项目文档

# Functional Requirement

# 1. Functional Requirements  
  
## 1.1 Customer Registration Function   
\*\*Function ID:\*\* FR-01   
\*\*Description:\*\* Customers can register for a new account by providing valid personal information and a valid email address. The system validates the email, checks for duplicate entries, and sends a confirmation email.   
\*\*Input:\*\* Name, email, password, and optional contact information.   
\*\*Output:\*\* A new customer account is created and stored in the Personal Information and Contact Information data entities. A confirmation email is sent to the customer's email address.  
  
## 1.2 Customer Login Function   
\*\*Function ID:\*\* FR-02   
\*\*Description:\*\* Customers can log in to the system by entering their registered email and password. The system validates the credentials and logs the login event.   
\*\*Input:\*\* Email and password.   
\*\*Output:\*\* The customer is authenticated and redirected to their personalized dashboard. The login event is recorded in the Purchase History data entity.  
  
## 1.3 Customer Logout Function   
\*\*Function ID:\*\* FR-03   
\*\*Description:\*\* Customers can log out of the system. The session is terminated and the logout event is recorded.   
\*\*Input:\*\* The customer clicks the "Logout" button.   
\*\*Output:\*\* The session is terminated, and the customer is redirected to the homepage or login page. The logout event is recorded in the Purchase History data entity.  
  
## 1.4 Personal Information Update Function   
\*\*Function ID:\*\* FR-04   
\*\*Description:\*\* Customers can update their personal information, such as name or email. The system validates the updated information and logs the event.   
\*\*Input:\*\* Updated personal information fields.   
\*\*Output:\*\* Updated personal information is stored in the Personal Information data entity. A confirmation email is sent, and the update event is recorded in the Purchase History data entity.  
  
## 1.5 Contact Information Management Function   
\*\*Function ID:\*\* FR-05   
\*\*Description:\*\* Customers can manage their contact information, such as phone number or address. The system validates the updated information and logs the event.   
\*\*Input:\*\* Updated contact information fields.   
\*\*Output:\*\* Updated contact information is stored in the Contact Information data entity. The update event is recorded in the Purchase History data entity.  
  
## 1.6 Purchase History Viewing Function   
\*\*Function ID:\*\* FR-06   
\*\*Description:\*\* Customers can view their purchase history, including order details and status. The system retrieves the history and logs the event.   
\*\*Input:\*\* The customer navigates to the "Purchase History" section.   
\*\*Output:\*\* The purchase history is displayed on the interface, including order number, date, product names, quantities, total amount, and status. The viewing event is recorded in the Purchase History data entity.  
  
## 1.7 Product Addition Function   
\*\*Function ID:\*\* FR-07   
\*\*Description:\*\* Administrators can add new products to the system, specifying details such as name, description, price, category, and inventory.   
\*\*Input:\*\* Product name, description, price, category, and inventory quantity.   
\*\*Output:\*\* A new product is stored in the Product data entity, associated with the selected category. The inventory is updated accordingly. The addition event is recorded in the Purchase History data entity.  
  
## 1.8 Product Removal Function   
\*\*Function ID:\*\* FR-08   
\*\*Description:\*\* Administrators can remove existing products from the system. The system checks for active orders or cart references before proceeding.   
\*\*Input:\*\* The product to be removed is selected and the "Remove Product" button is clicked.   
\*\*Output:\*\* The product is removed from the Product data entity, and the inventory is updated. The removal event is recorded in the Purchase History data entity.  
  
## 1.9 Product Information Update Function   
\*\*Function ID:\*\* FR-09   
\*\*Description:\*\* Administrators can update the information of existing products, such as name, description, price, category, or inventory.   
\*\*Input:\*\* Updated product information fields.   
\*\*Output:\*\* Updated product information is stored in the Product data entity. The inventory and category are updated accordingly. The update event is recorded in the Purchase History data entity.  
  
## 1.10 Product Categorization Function   
\*\*Function ID:\*\* FR-10   
\*\*Description:\*\* Administrators can assign or modify the category of a product. The system validates the category and updates the association.   
\*\*Input:\*\* Product and new category selection.   
\*\*Output:\*\* The product is assigned to the selected category in the Category data entity. The update is logged in the Purchase History data entity.  
  
## 1.11 Inventory Management Function   
\*\*Function ID:\*\* FR-11   
\*\*Description:\*\* Administrators can view and modify inventory levels for products. The system updates the inventory quantity and logs the event.   
\*\*Input:\*\* Updated inventory quantity for a product.   
\*\*Output:\*\* Updated inventory levels are stored in the Inventory data entity. The management event is recorded in the Purchase History data entity.  
  
## 1.12 Shopping Cart Creation Function   
\*\*Function ID:\*\* FR-12   
\*\*Description:\*\* Customers can add products to their shopping cart. The system checks product availability and updates inventory.   
\*\*Input:\*\* Product selection and "Add to Cart" action.   
\*\*Output:\*\* The product is added to the Shopping Cart data entity. Inventory quantity is updated. The customer is shown a confirmation message.  
  
## 1.13 Shopping Cart Modification Function   
\*\*Function ID:\*\* FR-13   
\*\*Description:\*\* Customers can modify the quantity or remove items from their shopping cart. The system updates the cart and inventory.   
\*\*Input:\*\* Cart item selection, quantity change, or "Remove" action.   
\*\*Output:\*\* The cart is updated in the Shopping Cart and CartItem data entities. Inventory levels are adjusted accordingly. The modification event is recorded in the Purchase History data entity.  
  
## 1.14 Shopping Cart Viewing Function   
\*\*Function ID:\*\* FR-14   
\*\*Description:\*\* Customers can view the contents of their shopping cart. The system displays a summary of the cart items and their total cost.   
\*\*Input:\*\* The customer navigates to the "Shopping Cart" section.   
\*\*Output:\*\* The system displays the list of cart items, including product name, quantity, price, and total amount. The viewing event is recorded in the Purchase History data entity.  
  
## 1.15 Checkout Function   
\*\*Function ID:\*\* FR-15   
\*\*Description:\*\* Customers can proceed to checkout, select a shipping address, and make a payment. The system processes the payment and creates an order.   
\*\*Input:\*\* Cart items, shipping address, and payment method.   
\*\*Output:\*\* An order is created in the Order data entity. Inventory levels are updated. A payment is processed and stored in the Payment data entity. An order confirmation email is sent. The checkout event is recorded in the Purchase History data entity.  
  
## 1.16 Order Confirmation Email Sending Function   
\*\*Function ID:\*\* FR-16   
\*\*Description:\*\* The system automatically sends an order confirmation email to the customer after successful checkout. The email includes order details and a confirmation message.   
\*\*Input:\*\* Order details from the Order data entity.   
\*\*Output:\*\* An Order Confirmation Email is stored in the EmailRecord data entity and sent to the customer's email address. The confirmation event is recorded in the Purchase History data entity.  
  
## 1.17 Payment Processing Function   
\*\*Function ID:\*\* FR-17   
\*\*Description:\*\* The system processes the customer's selected payment method and validates the payment details. A successful payment is stored and associated with the order.   
\*\*Input:\*\* Payment method and payment details.   
\*\*Output:\*\* The payment is stored in the Payment data entity. The order is created and stored in the Order data entity. Inventory levels are updated. The payment event is recorded in the Purchase History data entity.  
  
## 1.18 Plugin Development Function   
\*\*Function ID:\*\* FR-18   
\*\*Description:\*\* Administrators can develop or install new plugins to expand the system's functionality. The system validates the plugin and activates it.   
\*\*Input:\*\* Plugin details or file.   
\*\*Output:\*\* The plugin is stored in the Plugin data entity and activated. The system interface is updated to reflect the new functionality. The development or installation event is recorded in the Purchase History data entity.  
  
## 1.19 Plugin Configuration Function   
\*\*Function ID:\*\* FR-19   
\*\*Description:\*\* Administrators can configure the settings of an installed plugin. The system validates and applies the configuration.   
\*\*Input:\*\* Plugin configuration settings.   
\*\*Output:\*\* The Plugin data entity is updated with the new configuration. The system interface reflects the updated settings. The configuration event is recorded in the Purchase History data entity.  
  
## 1.20 User Interface Specification Compliance Checking Function   
\*\*Function ID:\*\* FR-20   
\*\*Description:\*\* Administrators can check the system's interface for compliance with the defined User Interface Specification. The system identifies and reports non-compliant elements.   
\*\*Input:\*\* The administrator initiates a compliance check.   
\*\*Output:\*\* A compliance report is generated and displayed to the administrator. Non-compliant elements are highlighted. The compliance check event is recorded in the Purchase History data entity.  
  
## 1.21 Manage CartItem Function   
\*\*Function ID:\*\* FR-21   
\*\*Description:\*\* Customers can modify the quantity or remove items in their shopping cart. The system updates the cart and inventory accordingly.   
\*\*Input:\*\* Cart item selection, quantity change, or "Remove" action.   
\*\*Output:\*\* The cart is updated in the CartItem data entity. Inventory levels are adjusted accordingly. The modification event is recorded in the Purchase History data entity.  
  
## 1.22 Manage OrderItem Function   
\*\*Function ID:\*\* FR-22   
\*\*Description:\*\* Administrators can edit, remove, or update the status of items in an order. The system updates the order and inventory accordingly.   
\*\*Input:\*\* Order item selection, quantity change, or "Remove" action.   
\*\*Output:\*\* The order item is updated in the OrderItem data entity. Inventory levels are adjusted accordingly. The modification event is recorded in the Purchase History data entity.  
  
## 1.23 Manage EmailRecord Function   
\*\*Function ID:\*\* FR-23   
\*\*Description:\*\* Administrators can view, edit, or delete email records. The system updates the EmailRecord data entity and logs the event.   
\*\*Input:\*\* Email record selection and action (view, edit, or delete).   
\*\*Output:\*\* The EmailRecord data entity is updated. The system displays a confirmation message. The management event is recorded in the Purchase History data entity.  
  
## 1.24 View EmailRecord Function   
\*\*Function ID:\*\* FR-24   
\*\*Description:\*\* Customers and administrators can view the details of an email record. The system displays the email content and logs the viewing event.   
\*\*Input:\*\* Email record selection.   
\*\*Output:\*\* The system displays the email details from the EmailRecord data entity. The viewing event is recorded in the Purchase History data entity.  
  
## 1.25 View CartItem Function   
\*\*Function ID:\*\* FR-25   
\*\*Description:\*\* Customers can view the details of a specific cart item. The system displays the product description, price, and quantity.   
\*\*Input:\*\* Cart item selection.   
\*\*Output:\*\* The system displays the cart item details from the CartItem data entity. The viewing event is recorded in the Purchase History data entity.  
  
## 1.26 View Payment Function   
\*\*Function ID:\*\* FR-26   
\*\*Description:\*\* Customers and administrators can view the details of a payment record. The system displays the payment method, amount, date, and status.   
\*\*Input:\*\* Payment record selection.   
\*\*Output:\*\* The system displays the payment details from the Payment data entity. The viewing event is recorded in the Purchase History data entity.  
  
## 1.27 Manage Administrator Function   
\*\*Function ID:\*\* FR-27   
\*\*Description:\*\* Administrators can manage other administrator accounts by updating or deactivating them. The system updates the administrator's information and logs the event.   
\*\*Input:\*\* Administrator account selection and action (update or deactivate).   
\*\*Output:\*\* The administrator account is updated in the Personal Information and Contact Information data entities. The management event is recorded in the Purchase History data entity.  
  
## 1.28 Manage Customer Function   
\*\*Function ID:\*\* FR-28   
\*\*Description:\*\* Administrators can manage customer accounts, including updating personal information or deactivating accounts.   
\*\*Input:\*\* Customer account selection and action (update or deactivate).   
\*\*Output:\*\* The customer's information is updated in the Personal Information and Contact Information data entities. The management event is recorded in the Purchase History data entity.  
  
## 1.29 Manage Category Function   
\*\*Function ID:\*\* FR-29   
\*\*Description:\*\* Administrators can add, update, or remove product categories. The system validates the category and updates the associations with products.   
\*\*Input:\*\* Category selection and action (add, update, or delete).   
\*\*Output:\*\* The Category data entity is updated. Product associations are modified accordingly. The management event is recorded in the Purchase History data entity.  
  
## 1.30 Manage User Interface Specification Function   
\*\*Function ID:\*\* FR-30   
\*\*Description:\*\* Administrators can modify the User Interface Specification to align with new design or functionality requirements. The system applies the changes and logs the event.   
\*\*Input:\*\* Updated UI specification details.   
\*\*Output:\*\* The User Interface Specification data entity is updated. The system interface is modified to reflect the changes. The management event is recorded in the Purchase History data entity.  
  
## 1.31 Manage Personal Information Function   
\*\*Function ID:\*\* FR-31   
\*\*Description:\*\* Customers or administrators can view or update a customer's personal information. The system validates the changes and logs the event.   
\*\*Input:\*\* Personal information selection and action (view or update).   
\*\*Output:\*\* The Personal Information data entity is updated if changes are made. The viewing or update event is recorded in the Purchase History data entity.  
  
## 1.32 Manage Contact Information Function   
\*\*Function ID:\*\* FR-32   
\*\*Description:\*\* Customers or administrators can view or update a customer's contact information. The system validates the changes and logs the event.   
\*\*Input:\*\* Contact information selection and action (view or update).   
\*\*Output:\*\* The Contact Information data entity is updated if changes are made. The viewing or update event is recorded in the Purchase History data entity.  
  
## 1.33 Order Cancellation Function   
\*\*Function ID:\*\* FR-33   
\*\*Description:\*\* Customers or administrators can cancel an order if it is in an eligible status. The system updates the order and inventory.   
\*\*Input:\*\* Order selection and "Cancel Order" action.   
\*\*Output:\*\* The order is marked as "Cancelled" in the Order data entity. Inventory levels are updated. The cancellation event is recorded in the Purchase History data entity.

# External Description

# 2. External Interfaces  
  
## 2.1 User Interface  
  
The system interacts with users through a web-based graphical user interface (GUI) and mobile application interface (if applicable). These interfaces are designed to be intuitive and user-friendly, allowing customers and administrators to perform actions such as registration, login, managing personal and contact information, viewing and modifying shopping carts, and managing product and order data.  
  
### 2.1.1 Personal Information and Contact Information Interface  
- \*\*Description:\*\* This interface allows users to input, view, and update their personal and contact information.  
- \*\*Inputs:\*\* Name, email, password, phone number, address.  
- \*\*Outputs:\*\* Display of current information, confirmation messages, and error messages.  
- \*\*Interaction Method:\*\* Form-based input and output with validation feedback provided in real-time.  
  
### 2.1.2 Shopping Cart Interface  
- \*\*Description:\*\* This interface allows customers to view and modify the contents of their shopping cart.  
- \*\*Inputs:\*\* Product selection, quantity adjustments, remove actions.  
- \*\*Outputs:\*\* Display of cart items, total cost, and confirmation messages.  
- \*\*Interaction Method:\*\* Interactive form with real-time updates of inventory and cart status.  
  
### 2.1.3 Purchase History Interface  
- \*\*Description:\*\* This interface provides a detailed view of a customer's purchase history and order status.  
- \*\*Inputs:\*\* Navigation to the "Purchase History" section.  
- \*\*Outputs:\*\* List of orders with order number, date, product names, quantities, total amount, and status.  
- \*\*Interaction Method:\*\* Tabular or list-based display with filtering and sorting capabilities.  
  
### 2.1.4 Product Management Interface  
- \*\*Description:\*\* This interface is used by administrators to manage product data, including adding, updating, or removing products.  
- \*\*Inputs:\*\* Product name, description, price, category, inventory quantity.  
- \*\*Outputs:\*\* Display of product lists, confirmation messages, and error messages.  
- \*\*Interaction Method:\*\* Form-based input and output with interactive product lists and filters.  
  
### 2.1.5 Order Management Interface  
- \*\*Description:\*\* This interface allows administrators to manage order items and statuses.  
- \*\*Inputs:\*\* Order item selection, quantity change, or "Remove" action.  
- \*\*Outputs:\*\* Updated order status and confirmation messages.  
- \*\*Interaction Method:\*\* Interactive form with real-time updates to order and inventory data.  
  
### 2.1.6 Email Record Interface  
- \*\*Description:\*\* This interface allows both customers and administrators to view email records.  
- \*\*Inputs:\*\* Email record selection.  
- \*\*Outputs:\*\* Display of email content, including subject, body, date, and recipient.  
- \*\*Interaction Method:\*\* List-based display with clickable entries to view detailed email records.  
  
### 2.1.7 Payment Interface  
- \*\*Description:\*\* This interface allows users to view payment records.  
- \*\*Inputs:\*\* Payment record selection.  
- \*\*Outputs:\*\* Display of payment method, amount, date, and status.  
- \*\*Interaction Method:\*\* List-based or tabular display with filtering capabilities for easy navigation.  
  
### 2.1.8 Plugin Management Interface  
- \*\*Description:\*\* This interface allows administrators to install, configure, and manage plugins.  
- \*\*Inputs:\*\* Plugin details or file, configuration settings.  
- \*\*Outputs:\*\* Display of plugin status, configuration options, and system interface updates.  
- \*\*Interaction Method:\*\* Form-based input for plugin installation and configuration, with real-time feedback on success or failure.  
  
### 2.1.9 Administrator Management Interface  
- \*\*Description:\*\* This interface allows administrators to manage other administrator accounts.  
- \*\*Inputs:\*\* Administrator account selection and action (update or deactivate).  
- \*\*Outputs:\*\* Updated administrator information and confirmation messages.  
- \*\*Interaction Method:\*\* Interactive form with real-time validation and logging of management events.  
  
### 2.1.10 User Interface Specification Compliance Interface  
- \*\*Description:\*\* This interface allows administrators to check the system's compliance with defined UI specifications.  
- \*\*Inputs:\*\* Compliance check initiation.  
- \*\*Outputs:\*\* Compliance report with highlighted non-compliant elements.  
- \*\*Interaction Method:\*\* Button or menu-based initiation, followed by a detailed report display.  
  
## 2.2 Hardware Interface  
  
The system does not require direct interaction with external hardware devices for its core functionalities. However, the following hardware interfaces may be relevant for extended operations:  
  
### 2.2.1 Point of Sale (POS) Devices (if applicable)  
- \*\*Description:\*\* If the system is extended to support in-store purchases, it may interface with POS devices.  
- \*\*Inputs:\*\* Product scanning, payment processing, and inventory updates from the POS.  
- \*\*Outputs:\*\* Transaction confirmation, inventory updates, and order creation.  
- \*\*Interaction Method:\*\* Communication via standard POS APIs or serial interfaces.  
  
## 2.3 Software Interface  
  
The system interacts with various software components and data entities to manage its operations. These include internal data storage and external tools.  
  
### 2.3.1 Personal Information Data Entity  
- \*\*Description:\*\* Stores customer personal information such as name, email, and password.  
- \*\*Inputs:\*\* New or updated personal information from the user interface.  
- \*\*Outputs:\*\* Updated personal information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.2 Contact Information Data Entity  
- \*\*Description:\*\* Stores customer contact information such as phone number and address.  
- \*\*Inputs:\*\* New or updated contact information from the user interface.  
- \*\*Outputs:\*\* Updated contact information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.3 Product Data Entity  
- \*\*Description:\*\* Stores product information such as name, description, price, category, and inventory quantity.  
- \*\*Inputs:\*\* Product details from the user interface or system processes (e.g., checkout).  
- \*\*Outputs:\*\* Updated product information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.4 Category Data Entity  
- \*\*Description:\*\* Stores product categories and their associations with products.  
- \*\*Inputs:\*\* Category details from the user interface or system processes.  
- \*\*Outputs:\*\* Updated category information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.5 Inventory Data Entity  
- \*\*Description:\*\* Stores and manages inventory levels for all products.  
- \*\*Inputs:\*\* Inventory updates from the user interface or system processes (e.g., product addition, shopping cart modification, checkout).  
- \*\*Outputs:\*\* Updated inventory levels for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.6 Shopping Cart Data Entity  
- \*\*Description:\*\* Stores the contents of a customer's shopping cart, including product IDs, quantities, and pricing.  
- \*\*Inputs:\*\* Product selections and quantity adjustments from the user interface.  
- \*\*Outputs:\*\* Updated shopping cart for display and use in checkout or modification functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.7 CartItem Data Entity  
- \*\*Description:\*\* Stores individual items in the shopping cart, including product details and quantities.  
- \*\*Inputs:\*\* Cart item modifications from the user interface.  
- \*\*Outputs:\*\* Updated cart items for display and use in checkout or modification functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.8 Order Data Entity  
- \*\*Description:\*\* Stores customer orders, including order number, date, product details, customer information, and status.  
- \*\*Inputs:\*\* Order creation or modification from the user interface or system processes.  
- \*\*Outputs:\*\* Updated order information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.9 OrderItem Data Entity  
- \*\*Description:\*\* Stores individual items within an order, including product ID, quantity, and price.  
- \*\*Inputs:\*\* Order item modifications from the user interface or system processes.  
- \*\*Outputs:\*\* Updated order items for display and use in order management functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.10 Payment Data Entity  
- \*\*Description:\*\* Stores payment information, including payment method, amount, date, and status.  
- \*\*Inputs:\*\* Payment details from the user interface or external payment gateways.  
- \*\*Outputs:\*\* Updated payment information for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database writes for internal payments and API calls for external payment gateways.  
  
### 2.3.11 EmailRecord Data Entity  
- \*\*Description:\*\* Stores all email records, including order confirmations, account updates, and other notifications.  
- \*\*Inputs:\*\* Email content and metadata from the user interface or system-generated events.  
- \*\*Outputs:\*\* Updated email records for display and use in other functions.  
- \*\*Interaction Method:\*\* Direct database reads and writes with validation and logging.  
  
### 2.3.12 Plugin Data Entity  
- \*\*Description:\*\* Stores plugin details, configuration settings, and activation status.  
- \*\*Inputs:\*\* Plugin files or configuration settings from the user interface.  
- \*\*Outputs:\*\* Updated plugin data for system use and display.  
- \*\*Interaction Method:\*\* Direct database writes for plugin installation and configuration.  
  
### 2.3.13 Purchase History Data Entity  
- \*\*Description:\*\* Logs all user interactions related to account activity, product and order management, and system events.  
- \*\*Inputs:\*\* Event details from functional processes (e.g., login, logout, checkout).  
- \*\*Outputs:\*\* Updated event logs for display and use in audit or reporting functions.  
- \*\*Interaction Method:\*\* Direct database writes with event tracking and logging.  
  
### 2.3.14 User Interface Specification Data Entity  
- \*\*Description:\*\* Stores the current UI specification for the system, including design rules and functionality requirements.  
- \*\*Inputs:\*\* Updated UI specification details from the user interface.  
- \*\*Outputs:\*\* Updated UI specification for system use and display.  
- \*\*Interaction Method:\*\* Direct database writes with validation and logging.  
  
## 2.4 Communication Interface  
  
The system communicates with external services and users through various communication channels, including email notifications and web-based interactions.  
  
### 2.4.1 Email Notification Interface  
- \*\*Description:\*\* The system sends email notifications to users for account confirmation, order confirmation, and other system events.  
- \*\*Inputs:\*\* Email content and recipient information from the system.  
- \*\*Outputs:\*\* Email sent to the specified address.  
- \*\*Interaction Method:\*\* Integration with an email service provider (e.g., SMTP, third-party email APIs) to send and log emails.  
  
### 2.4.2 Payment Gateway Interface  
- \*\*Description:\*\* The system communicates with external payment gateways to process customer payments.  
- \*\*Inputs:\*\* Payment method and payment details from the user interface.  
- \*\*Outputs:\*\* Payment confirmation or error message from the payment gateway.  
- \*\*Interaction Method:\*\* API calls to the payment gateway with secure data exchange and response handling.  
  
### 2.4.3 Web Browser Communication Interface  
- \*\*Description:\*\* The system is accessed through web browsers, and interactions are handled via HTTP/HTTPS protocols.  
- \*\*Inputs:\*\* User actions via browser (e.g., form submissions, navigation).  
- \*\*Outputs:\*\* Dynamic web page updates and data retrieval.  
- \*\*Interaction Method:\*\* RESTful API or web service communication with client-side rendering or server-side rendering depending on the architecture.  
  
### 2.4.4 Plugin Communication Interface  
- \*\*Description:\*\* The system may communicate with external plugins to extend its functionality.  
- \*\*Inputs:\*\* Plugin configuration or activation signals from the user interface.  
- \*\*Outputs:\*\* Updated system functionality or interface changes based on the plugin.  
- \*\*Interaction Method:\*\* Plugin APIs or system hooks to integrate and activate new functionalities.  
  
---  
  
This section defines all external interfaces that the system interacts with, ensuring clarity and consistency for developers and stakeholders. Each interface is categorized based on its role and method of interaction, with clear definitions and input/output examples to reduce implementation ambiguity.

# Use Case

Use Case Name: Customer Registration   
Use Case ID: UC-01   
Actors: Customer, Administrator   
Preconditions:   
1. The system is accessible to the user.   
2. The customer has not yet registered an account.   
3. The customer has a valid email address and necessary personal information.   
  
Postconditions:   
1. The customer has a successfully created account.   
2. The customer is logged in to the system.   
3. The customer’s information is stored in the Personal Information data entity.   
4. The customer is assigned a unique identifier for future interactions.   
  
Main Flow:   
1. The customer navigates to the registration page of the system interface.   
2. The system displays the registration form, which includes fields for name, email, password, and optional contact information.   
3. The customer fills in the required information and clicks the "Register" button.   
4. The system validates the email format and checks if the email is already registered.   
5. If the email is valid and not already registered, the system creates a new customer account and stores the information in the Personal Information data entity.   
6. The system sends a confirmation email to the customer’s provided email address.   
7. The customer receives and clicks the confirmation link in the email.   
8. The system confirms the customer’s email and logs the customer in automatically.   
  
Alternative Flow:   
1. If the email format is invalid in step 4, the system displays an error message and prompts the customer to correct the email.   
2. If the email is already registered in step 4, the system displays an error message indicating that the email is already in use and prompts the customer to provide a different email.   
3. If the customer fails to click the confirmation link within a specified time frame, the system may require the customer to request a new confirmation email.  
  
Use Case Name: Customer Login   
Use Case ID: UC-02   
Actors: Customer, System Interface Specification   
Preconditions:   
1. The system is accessible to the user.   
2. The customer has already registered an account.   
3. The customer has valid login credentials (email and password).   
4. The customer’s email has been confirmed.   
  
Postconditions:   
1. The customer is successfully logged into the system.   
2. The system displays the customer’s personalized dashboard or homepage.   
3. The system logs the login event in the Purchase History data entity.   
  
Main Flow:   
1. The customer navigates to the login page of the system interface.   
2. The system displays the login form, which includes fields for email and password.   
3. The customer enters their registered email and password and clicks the "Login" button.   
4. The system validates the email and password against the stored data in the Personal Information data entity.   
5. If the credentials are valid, the system logs the customer in and redirects them to their personalized homepage.   
6. The system records the login event in the Purchase History data entity.   
  
Alternative Flow:   
1. If the email is not found in the system in step 4, the system displays an error message indicating an invalid email.   
2. If the password is incorrect in step 4, the system displays an error message and prompts the customer to re-enter the password.   
3. If the customer exceeds the maximum number of login attempts, the system locks the account temporarily and notifies the customer via the Contact Information data entity.   
4. If the customer’s email has not been confirmed, the system displays a message prompting them to check their email for a confirmation link.  
  
Use Case Name: Customer Logout   
Use Case ID: UC-03   
Actors: Customer, System Interface Specification   
Preconditions:   
1. The system is accessible to the user.   
2. The customer is currently logged in.   
3. The customer has initiated the logout process through the system interface.   
  
Postconditions:   
1. The customer is no longer authenticated in the system.   
2. The session associated with the customer is terminated.   
3. The system redirects the customer to the homepage or login page.   
4. The logout event is recorded in the Purchase History data entity.   
  
Main Flow:   
1. The customer accesses the system interface and locates the logout option, typically in a user menu.   
2. The customer clicks the "Logout" button.   
3. The system confirms the customer's intent to log out (optional confirmation step).   
4. The system terminates the customer's session and invalidates the current authentication token.   
5. The system redirects the customer to the homepage or login page.   
6. The system logs the logout event in the Purchase History data entity.   
  
Alternative Flow:   
1. If the system does not support session termination for security reasons, the customer may be logged out automatically after a period of inactivity.   
2. If the customer closes the browser or navigates away without using the logout button, the system may log them out after a session timeout.   
3. If the logout process fails due to technical issues, the system displays an error message and prompts the customer to try again or contact support via the Contact Information data entity.  
  
Use Case Name: Personal Information Update   
Use Case ID: UC-04   
Actors: Customer, System Interface Specification   
Preconditions:   
1. The system is accessible to the user.   
2. The customer is already logged in.   
3. The customer has an existing account in the Personal Information data entity.   
  
Postconditions:   
1. The customer’s updated personal information is stored in the Personal Information data entity.   
2. The system displays a confirmation message that the update was successful.   
3. The update event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer navigates to the account settings page through the system interface.   
2. The system displays the current personal information fields, such as name, email, and contact information.   
3. The customer modifies the desired fields and clicks the "Update" button.   
4. The system validates the updated information (e.g., email format, required fields).   
5. If validation is successful, the system updates the Personal Information data entity with the new data.   
6. The system sends a confirmation email to the customer to notify them of the successful update.   
7. The system displays a success message to the customer.   
  
Alternative Flow:   
1. If the updated email is already registered in the system in step 4, the system displays an error message and prompts the customer to provide a different email.   
2. If the updated information fails validation (e.g., invalid format, missing required fields), the system displays an error message and prompts the customer to correct the data.   
3. If the update fails due to technical issues, the system displays an error message and suggests the customer try again later or contact support via the Contact Information data entity.  
  
Use Case Name: Contact Information Management   
Use Case ID: UC-05   
Actors: Customer, Administrator, Contact Information   
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has an existing account in the Personal Information data entity.   
4. The administrator has access to the system's user management features.   
  
Postconditions:   
1. The customer's contact information is updated or viewed in the Contact Information data entity.   
2. If an administrator modifies the contact information, the changes are reflected in the data entity.   
3. The system provides a confirmation message for any successful update.   
4. Any modification event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer accesses the account settings page via the system interface.   
2. The system displays the current contact information, such as phone number, address, and alternate email.   
3. The customer selects the option to update or view their contact information.   
4. If updating, the customer modifies the relevant fields and clicks the "Save Changes" button.   
5. The system validates the updated information (e.g., phone number format, address completeness).   
6. If validation is successful, the system updates the Contact Information data entity.   
7. The system displays a success message confirming the update.   
  
Alternative Flow:   
1. If the customer enters invalid data (e.g., incorrect phone number format), the system displays an error message and prompts the customer to correct it.   
2. If the customer does not make any changes and clicks "Save Changes," the system displays a message indicating no updates were made.   
3. If an administrator modifies the customer's contact information, the system updates the Contact Information data entity and logs the event.   
4. If the update fails due to system errors, the system displays an error message and suggests the customer retry later or contact support via the Contact Information data entity.  
  
Use Case Name: Purchase History Viewing   
Use Case ID: UC-06   
Actors: Customer, System Interface Specification, Purchase History   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is already logged in.   
3. The customer has at least one order recorded in the Purchase History data entity.   
4. The system interface includes a section for viewing purchase history.   
  
Postconditions:   
1. The customer is able to view their purchase history, including order details and status.   
2. The system displays the relevant data from the Purchase History data entity.   
3. If any changes are made to the display settings (e.g., sorting, filtering), these are applied and saved.   
4. The viewing event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer logs in and navigates to the "Purchase History" section in their account dashboard.   
2. The system retrieves the customer's purchase history from the Purchase History data entity.   
3. The system displays the list of orders, including order number, date, product names, quantities, total amount, and order status.   
4. The customer can click on an individual order to view more detailed information, such as shipping address and payment method.   
5. The system provides options to filter or sort the purchase history (e.g., by date or product category).   
  
Alternative Flow:   
1. If no purchase history is found for the customer in step 2, the system displays a message indicating that there are no previous orders.   
2. If the system fails to load the purchase history due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer attempts to access the purchase history without being logged in, they are redirected to the login page.   
4. If the administrator modifies the displayed order data for audit or support purposes, the changes are synchronized with the Purchase History data entity.  
  
Use Case Name: Product Addition   
Use Case ID: UC-07   
Actors: Administrator, Product, Category, Inventory   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in.   
3. The system has access to the Category and Inventory data entities.   
4. The administrator has the necessary permissions to add products.   
  
Postconditions:   
1. A new product is successfully added to the system.   
2. The product information is stored in the Product data entity.   
3. The product is assigned to the appropriate category.   
4. Inventory levels are updated accordingly.   
5. The addition event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Add Product" section through the system interface.   
2. The system displays a form for entering product details, including name, description, price, category, and inventory quantity.   
3. The administrator fills in the required product information and selects the appropriate category.   
4. The administrator specifies the initial inventory quantity and clicks the "Add Product" button.   
5. The system validates the product data (e.g., required fields, valid price format).   
6. If validation is successful, the system adds the product to the Product data entity and associates it with the selected category.   
7. The system updates the Inventory data entity to reflect the new product and its quantity.   
8. The system displays a confirmation message indicating the product was successfully added.   
  
Alternative Flow:   
1. If the product name or category is missing in step 3, the system displays an error message and prompts the administrator to provide the missing information.   
2. If the price format is invalid in step 5, the system displays an error message and prompts the administrator to correct it.   
3. If the selected category does not exist, the system displays an error message and prompts the administrator to choose a valid category.   
4. If the system fails to add the product due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.  
  
Use Case Name: Product Removal   
Use Case ID: UC-08   
Actors: Administrator, Product, Inventory, Category   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in.   
3. The administrator has the necessary permissions to remove products.   
4. The product to be removed exists in the Product data entity.   
  
Postconditions:   
1. The product is removed from the system.   
2. The product is no longer associated with any category.   
3. The Inventory data entity is updated to reflect the removal of the product.   
4. The removal event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Manage Products" section through the system interface.   
2. The system displays a list of available products, including product name, category, and inventory status.   
3. The administrator selects the product to be removed and clicks the "Remove Product" button.   
4. The system prompts the administrator to confirm the removal of the product.   
5. The administrator confirms the removal.   
6. The system removes the product from the Product data entity and disassociates it from its category.   
7. The system updates the Inventory data entity to reflect the removal.   
8. The system displays a confirmation message indicating the product was successfully removed.   
  
Alternative Flow:   
1. If the product is currently in the Shopping Cart or has active orders, the system displays a warning message and prevents removal until all references are resolved.   
2. If the administrator cancels the removal in step 5, the system cancels the operation and returns to the product management page.   
3. If the system fails to remove the product due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.  
  
Use Case Name: Product Information Update   
Use Case ID: UC-09   
Actors: Administrator, Product, Category, Inventory   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in.   
3. The administrator has the necessary permissions to update product information.   
4. The product to be updated exists in the Product data entity.   
  
Postconditions:   
1. The product's updated information is stored in the Product data entity.   
2. The product is correctly associated with the updated category.   
3. The Inventory data entity is updated to reflect any changes in stock or availability.   
4. The update event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Manage Products" section through the system interface.   
2. The system displays a list of products, including details such as product name, category, price, and inventory status.   
3. The administrator selects the product to be updated and clicks the "Edit Product" button.   
4. The system opens the product information form with the current details pre-filled.   
5. The administrator modifies the necessary fields, such as product name, description, price, category, or inventory quantity.   
6. The administrator clicks the "Save Changes" button.   
7. The system validates the updated information (e.g., price format, category existence).   
8. If validation is successful, the system updates the Product data entity and adjusts the category and inventory associations accordingly.   
9. The system displays a confirmation message indicating the product information has been successfully updated.   
  
Alternative Flow:   
1. If the product name or category is missing in step 5, the system displays an error message and prompts the administrator to provide the missing information.   
2. If the price format is invalid in step 7, the system displays an error message and prompts the administrator to correct it.   
3. If the selected category does not exist, the system displays an error message and prompts the administrator to choose a valid category.   
4. If the system fails to update the product information due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.  
  
Use Case Name: Product Categorization   
Use Case ID: UC-10   
Actors: Administrator, Product, Category   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in.   
3. The administrator has permission to manage product categories.   
4. The product exists in the Product data entity.   
5. The category exists in the Category data entity.   
  
Postconditions:   
1. The product is correctly assigned to the selected category in the Category data entity.   
2. The categorization event is logged in the Purchase History data entity for audit purposes.   
3. The product’s category information is updated in the Product data entity.   
4. The system displays a confirmation message that the categorization was successful.   
  
Main Flow:   
1. The administrator navigates to the "Manage Products" section through the system interface.   
2. The system displays a list of products, including product name, current category, and other relevant details.   
3. The administrator selects a product and clicks the "Assign Category" option.   
4. The system opens a form or dialog allowing the administrator to choose a new category for the product.   
5. The administrator selects the appropriate category from the list of available categories.   
6. The administrator clicks the "Save Changes" button.   
7. The system validates the selected category (e.g., ensuring it is a valid entity in the Category data entity).   
8. If validation is successful, the system updates the product's category association in the Product data entity and links it to the selected category.   
9. The system logs the categorization event in the Purchase History data entity.   
10. The system displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the selected category is invalid or does not exist in step 7, the system displays an error message and prompts the administrator to select a valid category.   
2. If the product is already in the selected category, the system displays a message indicating no changes were made.   
3. If the system fails to update the category due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.  
  
Use Case Name: Inventory Management   
Use Case ID: UC-11   
Actors: Administrator, Product, Inventory   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in.   
3. The administrator has the necessary permissions to manage inventory.   
4. The system contains at least one product in the Product data entity.   
5. The Inventory data entity is available for updating stock levels.   
  
Postconditions:   
1. The inventory levels for one or more products are updated in the Inventory data entity.   
2. The system displays a confirmation message for the inventory update.   
3. The update event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Inventory Management" section via the system interface.   
2. The system displays a list of products with their current inventory levels and status.   
3. The administrator selects a product and modifies its inventory quantity.   
4. The administrator clicks the "Update Inventory" button.   
5. The system validates the updated inventory quantity (e.g., ensuring it is a non-negative number).   
6. If validation is successful, the system updates the Inventory data entity with the new quantity.   
7. The system displays a success message confirming the inventory update.   
  
Alternative Flow:   
1. If the entered inventory quantity is negative or invalid in step 5, the system displays an error message and prompts the administrator to enter a valid quantity.   
2. If the selected product is not found in the Product data entity, the system displays an error message and prevents the update.   
3. If the system fails to update the inventory due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.  
  
Use Case Name: Shopping Cart Creation   
Use Case ID: UC-12   
Actors: Customer, Shopping Cart, Product, Inventory, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The system contains at least one available product in the Product data entity.   
4. The Inventory data entity reflects the current stock levels for the products.   
5. The system interface includes a section for adding products to the shopping cart.   
  
Postconditions:   
1. The product is added to the customer's Shopping Cart data entity.   
2. The inventory quantity for the product is updated in the Inventory data entity.   
3. The system displays a confirmation message that the product was successfully added.   
4. The cart is accessible for further modifications or checkout.   
  
Main Flow:   
1. The customer logs in and browses the product catalog via the system interface.   
2. The customer selects a product and clicks the "Add to Cart" button.   
3. The system checks the product's availability in the Inventory data entity.   
4. If the product is available, the system adds the product to the customer's Shopping Cart data entity.   
5. The system updates the inventory quantity by reducing the available stock by one.   
6. The system displays a confirmation message indicating the product has been added to the cart.   
7. The customer can review the cart contents and proceed to checkout or continue shopping.   
  
Alternative Flow:   
1. If the product is out of stock in step 3, the system displays an error message and prevents the addition.   
2. If the customer attempts to add more units than available, the system displays an error message and shows the maximum allowable quantity.   
3. If the system fails to update the inventory or add the product due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
4. If the customer adds the same product multiple times, the system updates the quantity in the cart instead of creating a new entry.  
  
Use Case Name: Shopping Cart Modification   
Use Case ID: UC-13   
Actors: Customer, Shopping Cart, Product, Inventory, System Interface Specification   
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one product in their Shopping Cart data entity.   
4. The system interface provides a way to modify the cart (e.g., update quantity, remove product).   
5. The Inventory data entity is available to update stock levels if needed.   
  
Postconditions:   
1. The customer's Shopping Cart is updated with the modified product quantity or removed product.   
2. The inventory levels are adjusted accordingly for any product quantity changes.   
3. The system displays the updated cart and any relevant confirmation messages.   
4. The modification event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer logs in and navigates to the "Shopping Cart" section via the system interface.   
2. The system displays the current contents of the customer's shopping cart, including product names, quantities, and prices.   
3. The customer selects a product and modifies its quantity or clicks the "Remove" button.   
4. The system validates the new quantity (e.g., not exceeding available stock) or confirms the removal.   
5. If valid, the system updates the Shopping Cart data entity with the new quantity or removes the product.   
6. If the quantity is reduced or increased, the system updates the inventory levels in the Inventory data entity.   
7. The system displays a confirmation message indicating the cart was successfully modified.   
  
Alternative Flow:   
1. If the customer tries to set a quantity higher than available stock in step 4, the system displays an error message and shows the maximum allowable quantity.   
2. If the customer enters an invalid quantity (e.g., negative number or non-numeric input), the system displays an error message and prompts for correction.   
3. If the system fails to update the cart or inventory due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
4. If the customer removes the last item from the cart, the system displays a message indicating the cart is now empty.  
  
Use Case Name: Shopping Cart Viewing   
Use Case ID: UC-14   
Actors: Customer, Shopping Cart, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one product in their Shopping Cart data entity.   
4. The system interface provides a "View Cart" feature.   
  
Postconditions:   
1. The customer is able to view the contents of their shopping cart.   
2. The system displays product details, including quantity, price, and total.   
3. The system provides options for further actions, such as updating the cart or proceeding to checkout.   
4. The viewing event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer logs in and navigates to the "Shopping Cart" section via the system interface.   
2. The system retrieves the customer's shopping cart data from the Shopping Cart data entity.   
3. The system displays the list of items in the cart, including product name, quantity, price, and total amount.   
4. The system shows a summary with the total cost and any applicable discounts or shipping fees.   
5. The customer can review the cart and choose to proceed to checkout or continue modifying the cart.   
  
Alternative Flow:   
1. If no items are found in the cart in step 2, the system displays a message indicating the cart is empty.   
2. If the system fails to load the cart due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer's session expires while viewing the cart, they are redirected to the login page.  
  
Use Case Name: Checkout   
Use Case ID: UC-15   
Actors: Customer, Shopping Cart, Product, Inventory, Payment, Order, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one item in their Shopping Cart data entity.   
4. The system interface includes a checkout process.   
5. The customer has valid payment information.   
6. The Inventory data entity reflects current stock levels for the items in the cart.   
  
Postconditions:   
1. The customer's order is successfully created and stored in the Order data entity.   
2. The Shopping Cart data entity is cleared of the purchased items.   
3. The inventory levels for the purchased items are updated in the Inventory data entity.   
4. The payment is processed and recorded in the Payment data entity.   
5. The customer receives an Order Confirmation Email.   
6. The checkout event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer navigates to the "Shopping Cart" section and clicks the "Proceed to Checkout" button.   
2. The system displays the checkout page with the cart contents, total amount, and shipping information.   
3. The customer reviews the items and confirms the quantities.   
4. The customer selects a shipping address from their Contact Information data entity or enters a new one.   
5. The customer selects a payment method (e.g., credit card, PayPal) and enters the required payment details.   
6. The system validates the payment information and processes the payment through the Payment data entity.   
7. If the payment is successful, the system creates a new order in the Order data entity and associates it with the customer.   
8. The system updates the Inventory data entity to reduce the stock of the purchased items.   
9. The system clears the Shopping Cart data entity of the purchased items.   
10. The system generates and sends an Order Confirmation Email to the customer.   
11. The system displays a confirmation message to the customer that the checkout was successful.   
  
Alternative Flow:   
1. If the customer cancels the checkout process at any point, the system returns to the Shopping Cart page without making any changes.   
2. If the customer selects an invalid or incomplete shipping address in step 4, the system displays an error message and prompts the customer to correct or add a valid address.   
3. If the payment validation fails in step 6 (e.g., invalid card number, insufficient funds), the system displays an error message and prompts the customer to provide a valid payment method.   
4. If the system fails to process the payment due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
5. If the inventory is insufficient for the items in the cart (e.g., out of stock), the system displays an error message and prevents the checkout until the issue is resolved.   
6. If the system fails to send the Order Confirmation Email, it logs the failure and displays a message to the customer, allowing them to request the email manually or contact support via the Contact Information data entity.  
  
Use Case Name: Order Confirmation   
Use Case ID: UC-16   
Actors: Customer, Order, Email, Purchase History, Payment   
Preconditions:   
1. The system is accessible to the user.   
2. The customer has completed the checkout process (UC-15).   
3. The payment for the order has been successfully processed and stored in the Payment data entity.   
4. The order details are stored in the Order data entity.   
5. The system has access to the Email service for sending confirmation messages.   
  
Postconditions:   
1. The customer receives an Order Confirmation Email.   
2. The order is marked as confirmed in the Order data entity.   
3. The confirmation event is logged in the Purchase History data entity for audit purposes.   
4. The system displays a confirmation message to the customer.   
  
Main Flow:   
1. After the customer completes the checkout process, the system automatically triggers the order confirmation process.   
2. The system retrieves the order details from the Order data entity, including order number, items, total amount, and shipping information.   
3. The system generates an Order Confirmation Email containing the order summary and a confirmation message.   
4. The system sends the email to the customer's registered email address through the Email data entity.   
5. The system updates the Order data entity to mark the order as confirmed.   
6. The system logs the confirmation event in the Purchase History data entity.   
7. The system displays a confirmation message to the customer on the interface, indicating the order was successfully confirmed.   
  
Alternative Flow:   
1. If the system fails to send the Order Confirmation Email, it logs the failure and displays a message to the customer, allowing them to request the email manually or contact support via the Contact Information data entity.   
2. If the customer's email address is invalid or not found in the Personal Information data entity, the system displays an error message and prompts the customer to update their contact information.   
3. If the system fails to update the Order data entity to mark it as confirmed, an error message is displayed, and the administrator is notified via the Contact Information data entity to resolve the issue.   
4. If the customer navigates away before the confirmation is completed, the system may still process the confirmation in the background and send the email.  
  
Use Case Name: Payment Processing   
Use Case ID: UC-17   
Actors: Customer, Payment, Order, Inventory, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one item in their Shopping Cart data entity.   
4. The customer has provided valid payment information.   
5. The inventory for the cart items is sufficient.   
6. The system interface includes a payment processing feature.   
  
Postconditions:   
1. The payment is successfully processed and recorded in the Payment data entity.   
2. The order is created and stored in the Order data entity.   
3. The inventory levels for the purchased items are updated in the Inventory data entity.   
4. The system displays a confirmation message to the customer.   
5. The payment event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer navigates to the "Checkout" page and proceeds to the payment section.   
2. The system displays the payment form, including fields for payment method (e.g., credit card, PayPal) and relevant details.   
3. The customer selects a payment method and enters the required payment information.   
4. The system validates the payment information (e.g., card expiration date, security code).   
5. If valid, the system processes the payment via the Payment data entity.   
6. The system confirms the payment success and creates an order in the Order data entity.   
7. The system updates the Inventory data entity to reduce the stock of the purchased items.   
8. The system clears the customer's Shopping Cart data entity.   
9. The system displays a success message to the customer indicating the payment and order were processed.   
  
Alternative Flow:   
1. If the customer enters invalid payment information in step 4, the system displays an error message and prompts the customer to correct it.   
2. If the payment processing fails (e.g., declined card, server error), the system displays an error message and allows the customer to try again or choose an alternative payment method.   
3. If the inventory is insufficient during the payment process, the system halts the transaction and displays an error message, prompting the customer to adjust their cart.   
4. If the system fails to update the inventory or create the order due to technical issues, an error message is displayed, and the customer is prompted to contact support via the Contact Information data entity.   
5. If the customer cancels the payment at any step, the system returns to the cart without processing the transaction.  
  
Use Case Name: Order Confirmation Email Sending   
Use Case ID: UC-16   
Actors: Customer, Order, Email, Purchase History, Payment   
Preconditions:   
1. The system is accessible to the user.   
2. The customer has completed the checkout process (UC-15).   
3. The payment for the order has been successfully processed and stored in the Payment data entity.   
4. The order details are stored in the Order data entity.   
5. The system has access to the Email service for sending confirmation messages.   
  
Postconditions:   
1. The customer receives an Order Confirmation Email.   
2. The order is marked as confirmed in the Order data entity.   
3. The confirmation event is logged in the Purchase History data entity for audit purposes.   
4. The system displays a confirmation message to the customer.   
  
Main Flow:   
1. After the customer completes the checkout process, the system automatically triggers the order confirmation process.   
2. The system retrieves the order details from the Order data entity, including order number, items, total amount, and shipping information.   
3. The system generates an Order Confirmation Email containing the order summary and a confirmation message.   
4. The system sends the email to the customer's registered email address through the Email data entity.   
5. The system updates the Order data entity to mark the order as confirmed.   
6. The system logs the confirmation event in the Purchase History data entity.   
7. The system displays a confirmation message to the customer on the interface, indicating the order was successfully confirmed.   
  
Alternative Flow:   
1. If the system fails to send the Order Confirmation Email, it logs the failure and displays a message to the customer, allowing them to request the email manually or contact support via the Contact Information data entity.   
2. If the customer's email address is invalid or not found in the Personal Information data entity, the system displays an error message and prompts the customer to update their contact information.   
3. If the system fails to update the Order data entity to mark it as confirmed, an error message is displayed, and the administrator is notified via the Contact Information data entity to resolve the issue.   
4. If the customer navigates away before the confirmation is completed, the system may still process the confirmation in the background and send the email.  
  
Use Case Name: Administrator Login   
Use Case ID: UC-00   
Actors: Administrator, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The administrator has an existing account in the system.   
3. The administrator has the correct login credentials (username and password).   
4. The administrator is not currently logged in.   
  
Postconditions:   
1. The administrator is successfully logged in to the system.   
2. The system displays the administrator dashboard.   
3. The login event is recorded in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the login page of the system interface.   
2. The system displays the login form, which includes fields for username and password.   
3. The administrator enters their username and password and clicks the "Login" button.   
4. The system validates the entered credentials against the stored data in the administrator account.   
5. If the credentials are valid, the system logs the administrator in and redirects them to the administrator dashboard.   
6. The system logs the login event in the Purchase History data entity.   
  
Alternative Flow:   
1. If the entered username is invalid or not found, the system displays an error message and prompts the administrator to re-enter the username.   
2. If the entered password is incorrect, the system displays an error message and allows the administrator to re-enter the password.   
3. If the administrator exceeds the maximum number of failed login attempts, the system may lock the account temporarily and notify the administrator via the Contact Information data entity.   
4. If the system fails to authenticate the administrator due to technical issues, an error message is displayed, and the administrator is prompted to try again or contact support via the Contact Information data entity.  
  
Use Case Name: Administrator Logout   
Use Case ID: UC-18   
Actors: Administrator, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The administrator is currently logged in.   
3. The administrator has initiated the logout process through the system interface.   
  
Postconditions:   
1. The administrator is no longer authenticated in the system.   
2. The session associated with the administrator is terminated.   
3. The system redirects the administrator to the homepage or login page.   
4. The logout event is recorded in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator accesses the system interface and locates the logout option, typically in a user menu.   
2. The administrator clicks the "Logout" button.   
3. The system confirms the administrator's intent to log out (optional confirmation step).   
4. The system terminates the administrator's session and invalidates the current authentication token.   
5. The system redirects the administrator to the homepage or login page.   
6. The system logs the logout event in the Purchase History data entity.   
  
Alternative Flow:   
1. If the system does not support session termination for security reasons, the administrator may be logged out automatically after a period of inactivity.   
2. If the administrator closes the browser or navigates away without using the logout button, the system may log them out after a session timeout.   
3. If the logout process fails due to technical issues, the system displays an error message and prompts the administrator to try again or contact support via the Contact Information data entity.  
  
Use Case Name: Plugin Development   
Use Case ID: UC-19   
Actors: Administrator, Plugin, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to develop or install plugins.   
3. The Plugin data entity is available for storing plugin metadata and configurations.   
4. The system interface provides a plugin management section.   
  
Postconditions:   
1. A new plugin is successfully added to the system and stored in the Plugin data entity.   
2. The plugin is activated or configured as needed.   
3. The system interface is updated to reflect the new plugin functionality.   
4. The development or installation event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Plugin Management" section via the system interface.   
2. The system displays a list of available plugins or provides an option to upload a new plugin.   
3. The administrator selects the option to develop or install a new plugin and uploads or inputs the plugin details.   
4. The system validates the plugin format and compatibility with the current system.   
5. If validation is successful, the system stores the plugin in the Plugin data entity.   
6. The administrator configures the plugin settings as needed through the system interface.   
7. The system activates the plugin and updates the interface to reflect the new functionality.   
8. The system logs the plugin development or installation event in the Purchase History data entity.   
9. The system displays a confirmation message indicating the plugin was successfully added and activated.   
  
Alternative Flow:   
1. If the plugin format is invalid in step 4, the system displays an error message and prompts the administrator to upload a valid plugin.   
2. If the plugin is not compatible with the system version, the system displays a compatibility warning and prevents the installation.   
3. If the system fails to activate the plugin due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
4. If the administrator cancels the plugin installation or development process, the system reverts to the plugin management page without making changes.  
  
Use Case Name: Plugin Configuration   
Use Case ID: UC-20   
Actors: Administrator, Plugin, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to configure plugins.   
3. The Plugin data entity contains at least one installed plugin.   
4. The system interface includes a plugin configuration section.   
  
Postconditions:   
1. The plugin settings are updated in the Plugin data entity.   
2. The plugin functions according to the new configuration.   
3. The system interface reflects the updated plugin settings.   
4. The configuration event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Plugin Management" section via the system interface.   
2. The system displays a list of installed plugins, including options to configure each one.   
3. The administrator selects a plugin and clicks the "Configure" button.   
4. The system opens a configuration form or interface for the selected plugin, displaying the current settings.   
5. The administrator modifies the plugin settings as needed and clicks the "Save Configuration" button.   
6. The system validates the configuration data (e.g., required fields, valid parameters).   
7. If validation is successful, the system updates the Plugin data entity with the new configuration.   
8. The system applies the new configuration and updates the system interface to reflect the changes.   
9. The system logs the configuration event in the Purchase History data entity.   
10. The system displays a confirmation message indicating the plugin was successfully configured.   
  
Alternative Flow:   
1. If the plugin configuration data is invalid in step 6, the system displays an error message and prompts the administrator to correct the input.   
2. If the plugin is not compatible with the new configuration, the system displays a warning and prevents the update.   
3. If the system fails to apply the new configuration due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
4. If the administrator cancels the configuration process, the system reverts to the plugin management page without making any changes.  
  
Use Case Name: User Interface Specification Compliance Checking   
Use Case ID: UC-21   
Actors: Administrator, User Interface Specification, Plugin, Product, Order, Contact Information, Purchase History   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to review and enforce interface compliance.   
3. The User Interface Specification data entity contains the required interface rules and standards.   
4. The system interface includes a compliance checking tool or report.   
  
Postconditions:   
1. The system interface is verified for compliance with the User Interface Specification.   
2. Non-compliant elements are identified and reported.   
3. The compliance check event is logged in the Purchase History data entity for audit purposes.   
4. The administrator receives a summary of compliance status and any required actions.   
  
Main Flow:   
1. The administrator navigates to the "Interface Compliance Check" section via the system interface.   
2. The system retrieves the latest User Interface Specification data entity.   
3. The system compares the current interface elements (e.g., layout, form fields, navigation, buttons) against the specification.   
4. The system identifies any discrepancies or non-compliant elements and generates a report.   
5. The system displays the compliance check results to the administrator, highlighting any issues.   
6. The administrator reviews the report and decides on corrective actions if needed.   
7. The system logs the compliance check event in the Purchase History data entity.   
8. The system displays a confirmation message that the compliance check was completed.   
  
Alternative Flow:   
1. If the User Interface Specification data entity is not available or outdated in step 2, the system displays an error message and prompts the administrator to update or provide the correct specification.   
2. If no non-compliant elements are found, the system displays a message indicating that the interface fully complies with the specification.   
3. If the compliance check fails due to technical issues, the system displays an error message and prompts the administrator to retry the operation or contact support via the Contact Information data entity.   
4. If the administrator requests to export the compliance report, the system generates and provides the report in a specified format (e.g., PDF, CSV).  
  
Use Case Name: Manage Cart Item   
Use Case ID: UC-22   
Actors: Customer, Shopping Cart, Product, Inventory, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one item in their Shopping Cart data entity.   
4. The system interface provides a way to manage individual cart items (e.g., edit quantity, remove item, apply discount).   
5. The Inventory data entity is available to update stock levels if needed.   
  
Postconditions:   
1. The customer's Shopping Cart data entity is updated with the modified cart item (e.g., quantity changed or item removed).   
2. The inventory levels are adjusted accordingly for any product quantity changes.   
3. The system displays the updated cart contents and any relevant confirmation messages.   
4. The modification event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The customer logs in and navigates to the "Shopping Cart" section via the system interface.   
2. The system displays a list of cart items, including product name, quantity, price, and total.   
3. The customer selects a specific cart item and chooses an action (e.g., edit quantity, remove item, apply discount).   
4. If editing quantity, the customer updates the number of units for the selected product and clicks the "Update" button.   
5. The system validates the updated quantity (e.g., not exceeding available stock, not less than one).   
6. If valid, the system updates the Cart Item in the Shopping Cart data entity and adjusts the inventory levels in the Inventory data entity if necessary.   
7. If removing an item, the system updates the Shopping Cart data entity by removing the selected item and restoring the inventory quantity.   
8. If applying a discount, the system checks if the selected product supports discounts and updates the cart accordingly.   
9. The system displays a confirmation message indicating the cart item was successfully managed.   
10. The system logs the cart item management event in the Purchase History data entity.   
  
Alternative Flow:   
1. If the customer tries to set a quantity higher than available stock in step 5, the system displays an error message and shows the maximum allowable quantity.   
2. If the customer enters an invalid quantity (e.g., negative number or non-numeric input), the system displays an error message and prompts for correction.   
3. If the system fails to update the cart or inventory due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
4. If the customer removes an item, the system does not update the inventory if the item was added multiple times by the same customer.   
5. If the discount is not applicable to the selected item, the system displays an error message and does not apply the discount.   
6. If the customer cancels the cart item management process, the system reverts to the cart view without making any changes.  
  
Use Case Name: Manage OrderItem   
Use Case ID: UC-23   
Actors: Administrator, Order, OrderItem, Product, Inventory, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage order items.   
3. The system contains at least one order in the Order data entity.   
4. The OrderItem data entity is available for viewing and modifying individual items within an order.   
5. The Inventory data entity is accessible for updating product stock levels if needed.   
6. The system interface provides a section for viewing and managing order items.   
  
Postconditions:   
1. The selected OrderItem is updated or modified in the OrderItem data entity.   
2. If an item is removed or quantity is adjusted, the Inventory data entity is updated accordingly.   
3. The system displays a confirmation message for the OrderItem management action.   
4. The modification event is logged in the Purchase History data entity for audit purposes.   
  
Main Flow:   
1. The administrator navigates to the "Order Management" section via the system interface.   
2. The system displays a list of orders, and the administrator selects a specific order to view.   
3. The system opens the selected order and shows the OrderItem details (e.g., product name, quantity, price, status).   
4. The administrator selects an OrderItem and chooses an action (e.g., edit quantity, remove item, update status).   
5. If editing quantity, the administrator updates the number of units and clicks the "Save Changes" button.   
6. The system validates the new quantity (e.g., ensuring it is not higher than the original ordered quantity or not negative).   
7. If valid, the system updates the OrderItem in the Order data entity and adjusts the Inventory data entity to reflect the change.   
8. If removing an item, the system updates the OrderItem data entity by removing the selected item and restores the inventory quantity.   
9. If updating the status of the OrderItem, the system modifies the status (e.g., "Shipped," "Processing," "Cancelled") and saves the change.   
10. The system logs the OrderItem management event in the Purchase History data entity.   
11. The system displays a confirmation message to the administrator indicating the OrderItem was successfully managed.   
  
Alternative Flow:   
1. If the administrator enters an invalid quantity in step 6 (e.g., negative number or non-numeric input), the system displays an error message and prompts for correction.   
2. If the system fails to update the OrderItem or Inventory data due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
3. If the administrator cancels the OrderItem management process at any point, the system reverts to the order view without making any changes.   
4. If the OrderItem status is changed to "Cancelled," the system automatically restores the inventory quantity and prevents further processing of the item.   
5. If the system detects a conflict with the updated OrderItem (e.g., insufficient stock for an increased quantity), an error message is displayed, and the administrator is prompted to resolve the issue before proceeding.  
  
Use Case Name: Manage EmailRecord   
Use Case ID: UC-24   
Actors: Administrator, EmailRecord, Email, Customer, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage email records.   
3. The EmailRecord data entity exists and contains at least one record.   
4. The system interface includes a section for viewing and managing email records.   
5. The email service is functioning correctly to retrieve or send emails if necessary.   
  
Postconditions:   
1. The EmailRecord data entity is updated with new, modified, or deleted email records.   
2. The system displays a confirmation message for the EmailRecord management action.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated EmailRecord data through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Email Record Management" section via the system interface.   
2. The system displays a list of existing email records from the EmailRecord data entity, including details such as email subject, recipient, status, and timestamp.   
3. The administrator selects an email record and chooses an action (e.g., view details, edit, delete, or send again).   
4. If editing, the administrator modifies the email details (e.g., recipient, content, or status) and clicks the "Save Changes" button.   
5. If deleting, the system prompts the administrator to confirm the deletion, and the administrator confirms.   
6. If sending again, the system verifies the recipient's email address from the Personal Information or Contact Information data entities and sends the email.   
7. The system updates the EmailRecord data entity with the changes made (e.g., edited, deleted, or resent).   
8. The system logs the management action in the Purchase History data entity.   
9. The system displays a confirmation message to the administrator indicating the EmailRecord was successfully managed.   
  
Alternative Flow:   
1. If the selected email record is not found in the EmailRecord data entity, the system displays an error message and returns to the email record list.   
2. If the administrator attempts to edit an email record but enters invalid or incomplete information (e.g., missing subject or recipient), the system displays an error message and prompts the administrator to correct the data.   
3. If the administrator cancels the management action at any step, the system reverts to the email record list without making changes.   
4. If the system fails to update or delete the EmailRecord data due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the email resend fails due to an invalid recipient email address or email service error, the system logs the failure and displays an error message, prompting the administrator to correct the recipient details or investigate the issue.  
  
Use Case Name: Manage Customer   
Use Case ID: UC-25   
Actors: Administrator, Customer, Personal Information, Contact Information, Purchase History, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage customer accounts.   
3. The customer account to be managed exists in the Personal Information data entity.   
4. The system interface includes a section for viewing and modifying customer data.   
5. The administrator has a valid reason to manage customer accounts (e.g., updating information, deactivating account).   
  
Postconditions:   
1. The customer account is updated, deactivated, or modified in the Personal Information and Contact Information data entities.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated customer account details through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Customer Management" section via the system interface.   
2. The system displays a list of existing customer accounts, including details such as name, email, registration date, and account status.   
3. The administrator selects a specific customer account and chooses an action (e.g., view details, edit information, deactivate account).   
4. If editing, the administrator modifies the customer’s Personal Information or Contact Information (e.g., name, email, phone number).   
5. The system validates the updated information (e.g., email format, phone number format).   
6. If validation is successful, the system updates the Personal Information and/or Contact Information data entities with the new data.   
7. If deactivating, the system prompts the administrator to confirm the action, and the administrator confirms.   
8. The system updates the customer’s account status to "Deactivated" in the Personal Information data entity.   
9. The system logs the management action (e.g., edit, deactivate) in the Purchase History data entity.   
10. The system displays a confirmation message to the administrator indicating the customer account was successfully managed.   
  
Alternative Flow:   
1. If the selected customer account is not found in the Personal Information data entity, the system displays an error message and returns to the customer list.   
2. If the administrator attempts to edit a customer account but enters invalid or incomplete information (e.g., missing required fields), the system displays an error message and prompts the administrator to correct the data.   
3. If the administrator cancels the management action at any step, the system reverts to the customer list without making changes.   
4. If the system fails to update the Personal Information or Contact Information data entities due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to deactivate a customer account that has active orders or pending payments, the system displays a warning and prevents deactivation until all related transactions are resolved.   
6. If the system fails to log the management event in the Purchase History data entity, an error message is displayed, and the administrator is notified to ensure proper audit tracking.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-26   
Actors: Administrator, Inventory, Product, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage inventory.   
3. The system contains at least one product in the Product data entity.   
4. The Inventory data entity is available for updating stock levels.   
5. The system interface provides a section for viewing and managing inventory.   
  
Postconditions:   
1. The inventory levels for one or more products are updated in the Inventory data entity.   
2. The system displays a confirmation message for the inventory management action.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated inventory levels through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Inventory Management" section via the system interface.   
2. The system displays a list of products along with their current inventory levels and status.   
3. The administrator selects a product and modifies its inventory quantity (e.g., increase or decrease stock).   
4. The administrator clicks the "Update Inventory" button.   
5. The system validates the updated inventory quantity (e.g., ensuring it is a non-negative number and within acceptable limits).   
6. If validation is successful, the system updates the Inventory data entity with the new quantity.   
7. The system logs the inventory management event in the Purchase History data entity.   
8. The system displays a confirmation message to the administrator indicating the inventory was successfully updated.   
  
Alternative Flow:   
1. If the entered inventory quantity is negative or invalid in step 5, the system displays an error message and prompts the administrator to enter a valid quantity.   
2. If the selected product is not found in the Product data entity, the system displays an error message and prevents the update.   
3. If the administrator cancels the inventory management process at any step, the system reverts to the inventory list without making changes.   
4. If the system fails to update the inventory due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the inventory update leads to a product becoming out of stock, the system updates the product status accordingly in the Product data entity and logs the change.  
  
Use Case Name: Manage Purchase History   
Use Case ID: UC-27   
Actors: Administrator, Customer, Purchase History, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage purchase history.   
3. The Purchase History data entity contains at least one record.   
4. The system interface includes a section for viewing, modifying, or deleting purchase history records.   
5. The administrator has a valid reason to manage purchase history (e.g., audit, correction of errors, or customer support).   
  
Postconditions:   
1. The Purchase History data entity is updated with new, modified, or deleted records.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated purchase history records through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Purchase History Management" section via the system interface.   
2. The system displays a list of purchase history records, including order numbers, customer identifiers, product details, dates, and transaction amounts.   
3. The administrator selects a specific record and chooses an action (e.g., view details, edit, delete, or export).   
4. If editing, the administrator modifies the record details (e.g., transaction amount, product information, or status).   
5. The system validates the updated data (e.g., correct format, valid product references).   
6. If validation is successful, the system updates the Purchase History data entity with the new information.   
7. If deleting, the system prompts the administrator to confirm the deletion, and the administrator confirms.   
8. The system removes the selected record from the Purchase History data entity.   
9. If exporting, the system generates a file containing the selected or all purchase history records in a specified format (e.g., CSV, PDF).   
10. The system logs the management action (e.g., edit, delete, or export) in the Purchase History data entity.   
11. The system displays a confirmation message to the administrator indicating the purchase history was successfully managed.   
  
Alternative Flow:   
1. If the selected purchase history record is not found in the Purchase History data entity, the system displays an error message and returns to the purchase history list.   
2. If the administrator attempts to edit a record but enters invalid or incomplete information (e.g., incorrect date format, missing required fields), the system displays an error message and prompts the administrator to correct the data.   
3. If the administrator cancels the management action at any step, the system reverts to the purchase history list without making changes.   
4. If the system fails to update or delete a record due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to delete a record that is referenced in other data entities (e.g., Order, EmailRecord), the system displays a warning and prevents the deletion until the dependencies are resolved.   
6. If the export file cannot be generated due to system errors, the system logs the failure and displays an error message, prompting the administrator to retry or contact support via the Contact Information data entity.  
  
Use Case Name: View Personal Information   
Use Case ID: UC-28   
Actors: Customer, System Interface Specification, Personal Information   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is already logged in.   
3. The customer has existing personal information stored in the Personal Information data entity.   
4. The system interface provides an option to view personal information (e.g., Account Settings page).   
  
Postconditions:   
1. The customer is able to view their personal information.   
2. The system displays the customer's name, email, and any other relevant personal details.   
3. The viewing event is logged in the Purchase History data entity for audit purposes.   
4. The personal information remains unchanged in the Personal Information data entity.   
  
Main Flow:   
1. The customer logs in and navigates to the "Account Settings" section via the system interface.   
2. The system retrieves the customer's personal information from the Personal Information data entity.   
3. The system displays the personal information in a read-only format on the interface.   
4. The customer reviews the displayed information for accuracy.   
5. The system logs the viewing event in the Purchase History data entity.   
6. The system displays a confirmation message to the customer that their information has been viewed.   
  
Alternative Flow:   
1. If no personal information is found for the customer in step 2, the system displays a message indicating that no personal information is available.   
2. If the system fails to load the personal information due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer's session expires while viewing personal information, they are redirected to the login page.   
4. If the customer is not logged in and attempts to access the personal information page, they are redirected to the login page.  
  
Use Case Name: Manage Administrator   
Use Case ID: UC-29   
Actors: Administrator, System Interface Specification, Personal Information, Contact Information, Purchase History   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage other administrator accounts.   
3. The administrator account to be managed exists in the Personal Information data entity.   
4. The system interface provides a section for viewing and modifying administrator data.   
5. The administrator has a valid reason to manage other administrator accounts (e.g., updating information, deactivating account).   
  
Postconditions:   
1. The administrator account is updated, deactivated, or modified in the Personal Information and Contact Information data entities.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated administrator account details through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Administrator Management" section via the system interface.   
2. The system displays a list of existing administrator accounts, including details such as name, email, login status, and role.   
3. The administrator selects a specific administrator account and chooses an action (e.g., view details, edit information, deactivate account).   
4. If editing, the administrator modifies the selected administrator’s Personal Information or Contact Information (e.g., name, email, phone number).   
5. The system validates the updated information (e.g., email format, phone number format).   
6. If validation is successful, the system updates the Personal Information and/or Contact Information data entities with the new data.   
7. If deactivating, the system prompts the administrator to confirm the action, and the administrator confirms.   
8. The system updates the selected administrator’s account status to "Deactivated" in the Personal Information data entity.   
9. The system logs the management action (e.g., edit, deactivate) in the Purchase History data entity.   
10. The system displays a confirmation message to the administrator indicating the administrator account was successfully managed.   
  
Alternative Flow:   
1. If the selected administrator account is not found in the Personal Information data entity, the system displays an error message and returns to the administrator list.   
2. If the administrator attempts to edit an account but enters invalid or incomplete information (e.g., missing required fields), the system displays an error message and prompts the administrator to correct the data.   
3. If the administrator cancels the management action at any step, the system reverts to the administrator list without making changes.   
4. If the system fails to update the Personal Information or Contact Information data entities due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to deactivate an account that is currently logged in, the system displays a warning and prevents deactivation until the session is terminated.   
6. If the system fails to log the management event in the Purchase History data entity, an error message is displayed, and the administrator is notified to ensure proper audit tracking.  
  
Use Case Name: Manage Payment   
Use Case ID: UC-30   
Actors: Administrator, Payment, Order, Customer, Contact Information, Purchase History   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage payment records.   
3. The Payment data entity contains at least one payment record.   
4. The system interface includes a section for viewing and managing payment data.   
5. The administrator has a valid reason to manage payment records (e.g., audit, update payment status, or resolve discrepancies).   
  
Postconditions:   
1. The Payment data entity is updated with new, modified, or deleted payment records.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated payment records through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Payment Management" section via the system interface.   
2. The system displays a list of payment records, including details such as order number, customer identifier, payment amount, method, and status.   
3. The administrator selects a specific payment record and chooses an action (e.g., view details, edit, delete, or update status).   
4. If editing, the administrator modifies the payment details (e.g., payment amount, method, or status).   
5. The system validates the updated data (e.g., correct payment format, valid payment method, and status).   
6. If validation is successful, the system updates the Payment data entity with the new information.   
7. If updating the payment status, the system modifies the status (e.g., "Completed," "Failed," "Refunded") and synchronizes it with the corresponding Order data entity.   
8. If deleting, the system prompts the administrator to confirm the deletion, and the administrator confirms.   
9. The system removes the selected payment record from the Payment data entity.   
10. The system logs the management action (e.g., edit, delete, or status update) in the Purchase History data entity.   
11. The system displays a confirmation message to the administrator indicating the payment was successfully managed.   
  
Alternative Flow:   
1. If the selected payment record is not found in the Payment data entity, the system displays an error message and returns to the payment list.   
2. If the administrator attempts to edit a payment record but enters invalid or incomplete information (e.g., incorrect payment amount format, invalid method), the system displays an error message and prompts the administrator to correct the data.   
3. If the administrator cancels the management action at any step, the system reverts to the payment list without making changes.   
4. If the system fails to update the Payment data entity due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to delete a payment record that is referenced in other data entities (e.g., Order or EmailRecord), the system displays a warning and prevents the deletion until the dependencies are resolved.   
6. If the system fails to log the management event in the Purchase History data entity, an error message is displayed, and the administrator is notified to ensure proper audit tracking.  
  
Use Case Name: View EmailRecord   
Use Case ID: UC-31   
Actors: Customer, Administrator, EmailRecord, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer or administrator is already logged in.   
3. There is at least one EmailRecord associated with the customer or available for viewing by the administrator.   
4. The system interface includes a section for viewing email records (e.g., "Email History" or "Sent Emails").   
5. The EmailRecord data entity is available for retrieving email-related data.   
  
Postconditions:   
1. The customer or administrator is able to view the details of the EmailRecord.   
2. The system displays the relevant information from the EmailRecord data entity, such as email subject, recipient, timestamp, and status.   
3. The viewing event is logged in the Purchase History data entity for audit purposes.   
4. The EmailRecord data remains unchanged in the database.   
  
Main Flow:   
1. The customer or administrator navigates to the "Email History" section of the system interface.   
2. The system retrieves the list of EmailRecords associated with the customer or available to the administrator.   
3. The system displays the list of EmailRecords, including key details such as email subject, recipient, date sent, and status.   
4. The customer or administrator selects a specific EmailRecord to view more detailed information.   
5. The system opens the selected EmailRecord and displays the full content (e.g., body of the email, attachments if available).   
6. The system logs the viewing event in the Purchase History data entity.   
7. The system displays a confirmation message to the customer or administrator that the EmailRecord was successfully viewed.   
  
Alternative Flow:   
1. If no EmailRecord is found for the customer in step 2, the system displays a message indicating that no email records are available for the customer.   
2. If the system fails to load the EmailRecord due to technical issues, an error message is displayed, and the customer or administrator is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer or administrator's session expires while viewing an EmailRecord, they are redirected to the login page.   
4. If the customer attempts to view an EmailRecord that was not sent to them, the system denies access and displays a message indicating that the record is not available.   
5. If the administrator is viewing an EmailRecord and the record contains sensitive or restricted content, the system filters or masks the content to ensure data privacy and compliance.  
  
Use Case Name: Manage Personal Information   
Use Case ID: UC-32   
Actors: Administrator, Customer, Personal Information, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator or customer.   
2. The administrator is logged in with the appropriate permissions to manage customer personal information.   
3. The customer is logged in with the appropriate permissions to manage their own personal information.   
4. The customer's personal information is stored in the Personal Information data entity.   
5. The system interface includes a section for viewing and managing personal information (e.g., "User Management" for administrator or "Account Settings" for customer).   
  
Postconditions:   
1. The customer's personal information is updated, viewed, or modified in the Personal Information data entity.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator or customer can view the updated personal information through the system interface.   
  
Main Flow:   
1. The administrator or customer navigates to the "User Management" section (for administrator) or the "Account Settings" section (for customer) via the system interface.   
2. The system displays the customer's personal information fields, such as name, email, date of birth, and any additional relevant details.   
3. The administrator or customer selects the option to update or view personal information.   
4. If updating, the administrator or customer modifies the relevant fields and clicks the "Save Changes" button.   
5. The system validates the updated information (e.g., name format, valid email address).   
6. If validation is successful, the system updates the Personal Information data entity with the new data.   
7. The system logs the management event in the Purchase History data entity.   
8. The system displays a success message confirming the personal information has been successfully managed.   
  
Alternative Flow:   
1. If the administrator or customer enters invalid data (e.g., incorrect email format), the system displays an error message and prompts the user to correct it.   
2. If the administrator attempts to manage a customer’s personal information and the selected customer is not found in the Personal Information data entity, the system displays an error message and returns to the user list.   
3. If the customer does not make any changes and clicks "Save Changes," the system displays a message indicating no updates were made.   
4. If the system fails to update the Personal Information data entity due to technical issues, an error message is displayed, and the user is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator tries to manage a customer’s personal information that is currently in use for active orders or email records, the system displays a warning and may prevent the update until dependencies are resolved.   
6. If the administrator or customer attempts to access the personal information of another customer without proper authorization, the system denies access and logs the attempt in the Purchase History data entity for security tracking.  
  
Use Case Name: Manage Contact Information   
Use Case ID: UC-33   
Actors: Customer, Administrator, Contact Information, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer or administrator is logged in with the appropriate permissions to manage contact information.   
3. The customer’s or administrator’s contact information is stored in the Contact Information data entity.   
4. The system interface includes a section for viewing and modifying contact information (e.g., "Account Settings" for customers or "User Management" for administrators).   
5. The system has access to validation rules for contact information (e.g., phone number format, address completeness).   
  
Postconditions:   
1. The contact information is updated, viewed, or modified in the Contact Information data entity.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The updated contact information is accessible for use in other functionalities, such as order shipping or communication.   
  
Main Flow:   
1. The customer or administrator navigates to the "Account Settings" section (for customers) or "User Management" section (for administrators) via the system interface.   
2. The system displays the current contact information, including phone number, shipping address, billing address, and alternate email.   
3. The customer or administrator selects the option to update or view contact information.   
4. If updating, the customer or administrator modifies the relevant fields and clicks the "Save Changes" button.   
5. The system validates the updated information (e.g., phone number format, address completeness).   
6. If validation is successful, the system updates the Contact Information data entity with the new data.   
7. The system logs the management event in the Purchase History data entity.   
8. The system displays a success message confirming the contact information has been successfully managed.   
  
Alternative Flow:   
1. If the customer or administrator enters invalid data (e.g., incorrect phone number format or incomplete address), the system displays an error message and prompts the user to correct it.   
2. If the customer or administrator does not make any changes and clicks "Save Changes," the system displays a message indicating no updates were made.   
3. If the system fails to update the Contact Information data entity due to technical issues, an error message is displayed, and the user is prompted to retry the operation or contact support via the Contact Information data entity.   
4. If the administrator attempts to manage a customer’s contact information and the selected customer is not found in the Personal Information data entity, the system displays an error message and returns to the user list.   
5. If the updated contact information conflicts with existing data (e.g., duplicate alternate email), the system displays a warning and prompts the user to resolve the conflict.   
6. If the user attempts to access the contact information of another user without proper authorization, the system denies access and logs the attempt in the Purchase History data entity for security tracking.  
  
Use Case Name: Manage Category   
Use Case ID: UC-34   
Actors: Administrator, Category, Product, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage product categories.   
3. The Category data entity contains at least one category or is ready for a new category to be added.   
4. The system interface includes a section for managing categories (e.g., "Category Management").   
5. The administrator has a valid reason to manage categories (e.g., adding a new category, updating an existing category, or deleting a category).   
  
Postconditions:   
1. The Category data entity is updated with new, modified, or deleted categories.   
2. The system displays a confirmation message for the management action performed.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated category list and its associations with products through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Category Management" section via the system interface.   
2. The system displays a list of existing categories, including details such as category name, description, and associated products.   
3. The administrator selects an action to manage a category (e.g., add a new category, edit an existing category, or delete a category).   
4. If adding a new category, the administrator fills in the required fields (e.g., category name, description) and clicks the "Add Category" button.   
5. The system validates the new category information (e.g., unique name, valid description).   
6. If validation is successful, the system adds the new category to the Category data entity.   
7. If editing, the administrator modifies the category details and clicks the "Save Changes" button.   
8. The system validates the updated category information (e.g., ensuring the category name remains unique and the description is complete).   
9. If validation is successful, the system updates the Category data entity with the new information.   
10. If deleting, the system prompts the administrator to confirm the deletion, and the administrator confirms.   
11. The system checks if any products are associated with the category and updates their associations accordingly (e.g., reassigning to a default category or removing category association).   
12. The system deletes the category from the Category data entity.   
13. The system logs the management action (e.g., add, edit, delete) in the Purchase History data entity.   
14. The system displays a confirmation message to the administrator indicating the category was successfully managed.   
  
Alternative Flow:   
1. If the entered category name is not unique in step 5 or 8, the system displays an error message and prompts the administrator to provide a different name.   
2. If the system fails to validate the category information due to missing or incorrect data (e.g., empty name or description), an error message is displayed, and the administrator is prompted to correct the input.   
3. If the administrator cancels the management action at any step, the system reverts to the category list without making changes.   
4. If the system fails to update the Category data entity due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to delete a category that is still associated with active products, the system displays a warning and prevents deletion until the associations are resolved or products are reassigned.   
6. If the system fails to log the management event in the Purchase History data entity, an error message is displayed, and the administrator is notified to ensure proper audit tracking.  
  
Use Case Name: View CartItem   
Use Case ID: UC-35   
Actors: Customer, Shopping Cart, CartItem, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer is logged in.   
3. The customer has at least one item in their Shopping Cart data entity.   
4. The system interface provides a section for viewing individual cart items.   
5. The CartItem data entity is available for retrieving details of items in the cart.   
  
Postconditions:   
1. The customer is able to view the details of a specific CartItem.   
2. The system displays the selected CartItem’s information, such as product name, quantity, price, and total item cost.   
3. The viewing event is logged in the Purchase History data entity for audit purposes.   
4. The CartItem data remains unchanged in the Shopping Cart data entity.   
  
Main Flow:   
1. The customer logs in and navigates to the "Shopping Cart" section via the system interface.   
2. The system displays a list of cart items, including product names, quantities, and prices.   
3. The customer selects a specific CartItem to view more details.   
4. The system retrieves the selected CartItem from the Shopping Cart data entity.   
5. The system displays the detailed information for the selected CartItem, such as product description, image, and any applicable discounts.   
6. The system logs the viewing event in the Purchase History data entity.   
7. The system displays a confirmation message to the customer indicating the CartItem details have been viewed.   
  
Alternative Flow:   
1. If no CartItem is selected in step 3, the system displays a message prompting the customer to select an item from the cart.   
2. If the system fails to load the selected CartItem due to technical issues, an error message is displayed, and the customer is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer's session expires while viewing a CartItem, they are redirected to the login page.   
4. If the selected CartItem is not found in the Shopping Cart data entity, the system displays an error message and returns to the cart list.  
  
Use Case Name: Cancel Order   
Use Case ID: UC-36   
Actors: Customer, Administrator, Order, Inventory, Purchase History   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer or administrator is logged in.   
3. The customer has an order in the Order data entity that is eligible for cancellation (e.g., "Pending," "Processing").   
4. The system interface provides a section for viewing and managing orders.   
5. The Inventory data entity is available to update stock levels if needed.   
  
Postconditions:   
1. The selected order is marked as "Cancelled" in the Order data entity.   
2. The inventory levels for the items in the order are updated to reflect the cancellation.   
3. The cancellation event is logged in the Purchase History data entity for audit purposes.   
4. The customer or administrator receives a confirmation message that the order was successfully cancelled.   
  
Main Flow:   
1. The customer or administrator navigates to the "Order Management" or "Purchase History" section via the system interface.   
2. The system displays a list of orders, including order number, status, date, and items.   
3. The customer or administrator selects an order that is eligible for cancellation and clicks the "Cancel Order" button.   
4. The system prompts the customer or administrator to confirm the cancellation.   
5. The customer or administrator confirms the cancellation.   
6. The system updates the Order data entity to mark the selected order as "Cancelled."   
7. The system retrieves the items in the order and updates the Inventory data entity to increase the stock levels accordingly.   
8. The system logs the cancellation event in the Purchase History data entity.   
9. The system displays a confirmation message to the customer or administrator indicating the order was successfully cancelled.   
  
Alternative Flow:   
1. If the selected order is not eligible for cancellation (e.g., already "Shipped" or "Completed"), the system displays an error message and prevents the cancellation.   
2. If the customer or administrator cancels the cancellation request in step 4, the system reverts to the order list without making any changes.   
3. If the system fails to update the Order or Inventory data entities due to technical issues, an error message is displayed, and the user is prompted to retry the operation or contact support via the Contact Information data entity.   
4. If the customer attempts to cancel an order that does not belong to them, the system denies access and logs the attempt in the Purchase History data entity for security tracking.   
5. If the system fails to log the cancellation event in the Purchase History data entity, an error message is displayed, and the user is notified to ensure proper audit tracking.  
  
Use Case Name: View Payment   
Use Case ID: UC-37   
Actors: Customer, Administrator, Payment, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the user.   
2. The customer or administrator is already logged in.   
3. There is at least one Payment record associated with the customer or available for viewing by the administrator.   
4. The system interface includes a section for viewing payment records (e.g., "Payment History" or "Transaction Details").   
5. The Payment data entity is available for retrieving payment-related data.   
  
Postconditions:   
1. The customer or administrator is able to view the details of the Payment record.   
2. The system displays relevant information from the Payment data entity, such as payment amount, method, date, and status.   
3. The viewing event is logged in the Purchase History data entity for audit purposes.   
4. The Payment data remains unchanged in the database.   
  
Main Flow:   
1. The customer or administrator navigates to the "Payment History" section of the system interface.   
2. The system retrieves the list of Payment records associated with the customer or available to the administrator.   
3. The system displays the list of Payment records, including key details such as payment amount, method, date, and status.   
4. The customer or administrator selects a specific Payment record to view more detailed information.   
5. The system opens the selected Payment record and displays the full content (e.g., transaction ID, payment summary, and confirmation status).   
6. The system logs the viewing event in the Purchase History data entity.   
7. The system displays a confirmation message to the customer or administrator that the Payment record was successfully viewed.   
  
Alternative Flow:   
1. If no Payment record is found for the customer in step 2, the system displays a message indicating that no payment records are available for the customer.   
2. If the system fails to load the Payment record due to technical issues, an error message is displayed, and the customer or administrator is prompted to try again or contact support via the Contact Information data entity.   
3. If the customer or administrator's session expires while viewing a Payment record, they are redirected to the login page.   
4. If the customer attempts to view a Payment record that is not associated with them, the system denies access and displays a message indicating that the record is not available.   
5. If the administrator is viewing a Payment record and the record contains sensitive or restricted content, the system filters or masks the content to ensure data privacy and compliance.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-38   
Actors: Administrator, Inventory, Product, System Interface Specification   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to manage inventory.   
3. The system contains at least one product in the Product data entity.   
4. The Inventory data entity is available for updating stock levels.   
5. The system interface provides a section for viewing and managing inventory.   
  
Postconditions:   
1. The inventory levels for one or more products are updated in the Inventory data entity.   
2. The system displays a confirmation message for the inventory management action.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator can view the updated inventory levels through the system interface.   
  
Main Flow:   
1. The administrator navigates to the "Inventory Management" section via the system interface.   
2. The system displays a list of products along with their current inventory levels and status.   
3. The administrator selects a product and modifies its inventory quantity (e.g., increase or decrease stock).   
4. The administrator clicks the "Update Inventory" button.   
5. The system validates the updated inventory quantity (e.g., ensuring it is a non-negative number and within acceptable limits).   
6. If validation is successful, the system updates the Inventory data entity with the new quantity.   
7. The system logs the inventory management event in the Purchase History data entity.   
8. The system displays a confirmation message to the administrator indicating the inventory was successfully updated.   
  
Alternative Flow:   
1. If the entered inventory quantity is negative or invalid in step 5, the system displays an error message and prompts the administrator to enter a valid quantity.   
2. If the selected product is not found in the Product data entity, the system displays an error message and prevents the update.   
3. If the administrator cancels the inventory management process at any step, the system reverts to the inventory list without making changes.   
4. If the system fails to update the inventory due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the inventory update leads to a product becoming out of stock, the system updates the product status accordingly in the Product data entity and logs the change.  
  
Use Case Name: Manage User Interface Specification   
Use Case ID: UC-39   
Actors: Administrator, User Interface Specification, Product, Category, Shopping Cart, Order, System Interface   
  
Preconditions:   
1. The system is accessible to the administrator.   
2. The administrator is logged in with the appropriate permissions to modify or review the User Interface Specification.   
3. The User Interface Specification data entity contains the current interface design and layout rules.   
4. The system interface includes a dedicated section for managing UI specifications.   
5. The administrator has a valid reason to manage the UI specification (e.g., updating layout for new features, improving accessibility, or aligning with new branding guidelines).   
  
Postconditions:   
1. The User Interface Specification data entity is updated with new or modified rules and settings.   
2. The system interface reflects the updated UI specification as soon as it is applied.   
3. The management event is logged in the Purchase History data entity for audit purposes.   
4. The administrator receives a confirmation message that the UI specification has been successfully managed.   
  
Main Flow:   
1. The administrator navigates to the "UI Specification Management" section via the system interface.   
2. The system displays the current User Interface Specification, including layout rules, form configurations, navigation settings, and visual styles.   
3. The administrator selects a specific UI rule or section to modify (e.g., form fields on the registration page, cart layout, or email formatting).   
4. The administrator updates the selected specification (e.g., adds new fields, removes existing ones, or adjusts styles).   
5. The system validates the updated specification (e.g., ensuring no conflicting layout rules, valid syntax for styling).   
6. If validation is successful, the system updates the User Interface Specification data entity with the new configuration.   
7. The system applies the updated specification to the relevant interface components.   
8. The system logs the management action (e.g., edit, add, or delete) in the Purchase History data entity.   
9. The system displays a confirmation message to the administrator indicating the User Interface Specification was successfully managed.   
  
Alternative Flow:   
1. If the administrator attempts to modify a specification that conflicts with existing system rules, the system displays a warning and prompts the administrator to resolve the conflict before proceeding.   
2. If the system fails to validate the updated specification in step 5 (e.g., invalid syntax or missing required fields), the system displays an error message and prompts the administrator to correct the input.   
3. If the administrator cancels the modification process at any step, the system reverts to the UI specification list without making changes.   
4. If the system fails to update the User Interface Specification data entity due to technical issues, an error message is displayed, and the administrator is prompted to retry the operation or contact support via the Contact Information data entity.   
5. If the administrator attempts to apply the new specification and the system cannot render the interface correctly, the system reverts to the previous valid configuration and logs the failure in the Purchase History data entity.   
6. If the administrator exports the updated specification for documentation or backup, the system generates and provides the file in a specified format (e.g., JSON, XML, or PDF).