项目文档

# Functional Requirement

# 1. Functional Requirements  
  
## 1.1 Customer Registration Function   
\*\*Function ID\*\*: FR-01   
\*\*Description\*\*: A customer can create a new account by providing personal information, including a valid email address and phone number. The system must validate the input data and send a confirmation message.   
\*\*Input\*\*: Name, email, password, phone number.   
\*\*Output\*\*: A new customer account is created, and the customer receives a confirmation message.   
  
## 1.2 Customer Login Function   
\*\*Function ID\*\*: FR-02   
\*\*Description\*\*: A customer can log in to the system using their registered email and password. The system must validate the credentials and initiate a session.   
\*\*Input\*\*: Registered email and password.   
\*\*Output\*\*: A successful login and session initiation, with the customer redirected to the homepage or dashboard.   
  
## 1.3 View Product Details Function   
\*\*Function ID\*\*: FR-03   
\*\*Description\*\*: A customer can view detailed information about a product, such as price, description, and availability. The system must retrieve the data from the Product data entity and update the view count if applicable.   
\*\*Input\*\*: Product ID or search query.   
\*\*Output\*\*: Displayed product details, including price, description, availability, and view count.   
  
## 1.4 Add Product to Cart Function   
\*\*Function ID\*\*: FR-04   
\*\*Description\*\*: A customer can add a product to their shopping cart. The system must verify product availability and update the cart accordingly.   
\*\*Input\*\*: Product ID, quantity (optional), and custom options (e.g., size, color).   
\*\*Output\*\*: Updated shopping cart with the added product and a confirmation message.   
  
## 1.5 Update Cart Contents Function   
\*\*Function ID\*\*: FR-05   
\*\*Description\*\*: A customer can modify the contents of their shopping cart, including changing the quantity or removing items. The system must update the cart and recalculate the total cost.   
\*\*Input\*\*: Cart item ID, action (increase quantity, decrease quantity, remove item), and new quantity (if applicable).   
\*\*Output\*\*: Updated cart summary and confirmation message of the cart change.   
  
## 1.6 Remove Product from Cart Function   
\*\*Function ID\*\*: FR-06   
\*\*Description\*\*: A customer can remove a product from their shopping cart. The system must verify the product exists in the cart, remove it, and update the cart summary.   
\*\*Input\*\*: Cart item ID.   
\*\*Output\*\*: Updated cart summary with the removed product and confirmation message.   
  
## 1.7 Checkout Order Function   
\*\*Function ID\*\*: FR-07   
\*\*Description\*\*: A customer can proceed with the checkout process by confirming shipping and billing details and selecting a payment method. The system must create an order, process the payment, update the inventory, and clear the cart.   
\*\*Input\*\*: Cart items, shipping and billing information, selected payment method.   
\*\*Output\*\*: A new order is created, inventory is updated, cart is cleared, and a confirmation message is sent to the customer.   
  
## 1.8 View Purchase History Function   
\*\*Function ID\*\*: FR-08   
\*\*Description\*\*: A customer can view their purchase history, including order dates, total amounts, and order status. The system must retrieve the data from the Order data entity and display it in a user-friendly format.   
\*\*Input\*\*: Customer account (logged-in session).   
\*\*Output\*\*: A list of orders with detailed information for each.   
  
## 1.9 Receive Order Confirmation Email Function   
\*\*Function ID\*\*: FR-09   
\*\*Description\*\*: After a successful checkout, the system must generate and send an order confirmation email to the customer via a configured Plugin. The email must include the order number, items, total amount, and shipping address.   
\*\*Input\*\*: Order details from the Order data entity.   
\*\*Output\*\*: A confirmation email sent to the customer and a log of the email delivery status.   
  
## 1.10 Administrator Login Function   
\*\*Function ID\*\*: FR-10   
\*\*Description\*\*: An administrator can log in to the system using their registered username and password. The system must validate the credentials and initiate a session.   
\*\*Input\*\*: Registered username and password.   
\*\*Output\*\*: A successful login and session initiation, with the administrator redirected to the admin dashboard.   
  
## 1.11 Manage Inventory Function   
\*\*Function ID\*\*: FR-11   
\*\*Description\*\*: An administrator can update the stock levels of products. The system must validate the input data and update the Inventory and Product data entities accordingly.   
\*\*Input\*\*: Product ID, new stock quantity.   
\*\*Output\*\*: Updated inventory and product availability status, along with a confirmation message.   
  
## 1.12 Update Product Information Function   
\*\*Function ID\*\*: FR-12   
\*\*Description\*\*: An administrator can modify product details such as name, price, description, and stock quantity. The system must validate and update the Product and Inventory data entities.   
\*\*Input\*\*: Product ID, updated attributes (name, price, description, stock quantity).   
\*\*Output\*\*: Updated product information in the catalog and cart interfaces, along with a confirmation message.   
  
## 1.13 Categorize Products Function   
\*\*Function ID\*\*: FR-13   
\*\*Description\*\*: An administrator can assign one or more categories to a product. The system must validate the categories and update the Product data entity.   
\*\*Input\*\*: Product ID, list of categories.   
\*\*Output\*\*: Updated product categories, and the product becomes searchable and browsable under those categories.   
  
## 1.14 Delete Product Function   
\*\*Function ID\*\*: FR-14   
\*\*Description\*\*: An administrator can remove a product from the system. The system must verify the product exists and ensure it is not referenced in any active order or cart before deletion.   
\*\*Input\*\*: Product ID.   
\*\*Output\*\*: The product is removed from the Product and Inventory data entities, and a confirmation message is displayed.   
  
## 1.15 Manage Plugin Function   
\*\*Function ID\*\*: FR-15   
\*\*Description\*\*: An administrator can modify the configuration or code of an existing plugin. The system must validate the updates and store the changes in the Plugin data entity.   
\*\*Input\*\*: Plugin ID, updated plugin name, description, configuration, or code.   
\*\*Output\*\*: Updated plugin information and confirmation message.   
  
## 1.16 View Plugin Documentation Function   
\*\*Function ID\*\*: FR-16   
\*\*Description\*\*: An administrator can view the documentation for a plugin, including configuration, usage, and integration instructions. The system must retrieve the documentation from the Plugin data entity.   
\*\*Input\*\*: Plugin ID.   
\*\*Output\*\*: Displayed plugin documentation and a log of the access.   
  
## 1.17 Manage Customer Function   
\*\*Function ID\*\*: FR-17   
\*\*Description\*\*: An administrator can view, edit, or delete customer accounts. The system must validate the input data and update the Customer data entity accordingly.   
\*\*Input\*\*: Customer ID, action (view, edit, delete), and updated customer information (if applicable).   
\*\*Output\*\*: Updated or deleted customer information, along with a confirmation message and system logs.   
  
## 1.18 Manage Product Function   
\*\*Function ID\*\*: FR-18   
\*\*Description\*\*: An administrator can add, update, or delete products in the system. The system must synchronize changes with the Product and Inventory data entities.   
\*\*Input\*\*: Product ID, action (add, update, delete), and product details (if applicable).   
\*\*Output\*\*: Updated or deleted product information in the catalog and inventory, with confirmation and logs.   
  
## 1.19 Manage Order Function   
\*\*Function ID\*\*: FR-19   
\*\*Description\*\*: An administrator can update the status, cancel, or initiate a refund for an order. The system must update the Order, Inventory, and Payment data entities accordingly and notify the customer.   
\*\*Input\*\*: Order ID, action (update status, cancel, refund).   
\*\*Output\*\*: Updated order status, inventory levels, and payment records, with confirmation and logs.   
  
## 1.20 Manage Payment Function   
\*\*Function ID\*\*: FR-20   
\*\*Description\*\*: An administrator can update the status, cancel, or initiate a refund for a payment. The system must validate the payment state and update the Payment and Order data entities.   
\*\*Input\*\*: Payment ID, action (update status, cancel, refund).   
\*\*Output\*\*: Updated payment status and related order information, with confirmation and logs.   
  
## 1.21 Manage Inventory Function (Extended)   
\*\*Function ID\*\*: FR-21   
\*\*Description\*\*: An administrator can perform stock adjustments (add, deduct, or set stock levels) for products. The system must validate the quantity and update both the Inventory and Product data entities.   
\*\*Input\*\*: Product ID, action (add, deduct, set stock), and quantity.   
\*\*Output\*\*: Updated inventory levels and product availability status, along with a confirmation message and logs.   
  
## 1.22 Manage Administrator Function   
\*\*Function ID\*\*: FR-22   
\*\*Description\*\*: An administrator can add, update, or delete other administrator accounts. The system must validate the input data and update the Administrator data entity accordingly.   
\*\*Input\*\*: Administrator ID, action (add, update, delete), and administrator details (if applicable).   
\*\*Output\*\*: Updated or deleted administrator account, with confirmation message and logs.

# External Description

# 2. External Interfaces   
  
## 2.1 User Interface Output   
  
The system interacts with customers and administrators through a user interface. These interactions include but are not limited to:   
  
- \*\*Customer Registration Form\*\*: A form allowing customers to input personal information such as name, email, password, and phone number.   
- \*\*Customer Login Page\*\*: A login interface where customers enter their registered email and password to authenticate and initiate a session.   
- \*\*Product Detail Page\*\*: A page displaying detailed product information including price, description, availability, and view count.   
- \*\*Shopping Cart Interface\*\*: An interface allowing customers to add, modify, or remove products from their cart. It provides a summary of the cart contents, including total cost and quantity.   
- \*\*Checkout Form\*\*: A form where customers confirm shipping and billing details and select a payment method.   
- \*\*Purchase History Page\*\*: A page displaying a list of the customer's previous orders, including order date, total amount, and status.   
- \*\*Order Confirmation Screen\*\*: A screen shown after a successful checkout, displaying the order confirmation and key details.   
- \*\*Administrator Login Page\*\*: A login interface for administrators to access the admin dashboard.   
- \*\*Admin Dashboard Interface\*\*: A comprehensive interface for administrators to manage inventory, products, plugins, customer accounts, and orders.   
- \*\*Product Management Interface\*\*: A tool for administrators to add, update, or delete products.   
- \*\*Inventory Management Interface\*\*: A tool for administrators to adjust stock levels and manage product availability.   
- \*\*Plugin Management Interface\*\*: An interface for administrators to modify plugin configurations or code.   
- \*\*Plugin Documentation Viewer\*\*: A user interface for administrators to view plugin documentation.   
- \*\*Customer Management Interface\*\*: A tool for administrators to view, edit, or delete customer accounts.   
- \*\*Order Management Interface\*\*: A tool for administrators to update order status, cancel orders, or process refunds.   
- \*\*Payment Management Interface\*\*: A tool for administrators to manage payment status, cancel payments, or process refunds.   
  
These interfaces must ensure user-friendly navigation and provide clear feedback to users regarding actions taken or errors encountered.   
  
## 2.2 Hardware Interface Output   
  
No hardware interfaces are required for this system. The system does not interact with external hardware devices and is designed to operate solely through software-based interactions.   
  
## 2.3 Software Interface Output   
  
The system interacts with several external software components and data entities to fulfill its functional requirements. These interfaces are described below:   
  
### 2.3.1 Product Data Entity   
- \*\*Description\*\*: This data entity stores product-related information such as product ID, name, price, description, stock quantity, and categories.   
- \*\*Interaction Method\*\*: The system retrieves and updates product data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Product ID, updated attributes (name, price, description, stock quantity).   
 - Output: Updated product information for display in the catalog and cart.   
  
### 2.3.2 Inventory Data Entity   
- \*\*Description\*\*: This data entity tracks the stock levels and availability status of products.   
- \*\*Interaction Method\*\*: The system updates and retrieves inventory data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Product ID, new stock quantity, or action (add, deduct, set stock).   
 - Output: Updated inventory levels and product availability status.   
  
### 2.3.3 Order Data Entity   
- \*\*Description\*\*: This data entity stores order information including order ID, customer ID, product IDs, quantities, total amount, shipping address, and order status.   
- \*\*Interaction Method\*\*: The system retrieves and updates order data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Order details (cart items, shipping and billing information, payment method).   
 - Output: Updated order status and related information (e.g., inventory levels, payment records).   
  
### 2.3.4 Payment Data Entity   
- \*\*Description\*\*: This data entity contains payment information such as payment ID, order ID, amount, payment method, status, and transaction logs.   
- \*\*Interaction Method\*\*: The system updates and retrieves payment data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Payment ID, action (update status, cancel, refund).   
 - Output: Updated payment status and related order information.   
  
### 2.3.5 Customer Data Entity   
- \*\*Description\*\*: This data entity stores customer account information such as customer ID, name, email, password, phone number, and account status.   
- \*\*Interaction Method\*\*: The system retrieves and updates customer data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Customer ID, action (view, edit, delete), and updated customer information.   
 - Output: Updated or deleted customer account information.   
  
### 2.3.6 Administrator Data Entity   
- \*\*Description\*\*: This data entity stores administrator account information including administrator ID, name, username, password, and role.   
- \*\*Interaction Method\*\*: The system retrieves and updates administrator data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Administrator ID, action (add, update, delete), and administrator details.   
 - Output: Updated or deleted administrator account information.   
  
### 2.3.7 Plugin Data Entity   
- \*\*Description\*\*: This data entity stores plugin-related information such as plugin ID, name, description, configuration, code, and documentation.   
- \*\*Interaction Method\*\*: The system retrieves and updates plugin data via database queries or stored procedures.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Plugin ID, updated plugin name, description, configuration, or code.   
 - Output: Updated plugin information and confirmation message.   
  
### 2.3.8 Email Plugin (Third-Party Service)   
- \*\*Description\*\*: A configured third-party email service plugin used to send confirmation emails to customers after successful checkout.   
- \*\*Interaction Method\*\*: The system triggers the plugin via API calls or event-driven mechanisms.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Order details (order number, items, total amount, shipping address).   
 - Output: A confirmation email sent to the customer and a log of the email delivery status.   
  
### 2.3.9 Session Management Module   
- \*\*Description\*\*: A software module responsible for managing user sessions (both customer and administrator).   
- \*\*Interaction Method\*\*: The system interacts with this module to initiate, maintain, or terminate sessions upon login or logout.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Credentials (email and password for customers; username and password for administrators).   
 - Output: Session initiation or termination, along with user authentication status.   
  
## 2.4 Communication Interface Output   
  
The system uses network-based communication to interact with external services and users. The following communication interfaces are required:   
  
### 2.4.1 Email Notification Service   
- \*\*Description\*\*: The system communicates with an external email service plugin to send order confirmation emails to customers.   
- \*\*Interaction Method\*\*: The system sends HTTP requests or uses an API to deliver email content, including order details.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Order details from the Order data entity.   
 - Output: A confirmation email sent to the customer with the order number, items, total amount, and shipping address.   
  
### 2.4.2 Web Browsing (HTTP/HTTPS)   
- \*\*Description\*\*: The system communicates with users through web browsers via HTTP or HTTPS protocols.   
- \*\*Interaction Method\*\*: Standard web protocols are used to load user interfaces and process user input (e.g., form submissions, AJAX requests).   
- \*\*Inputs/Outputs\*\*:   
 - Input: User inputs from web interfaces (e.g., login form, product search, cart updates).   
 - Output: Dynamic content updates and responses to user actions.   
  
### 2.4.3 Session Communication (Web Cookies/Token)   
- \*\*Description\*\*: The system maintains user sessions using cookies or token-based authentication (e.g., JWT).   
- \*\*Interaction Method\*\*: Cookies or tokens are stored in the user’s browser and sent with each request to authenticate the session.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Session cookies or tokens.   
 - Output: Session validation and access control to secure areas of the system.   
  
### 2.4.4 Plugin Documentation Access (Internal Web Service)   
- \*\*Description\*\*: The system provides access to plugin documentation through an internal web service or API.   
- \*\*Interaction Method\*\*: The system retrieves plugin documentation using HTTP requests and displays it within the user interface.   
- \*\*Inputs/Outputs\*\*:   
 - Input: Plugin ID.   
 - Output: Plugin documentation content and access logs.   
  
## Summary of External Interfaces   
  
| Interface Type | Interface Name | Description | Key Interaction Method |  
|----------------|----------------|-------------|------------------------|  
| User Interface | Customer Registration Form | Allows customers to create new accounts | Form submission with validation |  
| User Interface | Customer Login Page | Authenticates customer credentials and initiates a session | Form submission with session creation |  
| User Interface | Product Detail Page | Displays product information and updates view count | Dynamic data loading based on product ID |  
| User Interface | Shopping Cart Interface | Allows customers to manage their cart | Cart updates and confirmation messages |  
| User Interface | Checkout Form | Processes order details and payment method | Form submission with order creation |  
| User Interface | Purchase History Page | Displays customer order history | Data retrieval from Order data entity |  
| User Interface | Administrator Login Page | Authenticates administrator credentials | Form submission with session creation |  
| User Interface | Admin Dashboard Interface | Central control for administrators | Dynamic data loading and updates |  
| Software Interface | Product Data Entity | Stores and retrieves product information | Database queries |  
| Software Interface | Inventory Data Entity | Tracks product stock levels | Database queries |  
| Software Interface | Order Data Entity | Stores and retrieves order information | Database queries |  
| Software Interface | Payment Data Entity | Tracks payment status and details | Database queries |  
| Software Interface | Customer Data Entity | Stores and retrieves customer account details | Database queries |  
| Software Interface | Administrator Data Entity | Stores and retrieves administrator account details | Database queries |  
| Software Interface | Plugin Data Entity | Stores and retrieves plugin information | Database queries |  
| Communication Interface | Email Notification Service | Sends order confirmation emails | HTTP API or event-based trigger |  
| Communication Interface | Web Browsing (HTTP/HTTPS) | Loads and processes web-based user interfaces | Standard web protocols |  
| Communication Interface | Session Communication | Maintains user authentication state | Cookie or token-based authentication |  
| Communication Interface | Plugin Documentation Access | Retrieves plugin documentation | Internal web service call |   
  
This section ensures that all external interfaces required for the system are clearly defined, making it easier for developers to implement and integrate the system components.

# Use Case

Use Case Name: Customer Registration   
Use Case ID: UC-01   
Actors: Customer, Administrator   
Preconditions:   
- The customer has access to the system interface.   
- The system is operational and available for user interaction.   
- There is no existing customer account with the same email or phone number.   
  
Postconditions:   
- A new customer account is successfully created in the system.   
- The customer receives a confirmation message.   
- The customer is added to the Customer data entity.   
  
Main Flow:   
1. The customer navigates to the registration page.   
2. The customer fills in the registration form with personal information (e.g., name, email, password, phone number).   
3. The system validates the input data (e.g., checks for valid email format, password strength, and unique email/phone number).   
4. The system creates a new customer record and saves it to the Customer data entity.   
5. The system sends a confirmation message to the customer's email or phone.   
6. The customer receives the confirmation and is redirected to the login page.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields or incorrect format), the system displays an error message and prompts the customer to correct the information.   
2. If the email or phone number is already registered, the system displays an error message indicating the duplication and prevents account creation.   
3. If the system fails to send the confirmation message, the system logs the error and displays a message to the customer, allowing them to retry or contact support.  
  
Use Case Name: Customer Login   
Use Case ID: UC-02   
Actors: Customer   
Preconditions:   
- The customer has a valid account registered in the system.   
- The system is operational and available for user interaction.   
- The customer has access to the login interface.   
  
Postconditions:   
- The customer is successfully authenticated and logged into the system.   
- The customer's session is initiated.   
- The customer is redirected to the dashboard or homepage.   
  
Main Flow:   
1. The customer navigates to the login page.   
2. The customer enters their registered email and password.   
3. The system validates the email and password against the Customer data entity.   
4. If the credentials are valid, the system logs the customer in and starts a session.   
5. The customer is redirected to the homepage or dashboard.   
  
Alternative Flow:   
1. If the email is invalid or not registered, the system displays an error message and prompts the customer to try again.   
2. If the password is incorrect, the system displays an error message and allows the customer to re-enter the password.   
3. If the system fails to authenticate the customer (e.g., due to server error), it logs the error and displays a message to the customer, allowing them to retry or contact support.  
  
Use Case Name: View Product Details   
Use Case ID: UC-03   
Actors: Customer   
Preconditions:   
- The customer is logged in or has access to the public interface.   
- The system is operational and available for user interaction.   
- The requested product exists in the Product data entity.   
  
Postconditions:   
- The customer views the detailed information of the selected product.   
- The system retrieves and displays the product's attributes (e.g., price, description, availability).   
- The system updates the product view count if applicable.   
  
Main Flow:   
1. The customer browses the product catalog or searches for a specific product.   
2. The customer selects a product to view its details.   
3. The system retrieves the product information from the Product data entity.   
4. The system displays the product details to the customer.   
  
Alternative Flow:   
1. If the requested product does not exist, the system displays an error message and suggests similar products or returns to the catalog.   
2. If the system fails to retrieve product information (e.g., due to database error), it logs the error and displays a message to the customer, allowing them to retry or contact support.  
  
Use Case Name: Add Product to Cart   
Use Case ID: UC-04   
Actors: Customer   
Preconditions:   
- The customer is logged in or has access to the public interface.   
- The system is operational and available for user interaction.   
- The requested product exists in the Product data entity.   
- The product is in stock and available for purchase.   
  
Postconditions:   
- The selected product is added to the customer's Shopping Cart.   
- The Shopping Cart data entity is updated with the new product information.   
- The customer receives a confirmation message that the product was added.   
  
Main Flow:   
1. The customer browses the product catalog or searches for a product.   
2. The customer selects a product and views its details.   
3. The customer clicks the "Add to Cart" button.   
4. The system checks the product's availability in the Inventory data entity.   
5. The system adds the product to the Shopping Cart data entity.   
6. The system updates the cart summary displayed to the customer.   
7. The system provides a confirmation message that the product was successfully added.   
  
Alternative Flow:   
1. If the product is out of stock, the system displays a message indicating unavailability and does not add the product to the cart.   
2. If the product does not exist in the Product data entity, the system logs the error and displays a message to the customer.   
3. If the system fails to update the Shopping Cart (e.g., due to database error), it logs the error and displays a message to the customer, allowing them to retry or contact support.   
4. If the customer tries to add a product with a custom option (e.g., size, color), the system prompts the customer to select the necessary options before adding to the cart.  
  
Use Case Name: Update Cart Contents   
Use Case ID: UC-05   
Actors: Customer   
Preconditions:   
- The customer is logged in or has access to the public interface.   
- The system is operational and available for user interaction.   
- The customer has at least one item in their Shopping Cart.   
- The requested product(s) exist in the Product data entity.   
  
Postconditions:   
- The Shopping Cart is updated with the new quantity, removed items, or added items.   
- The Shopping Cart data entity is modified accordingly.   
- The customer receives a confirmation message of the cart update.   
  
Main Flow:   
1. The customer navigates to their Shopping Cart.   
2. The system displays the current items in the cart with options to edit quantities or remove items.   
3. The customer selects an action (e.g., increase quantity, decrease quantity, remove item).   
4. The system updates the Shopping Cart data entity based on the selected action.   
5. The system recalculates the total price and updates the cart summary.   
6. The system provides a confirmation message that the cart has been successfully updated.   
  
Alternative Flow:   
1. If the product is out of stock after updating the quantity, the system displays a message indicating the unavailability and suggests adjusting the quantity or removing the item.   
2. If the system fails to update the Shopping Cart (e.g., due to database error), it logs the error and displays a message to the customer, allowing them to retry or contact support.   
3. If the customer attempts to add a new product during the update, the system follows the Add Product to Cart flow.  
  
Use Case Name: Remove Product from Cart   
Use Case ID: UC-06   
Actors: Customer   
Preconditions:   
- The customer is logged in or has access to the public interface.   
- The system is operational and available for user interaction.   
- The customer has at least one item in their Shopping Cart.   
- The requested product exists in the Shopping Cart data entity.   
  
Postconditions:   
- The selected product is removed from the customer's Shopping Cart.   
- The Shopping Cart data entity is updated accordingly.   
- The system recalculates the cart total and updates the cart summary.   
- The customer receives a confirmation message that the product was successfully removed.   
  
Main Flow:   
1. The customer navigates to their Shopping Cart.   
2. The system displays the items currently in the cart.   
3. The customer selects a product and clicks the "Remove" button.   
4. The system verifies the product is in the cart.   
5. The system removes the product from the Shopping Cart data entity.   
6. The system updates the cart summary and total price.   
7. The system provides a confirmation message that the product has been removed.   
  
Alternative Flow:   
1. If the requested product is not in the cart, the system displays an error message and does not perform any action.   
2. If the system fails to update the Shopping Cart (e.g., due to database error), it logs the error and displays a message to the customer, allowing them to retry or contact support.  
  
Use Case Name: Checkout Order   
Use Case ID: UC-07   
Actors: Customer, Shopping Cart, Order, Payment, Inventory, Administrator, Plugin   
  
Preconditions:   
- The customer is logged in and has at least one item in their Shopping Cart.   
- The system is operational and available for user interaction.   
- The selected products are in stock and available for purchase.   
- The customer has a valid payment method configured or is ready to enter one.   
  
Postconditions:   
- The order is created and stored in the Order data entity.   
- The Payment data entity is updated with the transaction details.   
- The Inventory data entity is updated to reflect the deduction of stock.   
- The Shopping Cart is cleared of the purchased items.   
- The customer receives a confirmation message for the completed checkout.   
- The Administrator is notified of the new order if required.   
  
Main Flow:   
1. The customer navigates to the Shopping Cart and clicks the "Checkout" button.   
2. The system displays the cart summary and prompts the customer to enter or confirm shipping and billing details.   
3. The customer selects a payment method and confirms the order.   
4. The system processes the payment via the Payment data entity and Plugin if applicable (e.g., third-party payment gateway).   
5. The system creates a new order in the Order data entity, including product details, customer information, and payment status.   
6. The system updates the Inventory data entity by reducing the stock of the purchased items.   
7. The system clears the Shopping Cart and sends a confirmation message to the customer.   
8. The system logs the order details and may notify the Administrator.   
  
Alternative Flow:   
1. If the customer does not provide valid shipping or billing details, the system displays an error and prompts for corrections.   
2. If the selected payment method fails or is invalid, the system displays an error and allows the customer to choose a different method.   
3. If the Inventory update fails (e.g., due to insufficient stock or database error), the system logs the error and prevents order creation, displaying a message to the customer.   
4. If the system fails to process the payment, it logs the error and displays a message to the customer, allowing them to retry or contact support.   
5. If the customer cancels the checkout, the system returns to the Shopping Cart without making any changes.  
  
Use Case Name: View Purchase History   
Use Case ID: UC-08   
Actors: Customer, Order, Administrator   
  
Preconditions:   
- The customer is logged in.   
- The system is operational and available for user interaction.   
- The customer has previously placed at least one order.   
- The Order data entity contains the customer's order records.   
  
Postconditions:   
- The customer views their purchase history.   
- The system retrieves and displays the customer's order details.   
- The Administrator may receive a log of the customer's access to their purchase history.   
  
Main Flow:   
1. The customer navigates to their account section and selects "Purchase History".   
2. The system retrieves the customer's order records from the Order data entity.   
3. The system displays a list of orders with relevant details (e.g., order date, total amount, status).   
4. The customer can select an individual order to view more detailed information (e.g., items, shipping address).   
5. The system presents the detailed order information to the customer.   
  
Alternative Flow:   
1. If the customer has no previous orders, the system displays a message indicating that no purchase history is available.   
2. If the system fails to retrieve the order data (e.g., due to database error), it logs the error and displays a message to the customer, allowing them to retry or contact support.   
3. If the customer selects an order that does not exist, the system displays an error message and returns to the purchase history list.  
  
Use Case Name: Receive Order Confirmation Email   
Use Case ID: UC-09   
Actors: Customer, Order, Administrator, Plugin   
  
Preconditions:   
- The customer has completed the checkout process successfully.   
- The system is operational and available for user interaction.   
- The Order data entity contains the newly created order.   
- The system is configured to send confirmation emails via the Plugin.   
  
Postconditions:   
- The customer receives an order confirmation email.   
- The email includes order details (e.g., order number, items, total amount, shipping address).   
- The Administrator may receive a notification of the order confirmation.   
- The system logs the email delivery status.   
  
Main Flow:   
1. After the customer completes the checkout process (UC-07), the system triggers the "Send Order Confirmation Email" action.   
2. The system retrieves the order details from the Order data entity.   
3. The system generates the confirmation email content using the retrieved data.   
4. The system uses the Plugin to send the confirmation email to the customer's registered email address.   
5. The system logs the email as sent in the system records.   
6. The customer receives the confirmation email and can review their order details.   
  
Alternative Flow:   
1. If the customer's email address is invalid or missing, the system logs the error and displays a message to the customer, allowing them to update their email information.   
2. If the Plugin fails to send the email (e.g., due to connection issues or API errors), the system logs the error and displays a message to the customer, suggesting they check their inbox or contact support.   
3. If the system is not configured to send emails, the confirmation is displayed on the screen instead of being sent.   
4. If the order does not exist in the Order data entity, the system logs the error and does not send the confirmation email.  
  
Use Case Name: Administrator Login   
Use Case ID: UC-10   
Actors: Administrator   
Preconditions:   
- The administrator has access to the system interface.   
- The system is operational and available for user interaction.   
- The administrator has a valid account registered in the Administrator data entity.   
  
Postconditions:   
- The administrator is successfully authenticated and logged into the system.   
- The administrator's session is initiated.   
- The administrator is redirected to the admin dashboard.   
  
Main Flow:   
1. The administrator navigates to the admin login page.   
2. The administrator enters their registered username and password.   
3. The system validates the username and password against the Administrator data entity.   
4. If the credentials are valid, the system logs the administrator in and starts a session.   
5. The administrator is redirected to the admin dashboard.   
  
Alternative Flow:   
1. If the username is invalid or not registered, the system displays an error message and prompts the administrator to try again.   
2. If the password is incorrect, the system displays an error message and allows the administrator to re-enter the password.   
3. If the system fails to authenticate the administrator (e.g., due to server error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-11   
Actors: Administrator, Inventory, Product   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Inventory data entity is accessible and contains current stock levels.   
- The Product data entity is accessible and contains product information.   
  
Postconditions:   
- The inventory levels are updated in the Inventory data entity.   
- The system provides a confirmation message of the inventory update.   
- The product stock status is synchronized with the updated inventory data.   
- The administrator receives a summary of the changes made.   
  
Main Flow:   
1. The administrator navigates to the Inventory Management section from the admin dashboard.   
2. The system displays a list of products with their current stock levels.   
3. The administrator selects a product and modifies its stock quantity (e.g., increase or decrease).   
4. The system updates the Inventory data entity with the new stock quantity.   
5. The system updates the Product data entity to reflect the new availability status.   
6. The system provides a confirmation message indicating the inventory update was successful.   
7. The administrator reviews the updated inventory and product status.   
  
Alternative Flow:   
1. If the administrator enters an invalid stock quantity (e.g., negative number or non-numeric value), the system displays an error message and prompts for a valid input.   
2. If the system fails to update the Inventory data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
3. If the selected product does not exist in the Product data entity, the system logs the error and displays a message to the administrator.   
4. If the administrator tries to update inventory for a product that is not in the inventory list, the system displays an error and suggests checking the product catalog.  
  
Use Case Name: Update Product Information   
Use Case ID: UC-12   
Actors: Administrator, Product, Inventory   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The requested product exists in the Product data entity.   
- The administrator has access to the product management interface.   
  
Postconditions:   
- The product information is updated in the Product data entity.   
- The Inventory data entity is updated if stock quantity is modified.   
- The system provides a confirmation message that the update was successful.   
- The product details are reflected in the product catalog and cart interfaces.   
  
Main Flow:   
1. The administrator navigates to the product management section from the admin dashboard.   
2. The system displays a list of products with options to edit.   
3. The administrator selects a product and clicks the "Edit" button.   
4. The administrator modifies the product details (e.g., name, price, description, stock quantity).   
5. The system validates the updated information (e.g., price format, numeric stock quantity).   
6. The system updates the Product data entity with the new information.   
7. If stock quantity is updated, the system also updates the Inventory data entity.   
8. The system provides a confirmation message that the product information was successfully updated.   
  
Alternative Flow:   
1. If the administrator enters invalid data (e.g., negative price or non-numeric stock quantity), the system displays an error message and prompts for valid input.   
2. If the requested product does not exist in the Product data entity, the system logs the error and displays a message to the administrator.   
3. If the system fails to update the Product or Inventory data entities (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.  
  
Use Case Name: Categorize Products   
Use Case ID: UC-13   
Actors: Administrator, Product, Inventory   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Product data entity contains the product to be categorized.   
- The administrator has access to the product categorization interface.   
  
Postconditions:   
- The product is assigned to one or more categories.   
- The Product data entity is updated with the new category information.   
- The system provides a confirmation message that the categorization was successful.   
- The product becomes searchable and browsable under the assigned categories.   
  
Main Flow:   
1. The administrator navigates to the product categorization section from the admin dashboard.   
2. The system displays a list of products along with their current categories (if any).   
3. The administrator selects a product and clicks the "Categorize" button.   
4. The administrator assigns the product to one or more categories from the available options.   
5. The system validates the selected categories and ensures they exist in the system.   
6. The system updates the Product data entity with the new category information.   
7. The system provides a confirmation message that the product has been successfully categorized.   
8. The administrator reviews the updated product information and categories.   
  
Alternative Flow:   
1. If the selected category does not exist, the system displays an error message and allows the administrator to choose a valid category.   
2. If the product is already in the selected category, the system displays a message and prevents duplicate categorization.   
3. If the system fails to update the Product data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator selects multiple categories with conflicting attributes, the system prompts for clarification or adjustment.  
  
Use Case Name: Delete Product   
Use Case ID: UC-14   
Actors: Administrator, Product, Inventory   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The requested product exists in the Product data entity.   
- The administrator has access to the product deletion interface.   
  
Postconditions:   
- The selected product is removed from the Product data entity.   
- The Inventory data entity is updated to remove stock information for the deleted product.   
- The system provides a confirmation message that the product was successfully deleted.   
- The product is no longer visible in the catalog or cart interfaces.   
  
Main Flow:   
1. The administrator navigates to the product management section from the admin dashboard.   
2. The system displays a list of products with options to delete.   
3. The administrator selects a product and clicks the "Delete" button.   
4. The system verifies the product exists in the Product data entity.   
5. The system removes the product from the Product data entity.   
6. The system updates the Inventory data entity to remove the product's stock information.   
7. The system provides a confirmation message that the product was successfully deleted.   
  
Alternative Flow:   
1. If the requested product does not exist in the Product data entity, the system logs the error and displays a message to the administrator.   
2. If the system fails to delete the product (e.g., due to database constraints or error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
3. If the product is referenced in an existing order or cart, the system displays a warning message and prevents deletion until references are resolved.  
  
Use Case Name: Create Plugin   
Use Case ID: UC-15   
Actors: Administrator, Plugin   
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Plugin interface is accessible and configured.   
- The administrator has the necessary permissions to create plugins.   
  
Postconditions:   
- A new plugin is successfully created and registered in the system.   
- The Plugin data entity is updated with the new plugin information.   
- The system provides a confirmation message of the plugin creation.   
- The plugin is available for use or integration within the system.   
  
Main Flow:   
1. The administrator navigates to the Plugin Management section from the admin dashboard.   
2. The system displays an interface for creating a new plugin.   
3. The administrator fills in the plugin details (e.g., name, description, configuration parameters).   
4. The administrator uploads or provides the plugin code or integration details.   
5. The system validates the plugin configuration and code (e.g., correct format, required fields).   
6. The system registers the plugin and saves the information to the Plugin data entity.   
7. The system provides a confirmation message that the plugin was successfully created.   
8. The administrator reviews the new plugin in the plugin list.   
  
Alternative Flow:   
1. If the plugin configuration is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the plugin code is malformed or fails validation, the system logs the error and displays a message to the administrator, allowing them to re-upload or modify the code.   
3. If the system fails to register the plugin (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator tries to create a plugin with a duplicate name, the system displays an error message and prevents the creation.  
  
Use Case Name: Update Plugin   
Use Case ID: UC-16   
Actors: Administrator, Plugin   
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The requested plugin exists in the Plugin data entity.   
- The administrator has access to the plugin management interface.   
  
Postconditions:   
- The plugin is updated with new or modified configuration or code.   
- The Plugin data entity is updated with the latest plugin information.   
- The system provides a confirmation message of the plugin update.   
- The updated plugin is available for use or integration within the system.   
  
Main Flow:   
1. The administrator navigates to the Plugin Management section from the admin dashboard.   
2. The system displays a list of plugins with options to update.   
3. The administrator selects a plugin and clicks the "Update" button.   
4. The system loads the current plugin information for editing.   
5. The administrator modifies the plugin details (e.g., name, description, configuration parameters) or uploads updated code.   
6. The system validates the updated configuration and code (e.g., correct format, required fields).   
7. The system updates the Plugin data entity with the new information.   
8. The system provides a confirmation message that the plugin was successfully updated.   
9. The administrator reviews the updated plugin in the plugin list.   
  
Alternative Flow:   
1. If the updated plugin configuration is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the uploaded plugin code is malformed or fails validation, the system logs the error and displays a message to the administrator, allowing them to re-upload or modify the code.   
3. If the system fails to update the Plugin data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator tries to update a plugin with a duplicate name, the system displays an error message and prevents the update.  
  
Use Case Name: Delete Plugin   
Use Case ID: UC-17   
Actors: Administrator, Plugin   
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The requested plugin exists in the Plugin data entity.   
- The administrator has access to the plugin management interface.   
  
Postconditions:   
- The selected plugin is removed from the Plugin data entity.   
- The system provides a confirmation message that the plugin was successfully deleted.   
- The plugin is no longer active or available for use in the system.   
- The system logs the deletion action for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Plugin Management section from the admin dashboard.   
2. The system displays a list of plugins with options to delete.   
3. The administrator selects a plugin and clicks the "Delete" button.   
4. The system verifies the plugin exists in the Plugin data entity.   
5. The system removes the plugin from the Plugin data entity.   
6. The system deactivates or unregisters the plugin from the system.   
7. The system provides a confirmation message that the plugin was successfully deleted.   
8. The administrator reviews the updated plugin list to confirm the deletion.   
  
Alternative Flow:   
1. If the requested plugin does not exist in the Plugin data entity, the system logs the error and displays a message to the administrator.   
2. If the plugin is currently in use or referenced by another system component, the system displays a warning message and prevents deletion until dependencies are resolved.   
3. If the system fails to delete the plugin (e.g., due to database constraints or error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator cancels the deletion process, the system returns to the plugin management interface without making any changes.  
  
Use Case Name: View Plugin Documentation   
Use Case ID: UC-18   
Actors: Administrator, Plugin   
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The requested plugin exists in the Plugin data entity.   
- The plugin documentation is available and accessible in the system.   
  
Postconditions:   
- The administrator views the documentation for the selected plugin.   
- The system retrieves and displays the plugin's documentation.   
- The administrator receives a summary of the plugin's functionality and usage instructions.   
- The system logs the access to the plugin documentation.   
  
Main Flow:   
1. The administrator navigates to the Plugin Management section from the admin dashboard.   
2. The system displays a list of plugins with options to view documentation.   
3. The administrator selects a plugin and clicks the "View Documentation" button.   
4. The system retrieves the plugin's documentation from the Plugin data entity.   
5. The system displays the documentation to the administrator, including details such as configuration, usage, and integration instructions.   
6. The administrator reviews the documentation to understand the plugin's purpose and functionality.   
  
Alternative Flow:   
1. If the requested plugin does not exist in the Plugin data entity, the system logs the error and displays a message to the administrator.   
2. If the plugin documentation is missing or not available, the system displays an error message and suggests checking the plugin configuration or contacting support.   
3. If the system fails to retrieve the documentation (e.g., due to file access issues), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator selects a plugin that is not yet activated, the system displays a message indicating that the plugin is inactive and may need to be activated before viewing the documentation.  
  
Use Case Name: Manage Customer   
Use Case ID: UC-19   
Actors: Administrator, Customer   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Customer data entity is accessible and contains customer records.   
- The administrator has access to the customer management interface.   
  
Postconditions:   
- The customer record is updated or modified in the Customer data entity.   
- The system provides a confirmation message of the customer management action (e.g., update, suspend, or delete).   
- The administrator receives a summary of the changes made to the customer record.   
- The system logs the administrator's action for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Customer Management section from the admin dashboard.   
2. The system displays a list of registered customers with options to view, edit, or delete.   
3. The administrator selects a specific customer and clicks the "Edit" button.   
4. The system loads the customer's current information (e.g., name, email, phone number, account status).   
5. The administrator modifies the customer's information or updates their account status (e.g., suspend or activate account).   
6. The system validates the updated information (e.g., correct email format, valid phone number).   
7. The system updates the Customer data entity with the new or modified information.   
8. The system provides a confirmation message that the customer information was successfully updated.   
9. The administrator reviews the updated customer record in the customer list.   
  
Alternative Flow:   
1. If the administrator enters invalid customer data (e.g., duplicate email or phone number), the system displays an error message and prompts for valid input.   
2. If the requested customer does not exist in the Customer data entity, the system logs the error and displays a message to the administrator.   
3. If the system fails to update the Customer data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the administrator attempts to delete a customer account, the system follows the Delete Customer use case flow.   
5. If the administrator tries to perform an action on a suspended or inactive account, the system displays a warning message and allows the administrator to activate or unsuspend the account before proceeding.  
  
Use Case Name: Manage Product   
Use Case ID: UC-20   
Actors: Administrator, Product, Inventory   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The administrator has access to the product management interface.   
- The Product data entity is accessible and contains product records.   
  
Postconditions:   
- The product is either added, updated, or removed from the Product data entity.   
- The Inventory data entity is synchronized with the product status (e.g., stock levels updated).   
- The system provides confirmation messages for the specific action performed.   
- The product catalog is updated to reflect the changes.   
- The system logs the administrator's actions for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Product Management section from the admin dashboard.   
2. The system displays a list of existing products with options to add, edit, or delete.   
3. The administrator selects an action (Add, Update, or Delete) for a product.   
4. If the action is "Add", the administrator fills in the product details (e.g., name, description, price, stock quantity, category).   
5. If the action is "Update", the administrator selects a product and modifies its attributes (e.g., price, description, stock quantity).   
6. If the action is "Delete", the administrator selects a product and confirms the deletion.   
7. The system validates the input data (e.g., correct format, no duplicate product names, valid stock quantity).   
8. The system updates the Product data entity with the new or modified information.   
9. The system updates the Inventory data entity if stock quantity is changed.   
10. The system provides a confirmation message indicating the success of the action.   
11. The administrator reviews the updated product list and inventory status.   
  
Alternative Flow:   
1. If the administrator attempts to add a product with a duplicate name, the system displays an error message and prompts for a unique product name.   
2. If the product does not exist in the Product data entity when attempting to update or delete, the system logs the error and displays a message to the administrator.   
3. If the input data is invalid (e.g., negative price, non-numeric stock quantity), the system displays an error message and prompts for corrections.   
4. If the system fails to update or delete the product (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
5. If the administrator cancels the action, the system returns to the product management interface without making any changes.   
6. If the product is referenced in an active order or shopping cart, the system displays a warning and prevents deletion until dependencies are resolved.  
  
Use Case Name: Manage Order   
Use Case ID: UC-21   
Actors: Administrator, Order, Customer, Product, Inventory, Payment, Plugin   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Order data entity is accessible and contains order records.   
- The administrator has access to the order management interface.   
  
Postconditions:   
- The order record is either updated or canceled in the Order data entity.   
- The Inventory data entity is updated to reflect changes in stock levels based on the order status.   
- The Payment data entity is updated if the order status affects the transaction (e.g., refund initiated).   
- The system provides a confirmation message for the specific action performed (e.g., order status change, cancellation).   
- The customer is notified of the order status change via email or in-app notification if applicable.   
- The system logs the administrator's action for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Order Management section from the admin dashboard.   
2. The system displays a list of orders with their current status (e.g., pending, processing, shipped, canceled).   
3. The administrator selects a specific order and clicks the "Manage" button.   
4. The system retrieves the order details, including customer information, product list, payment status, and inventory allocation.   
5. The administrator reviews the order details and selects an action (e.g., update status, cancel order, initiate refund).   
6. The system validates the selected action (e.g., ensures the order is eligible for cancellation or status update).   
7. If the action is "Update Status", the system modifies the order status in the Order data entity (e.g., from "pending" to "shipped").   
8. If the action is "Cancel Order", the system marks the order as canceled and updates the Inventory data entity to return the reserved stock.   
9. If the action is "Initiate Refund", the system processes the refund via the Payment data entity and Plugin if applicable (e.g., third-party payment gateway).   
10. The system provides a confirmation message that the action was successfully performed.   
11. The system updates the order list and notifies the customer of the change.   
12. The administrator reviews the updated order status and logs the action for auditing.   
  
Alternative Flow:   
1. If the selected order does not exist in the Order data entity, the system logs the error and displays a message to the administrator.   
2. If the order is in a status that does not allow the selected action (e.g., attempting to cancel a shipped order), the system displays a warning message and prevents the action.   
3. If the system fails to update the Order, Inventory, or Payment data entities (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the refund process fails (e.g., due to payment gateway issues), the system logs the error and displays a message to the administrator, allowing them to investigate or contact the payment provider.   
5. If the administrator cancels the action, the system returns to the order management interface without making any changes.   
6. If the order contains products that are no longer available in the system, the system displays a warning message and allows the administrator to proceed or adjust the order details.  
  
Use Case Name: Manage Payment   
Use Case ID: UC-22   
Actors: Administrator, Payment, Plugin   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Payment data entity is accessible and contains transaction records.   
- The administrator has access to the payment management interface.   
- The Plugin is configured to support payment-related actions (e.g., refund processing, payment gateway integration).   
  
Postconditions:   
- The payment record is either updated, canceled, or refunded in the Payment data entity.   
- The system provides a confirmation message for the specific action performed.   
- The Order data entity is updated to reflect the new payment status.   
- The customer is notified of the payment status change via email or in-app notification if applicable.   
- The system logs the administrator's action for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Payment Management section from the admin dashboard.   
2. The system displays a list of payment records, including transaction status, amount, customer, and associated order.   
3. The administrator selects a specific payment and clicks the "Manage" button.   
4. The system retrieves the payment details, including transaction information, payment method, and related order status.   
5. The administrator reviews the payment details and selects an action (e.g., update status, cancel payment, initiate refund).   
6. The system validates the selected action (e.g., ensures the payment is eligible for refund or status update).   
7. If the action is "Update Status", the system modifies the payment status in the Payment data entity (e.g., from "pending" to "completed" or "failed").   
8. If the action is "Cancel Payment", the system marks the payment as canceled and updates the related order status accordingly.   
9. If the action is "Initiate Refund", the system processes the refund via the Plugin (e.g., third-party payment gateway) and updates the Payment data entity with the refund status.   
10. The system provides a confirmation message that the action was successfully performed.   
11. The system updates the payment list and notifies the customer of the change.   
12. The administrator reviews the updated payment status and logs the action for auditing.   
  
Alternative Flow:   
1. If the selected payment does not exist in the Payment data entity, the system logs the error and displays a message to the administrator.   
2. If the payment is in a status that does not allow the selected action (e.g., attempting to refund a completed transaction), the system displays a warning message and prevents the action.   
3. If the system fails to update the Payment data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the refund process via the Plugin fails (e.g., due to payment gateway issues), the system logs the error and displays a message to the administrator, allowing them to investigate or contact the payment provider.   
5. If the administrator cancels the action, the system returns to the payment management interface without making any changes.   
6. If the payment is linked to an order that has already been shipped or processed, the system displays a warning message and allows the administrator to proceed or adjust the order status first.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-23   
Actors: Administrator, Inventory, Product   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The administrator has access to the inventory management interface.   
- The Inventory data entity is accessible and contains current stock levels.   
- The Product data entity is accessible and contains product information.   
  
Postconditions:   
- The inventory levels are updated in the Inventory data entity.   
- The product availability status is synchronized with the updated inventory data in the Product data entity.   
- The system provides a confirmation message of the inventory management action (e.g., stock added, stock deducted, or stock updated).   
- The administrator receives a summary of the changes made.   
- The system logs the administrator's action for auditing purposes.   
  
Main Flow:   
1. The administrator navigates to the Inventory Management section from the admin dashboard.   
2. The system displays a list of products along with their current stock levels.   
3. The administrator selects a product to manage its inventory.   
4. The administrator chooses an action (e.g., add stock, deduct stock, or update stock level).   
5. The system prompts the administrator to enter the quantity to add, deduct, or set for the selected product.   
6. The system validates the input quantity (e.g., numeric value, no negative numbers).   
7. The system updates the Inventory data entity with the new stock level.   
8. The system updates the Product data entity to reflect the new availability status based on the inventory level.   
9. The system provides a confirmation message that the inventory was successfully updated.   
10. The administrator reviews the updated inventory and product availability.   
  
Alternative Flow:   
1. If the administrator enters an invalid quantity (e.g., negative number or non-numeric value), the system displays an error message and prompts for valid input.   
2. If the requested product does not exist in the Product data entity, the system logs the error and displays a message to the administrator.   
3. If the system fails to update the Inventory data entity (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
4. If the selected product has a stock level of zero and the administrator chooses to deduct stock, the system displays an error message and prevents the action.   
5. If the administrator cancels the action, the system returns to the inventory management interface without making any changes.   
6. If the product is referenced in an active order or shopping cart, the system displays a warning and may restrict certain inventory adjustments until the order is resolved.  
  
Use Case Name: Manage Administrator   
Use Case ID: UC-24   
Actors: Administrator   
  
Preconditions:   
- The administrator is logged in (UC-10).   
- The system is operational and available for user interaction.   
- The Administrator data entity is accessible and contains administrator records.   
- The administrator has access to the administrator management interface.   
  
Postconditions:   
- The administrator record is either added, updated, or removed from the Administrator data entity.   
- The system provides a confirmation message for the specific action performed (e.g., creation, modification, or deletion of an administrator account).   
- The system logs the administrator's action for auditing purposes.   
- The changes are reflected in the administrator access and permissions within the system.   
  
Main Flow:   
1. The administrator navigates to the Administrator Management section from the admin dashboard.   
2. The system displays a list of existing administrators with options to add, edit, or delete.   
3. The administrator selects an action (Add, Update, or Delete) for an administrator account.   
4. If the action is "Add", the administrator fills in the new administrator's details (e.g., name, email, password, role/permissions).   
5. If the action is "Update", the administrator selects an existing administrator and modifies their attributes (e.g., email, role, or status).   
6. If the action is "Delete", the administrator selects an existing administrator and confirms the deletion.   
7. The system validates the input data (e.g., correct email format, valid permissions, no duplicate administrator emails).   
8. The system updates the Administrator data entity with the new or modified information.   
9. The system provides a confirmation message indicating the success of the action.   
10. The administrator reviews the updated administrator list and verifies the changes.   
  
Alternative Flow:   
1. If the administrator attempts to add a new administrator with a duplicate email, the system displays an error message and prompts for a unique email address.   
2. If the requested administrator does not exist in the Administrator data entity when attempting to update or delete, the system logs the error and displays a message to the administrator.   
3. If the input data is invalid (e.g., missing required fields, incorrect permission settings), the system displays an error message and prompts the administrator to correct the information.   
4. If the system fails to update or delete the administrator record (e.g., due to database error), it logs the error and displays a message to the administrator, allowing them to retry or contact support.   
5. If the administrator cancels the action, the system returns to the administrator management interface without making any changes.   
6. If the administrator is currently logged in and attempts to delete their own account, the system displays a warning message and prevents the action.