use case and returns them to the plugin management page.   
5. If the Documentation Module fails to log the event, the system logs the error but continues with the plugin configuration.  
  
Use Case Name:   
Plugin Deletion   
  
Use Case ID: UC-24   
  
Actors:   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Plugin System contains at least one registered Plugin entity.   
4. The Administrator has the necessary permissions to delete plugins.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The selected Plugin is successfully deleted from the system.   
2. The system logs the plugin deletion event in the Documentation module for audit purposes.   
3. The Plugin System is updated to reflect the deletion.   
4. The Administrator is informed that the plugin was successfully deleted.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Management" section in the admin dashboard.   
2. The system displays a list of registered plugins.   
3. The Administrator selects a specific Plugin and clicks the "Delete Plugin" option.   
4. The system confirms the Plugin entity to be deleted.   
5. The system prompts the Administrator for confirmation of the deletion.   
6. The Administrator confirms the deletion.   
7. The system removes the selected Plugin from the Plugin System.   
8. The system logs the deletion event in the Documentation module.   
9. The system displays a confirmation message that the plugin was successfully deleted.   
10. The Administrator is redirected to the plugin management page.   
  
Alternative Flow:   
1. If the system cannot find the selected Plugin entity, it displays an error message and offers to return to the plugin list.   
2. If the Administrator cancels the deletion confirmation, the system terminates the use case and returns them to the plugin details page.   
3. If the Plugin System fails to delete the plugin, the system logs the error and displays a message to the Administrator, offering to retry the action or contact support.   
4. If the Documentation Module fails to log the event, the system logs the error but continues with the plugin deletion.   
5. If the plugin is in use by other system modules, the system displays a warning message and prevents deletion to avoid system instability.  
  
Use Case Name:   
Access API Documentation   
  
Use Case ID: UC-25   
  
Actors:   
- Administrator   
- Developer   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator or Developer.   
2. The Documentation Module is configured and contains API documentation.   
3. The Administrator or Developer has the necessary permissions to access the API documentation.   
4. The Plugin System is active and may provide additional documentation features.   
5. The API documentation is available in the system (either generated or manually uploaded).   
  
Postconditions:   
1. The user successfully views the API documentation.   
2. The system logs the access event in the Documentation module for audit and tracking purposes.   
3. The Plugin System may be triggered to provide additional documentation details or enhancements.   
4. The user is informed that the API documentation is available for review.   
  
Main Flow:   
1. The Administrator or Developer navigates to the "API Documentation" section in the system.   
2. The system retrieves the API documentation from the Documentation Module.   
3. The system displays the API documentation, including endpoints, parameters, and usage examples.   
4. The system logs the access event in the Documentation module.   
5. The system may load additional documentation content via the Plugin System if configured.   
6. The user is informed that the API documentation is available for viewing and use.   
  
Alternative Flow:   
1. If the system cannot find the requested API documentation, it displays an error message and suggests checking the documentation configuration.   
2. If the Plugin System fails to load additional documentation content, the system logs the error and continues to display the available documentation.   
3. If the Documentation Module fails to log the access event, the system logs the error but continues to display the API documentation.   
4. If the user cancels or navigates away from the documentation page, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
Update Plugin Documentation   
  
Use Case ID: UC-26   
  
Actors:   
- Administrator   
- Plugin System   
- Documentation Module   
- Developer   
  
Preconditions:   
1. The system is accessible by the Administrator or Developer.   
2. The Administrator or Developer is logged in.   
3. The selected Plugin exists in the Plugin System and is registered.   
4. The Documentation Module is operational and allows editing of plugin-related documentation.   
5. The Administrator has the necessary permissions to update plugin documentation.   
  
Postconditions:   
1. The Plugin’s documentation is successfully updated in the Documentation Module.   
2. The system logs the documentation update event for audit purposes.   
3. The Plugin System is notified of the update if required.   
4. The Administrator or Developer is informed that the documentation was successfully updated.   
  
Main Flow:   
1. The Administrator or Developer navigates to the "Plugin Documentation" section in the admin dashboard or documentation portal.   
2. The system displays a list of plugins with links to their associated documentation.   
3. The Administrator or Developer selects a specific Plugin and clicks the "Edit Documentation" option.   
4. The system retrieves the current documentation for the selected Plugin from the Documentation Module.   
5. The system displays an editable form with the documentation content.   
6. The Administrator or Developer modifies the documentation content as needed and submits the form.   
7. The system validates the updated documentation content (e.g., format, completeness).   
8. The system updates the Plugin’s documentation in the Documentation Module.   
9. The system logs the update event in the Documentation Module.   
10. The system displays a confirmation message that the plugin documentation has been updated.   
11. The Administrator or Developer is redirected to the plugin details or documentation page.   
  
Alternative Flow:   
1. If the system cannot find the selected Plugin or its documentation, it displays an error message and offers to return to the plugin list.   
2. If the documentation content is invalid or incomplete, the system displays an error message and prompts the Administrator or Developer to correct it.   
3. If the Documentation Module fails to update the documentation, the system logs the error and displays a message to the Administrator or Developer, offering to retry the action or contact support.   
4. If the Administrator or Developer cancels the update process, the system terminates the use case and returns them to the plugin details or documentation page.   
5. If the system fails to log the update event, it logs the error but continues to display the confirmation message that the documentation was updated.  
  
Use Case Name:   
Add to ShoppingCart   
  
Use Case ID: UC-27   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Customer.   
2. The Customer is either logged in or browsing as a guest.   
3. The selected Product entity exists in the system and is available for purchase.   
4. The ShoppingCart feature is enabled in the Order System.   
5. The Administrator has configured the Plugin System for cart-related actions (e.g., inventory sync, notifications).   
6. The Documentation Module is active for logging events.   
  
Postconditions:   
1. The selected Product is successfully added to the ShoppingCart (Order entity).   
2. The system updates the ShoppingCart in the Order entity.   
3. The system logs the action in the Documentation Module for audit and tracking purposes.   
4. The Customer is informed that the product has been added to the ShoppingCart.   
5. The Plugin System may be triggered to update inventory or provide additional feedback (e.g., cart size limits).   
  
Main Flow:   
1. The Customer views a Product entity on the product details page or in the catalog.   
2. The system displays an "Add to ShoppingCart" button or option.   
3. The Customer clicks the "Add to ShoppingCart" button.   
4. The system retrieves the product details from the Product entity.   
5. The system adds the product to the Customer’s ShoppingCart (a subset of the Order entity).   
6. The system updates the ShoppingCart display (e.g., cart icon with item count).   
7. The system logs the "Add to ShoppingCart" event in the Documentation Module.   
8. The system may trigger the Plugin System to perform actions such as inventory update or display a recommendation for similar items.   
9. The system displays a confirmation message indicating that the product has been added to the ShoppingCart.   
  
Alternative Flow:   
1. If the selected Product is out of stock, the system displays a warning message and does not add the product to the ShoppingCart.   
2. If the system detects that the same product is already in the ShoppingCart, it increases the quantity instead of adding a new item.   
3. If the ShoppingCart reaches a predefined capacity limit, the system displays a message and prevents further additions.   
4. If the Order System fails to update the ShoppingCart, the system logs the error and displays a message to the Customer, offering to retry the action or contact support.   
5. If the Plugin System fails to perform an associated action (e.g., inventory sync), the system logs the error but continues with the cart update.   
6. If the Customer cancels the action or navigates away from the page, the system terminates the use case and returns them to the previous view.  
  
Use Case Name:   
Remove from ShoppingCart   
  
Use Case ID: UC-28   
  
Actors:   
- Customer   
- Order System   
- Product System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Customer.   
2. The Customer is either logged in or browsing as a guest.   
3. The Customer has at least one Product in their ShoppingCart (Order entity).   
4. The Order System is operational and can manage cart items.   
5. The Plugin System is configured for cart-related actions (e.g., inventory sync, notifications).   
6. The Documentation Module is active for logging events.   
  
Postconditions:   
1. The selected Product is successfully removed from the ShoppingCart (Order entity).   
2. The system updates the ShoppingCart in the Order entity.   
3. The system logs the action in the Documentation Module for audit and tracking purposes.   
4. The Customer is informed that the product has been removed from the ShoppingCart.   
5. The Plugin System may be triggered to update inventory or provide additional feedback (e.g., restocking notifications).   
  
Main Flow:   
1. The Customer views their ShoppingCart on the Order details page or via the cart icon.   
2. The system displays a list of products currently in the ShoppingCart.   
3. The Customer selects a product and clicks the "Remove from ShoppingCart" option.   
4. The system confirms the cart item to be removed by referencing the Order entity.   
5. The system removes the selected Product from the ShoppingCart.   
6. The system updates the ShoppingCart display to reflect the change.   
7. The system logs the removal event in the Documentation Module.   
8. The system may trigger the Plugin System to perform actions such as inventory update or recommend similar items.   
9. The system displays a confirmation message indicating that the product has been removed from the ShoppingCart.   
  
Alternative Flow:   
1. If the system cannot find the selected Product in the ShoppingCart, it displays an error message and prompts the Customer to verify the cart contents.   
2. If the Customer cancels the removal action, the system terminates the use case and returns them to the cart view.   
3. If the Order System fails to update the ShoppingCart, the system logs the error and displays a message to the Customer, offering to retry the action or contact support.   
4. If the Plugin System fails to execute an associated action (e.g., inventory sync), the system logs the error but continues with the cart update.   
5. If the Customer navigates away from the cart page, the system terminates the use case and returns them to the previous page.  
  
Use Case Name:   
View Order Status   
  
Use Case ID: UC-29   
  
Actors:   
- Customer   
- Order System   
- OrderStatus Module   
- Administrator   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the customer.   
2. The customer is logged in.   
3. The customer has at least one Order entity in the system.   
4. The Order System is operational and can retrieve order status information.   
5. The OrderStatus Module is configured to provide status updates (e.g., "Processing", "Shipped", "Delivered", "Cancelled").   
6. The Plugin System is active and may be used to fetch real-time tracking information.   
7. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The customer views the current status of the selected order.   
2. The system logs the status view event in the Documentation Module for audit and analytics.   
3. The Plugin System may be triggered to provide additional status details (e.g., tracking number, shipping timeline).   
4. The customer is informed of the order's current status and any related information.   
  
Main Flow:   
1. The customer navigates to the "Order History" or "My Orders" section in their account.   
2. The system displays a list of orders with summary information including order ID and status.   
3. The customer selects a specific order to view its status.   
4. The system retrieves the OrderStatus entity for the selected order from the database.   
5. The system displays the current status of the order along with any additional details (e.g., estimated delivery time, shipping provider, tracking information).   
6. If the Plugin System is configured, the system may fetch real-time tracking data or other status-related information.   
7. The system logs the status view event in the Documentation Module.   
8. The customer is informed of the current order status and can proceed with other actions (e.g., contact support, view related email).   
  
Alternative Flow:   
1. If the system cannot find the selected OrderStatus entity, it displays an error message and offers to return to the order list.   
2. If the Plugin System fails to retrieve real-time tracking or status details, the system logs the error and continues to display the last known status from the system.   
3. If the Order System fails to retrieve the order, the system logs the error and displays a message to the customer, offering to retry or contact support.   
4. If the Documentation Module fails to log the event, the system logs the error but continues to display the order status.   
5. If the customer cancels or navigates away from the status view, the system terminates the use case and returns them to the order history or dashboard.  
  
Use Case Name:   
Manage Payment Method   
  
Use Case ID: UC-30   
  
Actors:   
- Administrator   
- Payment System   
- Plugin System   
- Documentation Module   
  
Preconditions:   
1. The system is accessible by the Administrator.   
2. The Administrator is logged in.   
3. The Payment System is operational and supports multiple payment methods.   
4. The Plugin System is configured and active for payment gateway integrations.   
5. The Documentation Module is available for logging events.   
  
Postconditions:   
1. The Administrator can add, update, or delete payment methods available in the system.   
2. The system updates the PaymentMethod entity in the database accordingly.   
3. The system logs all changes in the Documentation Module for audit purposes.   
4. The Plugin System may be triggered to synchronize with external payment gateways.   
5. The updated payment methods are available for customers during checkout.   
  
Main Flow:   
1. The Administrator navigates to the "Payment Method Management" section in the admin dashboard.   
2. The system displays a list of currently configured PaymentMethod entities.   
3. The Administrator selects an action: "Add", "Update", or "Delete" a payment method.   
4. If adding a new payment method, the system displays a form for entering the method name, description, and gateway configuration.   
5. The Administrator fills in the required details and submits the form.   
6. The system validates the input data and confirms the method is supported by the Plugin System.   
7. The system updates the PaymentMethod entity in the database.   
8. The system logs the action in the Documentation Module.   
9. The system may trigger the Plugin System to update or activate the payment gateway integration.   
10. The system displays a confirmation message that the payment method has been successfully managed.   
11. The Administrator is redirected to the payment method management page.   
  
Alternative Flow:   
1. If the system detects invalid input (e.g., missing fields, unsupported payment method), it displays an error message and prompts the Administrator to correct the data.   
2. If the Payment System fails to validate or apply the new method, the system logs the error and displays a message to the Administrator, offering to retry or contact support.   
3. If the Plugin System fails to synchronize with the external gateway, the system logs the error but continues with the database update.   
4. If the Administrator chooses to update an existing payment method, the system displays an editable form, validates the changes, and updates the PaymentMethod entity.   
5. If the Administrator chooses to delete a payment method, the system prompts for confirmation, removes the method from the database, and logs the deletion.   
6. If the selected payment method is in use by active orders, the system displays a warning and prevents deletion to avoid data inconsistency.   
7. If the system fails to log the event in the Documentation Module, it logs the error but continues with the payment method update.   
8. If the Administrator cancels the action or navigates away from the payment method management page, the system terminates the use case and returns them to the dashboard.