项目文档

# Functional Requirement

1. Functional Requirements   
  
1.1 Product Creation Function   
 Function ID: FR-01   
 Description: Administrators can create new products by entering details such as name, description, price, category, and stock quantity. The system validates the data and uses the Plugin module to process any additional configurations.   
 Input: Product details including name, description, price, category, stock quantity, and other optional configurations.   
 Output: A new Product entity is created in the database with the provided details.   
  
1.2 Product Update Function   
 Function ID: FR-02   
 Description: Administrators can update existing product details such as name, description, price, category, and stock quantity. The system validates the updated data and uses the Plugin module for additional processing.   
 Input: Updated product details including name, description, price, category, stock quantity, and other optional configurations.   
 Output: The Product entity in the database is updated with the new details.   
  
1.3 Product Deletion Function   
 Function ID: FR-03   
 Description: Administrators can delete a product after confirmation. The system uses the Plugin module to perform pre-deletion checks and removes the Product entity from the database.   
 Input: Product ID and confirmation of deletion.   
 Output: The selected Product entity is removed from the database.   
  
1.4 Product Inventory Management Function   
 Function ID: FR-04   
 Description: Administrators can modify the stock quantity of a product. The system validates the new quantity and uses the Plugin module for inventory-related checks or actions.   
 Input: Product ID and new stock quantity.   
 Output: The Product entity's StockQuantity attribute is updated in the database.   
  
1.5 Product Detail Viewing Function   
 Function ID: FR-05   
 Description: Customers and administrators can view detailed information about a product, such as name, description, price, category, and availability.   
 Input: Product ID or selection from the product catalog.   
 Output: Displayed product details on a dedicated page.   
  
1.6 Customer Registration Function   
 Function ID: FR-06   
 Description: Administrators can register new customers by entering their details such as name, email, password, address, and contact information. The system validates the data and uses the Plugin module for additional checks.   
 Input: Customer details including name, email, password, address, and contact information.   
 Output: A new Customer entity is created in the database and a confirmation email is sent.   
  
1.7 Customer Profile Viewing Function   
 Function ID: FR-07   
 Description: Administrators can view a customer's profile, including name, email, address, and order history.   
 Input: Customer ID or selection from the customer list.   
 Output: Displayed customer profile details.   
  
1.8 Customer Information Update Function   
 Function ID: FR-08   
 Description: Administrators can update a customer's information, such as name, email, address, or contact details. The system validates the updated data and uses the Plugin module for additional checks.   
 Input: Updated customer details including name, email, address, and contact information.   
 Output: The Customer entity is updated in the database.   
  
1.9 Order Placement Function   
 Function ID: FR-09   
 Description: Customers can place an order by confirming their cart selection, shipping and billing information, and payment method. The system creates an Order entity, processes the Payment, and sends an order confirmation Email.   
 Input: Cart items, shipping and billing information, and selected payment method.   
 Output: A new Order entity is created, the Payment is processed, and a confirmation Email is sent.   
  
1.10 Order Cancellation Function   
 Function ID: FR-10   
 Description: Customers or administrators can cancel an order if it has not been shipped. The system updates the Order status, restores inventory levels, and sends a cancellation confirmation Email.   
 Input: Order ID and confirmation of cancellation.   
 Output: The Order status is updated to "Canceled," inventory levels are restored, and a confirmation Email is sent.   
  
1.11 Order Tracking Function   
 Function ID: FR-11   
 Description: Customers and administrators can track the status of an order. The system retrieves the Order entity and may use the Plugin module to verify or fetch the latest status.   
 Input: Order ID or selection from the order list.   
 Output: Displayed order status, shipping information, and estimated delivery time.   
  
1.12 Order History Management Function   
 Function ID: FR-12   
 Description: Administrators and customers can view and manage order history, including order status updates and cancellations.   
 Input: Order ID or selection from the order history list.   
 Output: Updated Order status or cancellation confirmation.   
  
1.13 Order Status Management Function   
 Function ID: FR-13   
 Description: Administrators can update an order’s status (e.g., Processing, Shipped, Delivered). The system validates the status transition and uses the Plugin module for any pre-status change actions.   
 Input: Order ID and new status.   
 Output: The OrderStatus entity is updated in the database, and any related actions (e.g., Email notification) are triggered.   
  
1.14 Payment Processing Function   
 Function ID: FR-14   
 Description: The system processes customer payments by validating the Order, using the Plugin module to execute the payment, and updating the Payment and Order status accordingly.   
 Input: Order ID and selected payment method.   
 Output: A new Payment entity is created and the Order status is updated.   
  
1.15 Payment Verification Function   
 Function ID: FR-15   
 Description: Administrators can verify the status of a payment. The system retrieves the Payment entity and uses the Plugin module to confirm the transaction with the payment gateway.   
 Input: Payment ID or selection from the payment list.   
 Output: The Payment status is updated and the verification result is displayed.   
  
1.16 Payment Refund Function   
 Function ID: FR-16   
 Description: Administrators can process a refund for a completed payment. The system validates the Order status, uses the Plugin module to execute the refund, updates the Payment and Order status, and sends a refund confirmation Email.   
 Input: Payment ID and confirmation of refund.   
 Output: The Payment status is updated to "Refunded," inventory levels are restored, and a confirmation Email is sent.   
  
1.17 Payment Detail Viewing Function   
 Function ID: FR-17   
 Description: Administrators and customers can view detailed information about a payment, including transaction ID, amount, date, and associated order.   
 Input: Payment ID or selection from the payment history list.   
 Output: Displayed payment details.   
  
1.18 Email Notification Function   
 Function ID: FR-18   
 Description: The system sends Email notifications to customers for order confirmation, status updates, and other system events. The system uses the Email module and may involve the Plugin module for post-notification processing.   
 Input: Order ID, Email template, and customer contact information.   
 Output: A confirmation or update Email is sent to the customer and logged in the Email entity.   
  
1.19 Email Template Management Function   
 Function ID: FR-19   
 Description: Administrators can create, update, or delete Email templates for notifications. The system uses the Plugin module for any pre-processing or validation.   
 Input: Email template details including subject, content, and placeholders.   
 Output: The EmailTemplate entity is created, updated, or deleted in the database.   
  
1.20 Email Settings Management Function   
 Function ID: FR-20   
 Description: Administrators can configure and update the system’s Email settings, such as SMTP server, port, sender address, and authentication details.   
 Input: Updated Email settings (e.g., server address, port, sender email).   
 Output: The Email settings are updated in the configuration database.   
  
1.21 Plugin Installation Function   
 Function ID: FR-21   
 Description: Administrators can install new plugins from a repository or uploaded files. The system validates the plugin for compatibility and security before installation.   
 Input: Plugin file or selection from a repository.   
 Output: A new Plugin entity is created in the database and the plugin is activated.   
  
1.22 Plugin Update Function   
 Function ID: FR-22   
 Description: Administrators can update existing plugins with new configurations or versions. The system validates the new plugin version or configuration before applying the update.   
 Input: Plugin ID, new configuration, or uploaded plugin file.   
 Output: The Plugin entity is updated in the database and the plugin is reactivated.   
  
1.23 Plugin Uninstallation Function   
 Function ID: FR-23   
 Description: Administrators can uninstall plugins after confirmation. The system uses the Plugin module to perform pre-uninstallation checks and removes the plugin from the system.   
 Input: Plugin ID and confirmation of uninstallation.   
 Output: The Plugin entity is removed from the database and the plugin is deactivated.   
  
1.24 Plugin Configuration Function   
 Function ID: FR-24   
 Description: Administrators can configure the settings of an installed plugin. The system validates and applies the new configuration to the Plugin entity.   
 Input: Plugin ID and updated configuration parameters.   
 Output: The Plugin entity is updated in the configuration database.   
  
1.25 Plugin Information Viewing Function   
 Function ID: FR-25   
 Description: Administrators can view detailed information about a plugin, including name, version, status, description, and configuration parameters.   
 Input: Plugin ID or selection from the plugin list.   
 Output: Displayed plugin details.   
  
1.26 Product Category Management Function   
 Function ID: FR-26   
 Description: Administrators can create, update, or delete product categories. The system validates the input data and uses the Plugin module for any additional checks.   
 Input: Category details including name, description, and parent category.   
 Output: The ProductCategory entity is created, updated, or deleted in the database.   
  
1.27 Payment Method Management Function   
 Function ID: FR-27   
 Description: Administrators can create, update, or delete payment methods. The system validates the input data and uses the Plugin module for integration checks.   
 Input: Payment method details including name, type, description, and configuration.   
 Output: The PaymentMethod entity is created, updated, or deleted in the database.   
  
1.28 Administrator Login Function   
 Function ID: FR-28   
 Description: Administrators can log into the system using valid credentials. The system validates the credentials and uses the Plugin module for additional security checks.   
 Input: Administrator username and password.   
 Output: Administrator is authenticated and redirected to the dashboard.

# External Description

# 2. External Interfaces  
  
This chapter defines and describes all the external interfaces of the system, including user interfaces, hardware interfaces, software interfaces, and communication interfaces. These interfaces are essential for the system to interact with users, external hardware, other software components, and communication channels.  
  
## 2.1 User Interface Output  
  
The system interacts with users through a series of well-defined user interfaces. These interfaces are used to display information, collect input, and confirm actions.  
  
### 2.1.1 Product Catalog Page   
\*\*Description\*\*: This interface allows customers and administrators to view a list of available products. Users can select a product to view its details.   
\*\*Interaction Method\*\*: Web-based interface with clickable product listings.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Product ID or selection from the product catalog.   
- \*\*Output\*\*: Displayed product details on a dedicated page.   
  
### 2.1.2 Product Management Interface   
\*\*Description\*\*: This interface is used by administrators to create, update, delete, and manage products. It includes fields for product name, description, price, category, and stock quantity.   
\*\*Interaction Method\*\*: Web-based form interface for data entry and modification.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Product details including name, description, price, category, stock quantity, and other optional configurations.   
- \*\*Output\*\*: Confirmation message or error message based on system validation.   
  
### 2.1.3 Customer Management Interface   
\*\*Description\*\*: This interface is used by administrators to register, view, and update customer information. It includes fields for name, email, password, address, and contact information.   
\*\*Interaction Method\*\*: Web-based form interface for customer data entry and modification.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Customer details including name, email, password, address, and contact information.   
- \*\*Output\*\*: Displayed customer profile details or confirmation message.   
  
### 2.1.4 Order Management Interface   
\*\*Description\*\*: This interface is used by both customers and administrators to place, cancel, track, and view order history. It includes order status, shipping information, and estimated delivery time.   
\*\*Interaction Method\*\*: Web-based interface with order list and status update forms.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Order ID or selection from the order list.   
- \*\*Output\*\*: Displayed order status, shipping information, and estimated delivery time, or updated order status.   
  
### 2.1.5 Payment Management Interface   
\*\*Description\*\*: This interface is used by administrators and customers to view payment details and initiate refunds. It includes fields for transaction ID, amount, date, and associated order.   
\*\*Interaction Method\*\*: Web-based interface with payment history and refund initiation forms.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Payment ID or selection from the payment history list.   
- \*\*Output\*\*: Displayed payment details or confirmation of refund processing.   
  
### 2.1.6 Plugin Management Interface   
\*\*Description\*\*: This interface is used by administrators to install, update, uninstall, and configure plugins. It includes plugin details such as name, version, status, description, and configuration parameters.   
\*\*Interaction Method\*\*: Web-based interface with plugin installation, configuration, and management forms.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Plugin file or selection from a repository, plugin ID, new configuration, or confirmation of uninstallation.   
- \*\*Output\*\*: Displayed plugin details or confirmation of installation, update, or uninstallation.   
  
### 2.1.7 Email Settings Interface   
\*\*Description\*\*: This interface is used by administrators to configure and update email settings such as SMTP server, port, sender address, and authentication details.   
\*\*Interaction Method\*\*: Web-based form for configuration entry and modification.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Updated Email settings (e.g., server address, port, sender email).   
- \*\*Output\*\*: Confirmation message or error message based on system validation.   
  
### 2.1.8 Administrator Login Interface   
\*\*Description\*\*: This interface is used by administrators to log in to the system using valid credentials. It includes fields for username and password.   
\*\*Interaction Method\*\*: Web-based login form with authentication and error feedback.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Administrator username and password.   
- \*\*Output\*\*: Administrator is authenticated and redirected to the dashboard, or an authentication error is displayed.   
  
## 2.2 Hardware Interface Output  
  
There are no direct hardware interfaces required for the current functional scope of the system. However, the system may be deployed on servers or other computing hardware, and it is expected to be compatible with standard enterprise-class hardware.  
  
### 2.2.1 Server Infrastructure   
\*\*Description\*\*: The system is expected to be deployed on enterprise-grade servers capable of handling concurrent user requests and database transactions.   
\*\*Interaction Method\*\*: Standard server deployment with no direct user interaction.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: None directly from user.   
- \*\*Output\*\*: None directly to user.   
  
## 2.3 Software Interface Output  
  
The system interacts with various software components, including databases, third-party tools, and internal modules. These interfaces are defined as follows:  
  
### 2.3.1 Database Interface   
\*\*Description\*\*: The system interacts with a relational database to store and retrieve product, customer, order, payment, email, and plugin data.   
\*\*Interaction Method\*\*: SQL queries and ORM-based interactions for data manipulation.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Data such as product details, customer details, order status, payment methods, and plugin configurations.   
- \*\*Output\*\*: Persisted data in the database, including Product, Customer, Order, Payment, EmailTemplate, Plugin, and ProductCategory entities.   
  
### 2.3.2 Plugin Module Interface   
\*\*Description\*\*: The system uses the Plugin module to perform additional processing, validation, and checks for various functions such as product creation, order status updates, and plugin management.   
\*\*Interaction Method\*\*: API-style integration with the Plugin module to execute specific actions.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Plugin file or configuration, or data requiring validation or processing (e.g., product details, order status changes).   
- \*\*Output\*\*: Execution results from the Plugin module, such as validation status, additional processing output, or error messages.   
  
### 2.3.3 Email Module Interface   
\*\*Description\*\*: The system uses the Email module to send notifications to customers. It may also interact with the Plugin module for post-notification processing.   
\*\*Interaction Method\*\*: API-style integration with the Email module to send and log emails.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Email template, customer contact information, and system event data (e.g., order confirmation, refund notification).   
- \*\*Output\*\*: Sent email notifications and logged email records in the Email entity.   
  
### 2.3.4 Payment Gateway Interface   
\*\*Description\*\*: The system interacts with external payment gateways to process, verify, and refund payments.   
\*\*Interaction Method\*\*: API-style integration with supported payment gateways.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Order ID, payment method, and transaction details.   
- \*\*Output\*\*: Payment confirmation, transaction status, or refund processing result.   
  
### 2.3.5 Email Template Management Interface   
\*\*Description\*\*: The system provides an interface to manage email templates, including creation, update, and deletion of templates.   
\*\*Interaction Method\*\*: Internal system module for email template handling.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Email template details including subject, content, and placeholders.   
- \*\*Output\*\*: Created, updated, or deleted EmailTemplate entity in the database.   
  
## 2.4 Communication Interface Output  
  
The system communicates with users and external services through various communication channels, including email and web-based interactions.  
  
### 2.4.1 Email Notification Communication   
\*\*Description\*\*: The system sends email notifications to customers for events such as order confirmation, status updates, and payment refunds.   
\*\*Interaction Method\*\*: SMTP-based email sending, with integration to the Email module and optionally the Plugin module.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Order ID, Email template, and customer contact information.   
- \*\*Output\*\*: A confirmation or update email is sent to the customer and logged in the Email entity.   
  
### 2.4.2 Web-Based Communication   
\*\*Description\*\*: The system communicates with users through web-based interfaces for all administrative and customer-facing functions.   
\*\*Interaction Method\*\*: HTTP/HTTPS-based communication for rendering web pages and handling form submissions.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Web form data, user selections, and navigation actions.   
- \*\*Output\*\*: Rendered web pages, confirmation messages, and error notifications.   
  
### 2.4.3 Plugin Communication   
\*\*Description\*\*: The system communicates with installed plugins to perform pre-processing, validation, and post-processing actions.   
\*\*Interaction Method\*\*: Internal API-style communication with plugin modules.   
\*\*Input/Output\*\*:   
- \*\*Input\*\*: Plugin ID, configuration parameters, or data requiring processing (e.g., product creation, order status change).   
- \*\*Output\*\*: Plugin execution results, including validation status, processing output, or error messages.   
  
---  
  
By defining these external interfaces, the system ensures a clear and consistent method of interaction with users, internal modules, and external services. This clarity helps reduce implementation errors and improves communication between developers and stakeholders.

# Use Case

Use Case Name: Create Product   
Use Case ID: UC-01   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to create a product. The Plugin module is active and properly configured.   
Postconditions: A new product is successfully added to the system and is available for viewing and ordering by customers.   
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Create Product" option.   
3. The system displays a product creation form.   
4. The administrator fills in the product details (e.g., name, description, price, category, and availability).   
5. The administrator submits the form.   
6. The system validates the input data.   
7. The system uses the Plugin module to process additional configurations (if any).   
8. The system creates a new Product entity with the provided details.   
9. The system confirms the successful creation of the product to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the product creation process.  
  
Use Case Name: Update Product   
Use Case ID: UC-02   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to update a product. The product to be updated already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The selected product is successfully updated with the new details and remains available for viewing and ordering by customers.   
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Update Product" option.   
3. The system displays a list of existing products.   
4. The administrator selects the product they wish to update.   
5. The system displays the product details in an editable form.   
6. The administrator modifies the necessary product information (e.g., name, description, price, category, or availability).   
7. The administrator submits the updated form.   
8. The system validates the updated input data.   
9. The system uses the Plugin module to process any additional configurations (if any).   
10. The system updates the Product entity with the new details.   
11. The system confirms the successful update of the product to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the product update process.  
  
Use Case Name: Delete Product   
Use Case ID: UC-03   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to delete a product. The product to be deleted already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The selected product is successfully removed from the system and is no longer available for viewing or ordering by customers.   
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Delete Product" option.   
3. The system displays a list of existing products.   
4. The administrator selects the product they wish to delete.   
5. The system prompts the administrator to confirm the deletion.   
6. The administrator confirms the deletion.   
7. The system uses the Plugin module to perform any pre-deletion checks or actions (if configured).   
8. The system removes the selected Product entity from the database.   
9. The system confirms the successful deletion of the product to the administrator.   
  
Alternative Flow:   
1. If the administrator cancels the deletion, the system returns to the product list without making any changes.   
2. If the Plugin module fails during pre-deletion processing, the system logs the error and notifies the administrator, halting the deletion process.  
  
Use Case Name: View Product Details   
Use Case ID: UC-04   
Actors: Customer, Administrator   
Preconditions: The system contains at least one Product entity. The user (either a customer or administrator) is logged into the system and has access to the product catalog.   
Postconditions: The user is able to view the detailed information of a selected product.   
Main Flow:   
1. The user navigates to the product catalog via the system interface.   
2. The system displays a list of available products.   
3. The user selects a specific product from the list.   
4. The system retrieves the Product entity from the database.   
5. The system displays the product details (e.g., name, description, price, category, and availability) on a dedicated page.   
6. The user reviews the product information.   
  
Alternative Flow:   
1. If the selected product does not exist or has been removed, the system displays a message indicating that the product is unavailable.   
2. If there is an error retrieving the product data, the system logs the error and displays a generic error message to the user.  
  
Use Case Name: Manage Product Inventory   
Use Case ID: UC-05   
Actors: Administrator, Product, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage product inventory. The Product entity exists in the system. The Plugin module is active and properly configured.   
Postconditions: The inventory levels of the selected product are updated in the system, and the changes are reflected for customers and administrators.   
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Manage Product Inventory" option.   
3. The system displays a list of products with their current inventory levels.   
4. The administrator selects the product they wish to manage.   
5. The system displays the inventory management form for the selected Product.   
6. The administrator updates the inventory quantity (e.g., adds or subtracts stock).   
7. The administrator submits the changes.   
8. The system validates the inventory input data.   
9. The system uses the Plugin module to perform any inventory-related checks or actions (if configured).   
10. The system updates the inventory level of the Product entity in the database.   
11. The system confirms the successful inventory update to the administrator.   
  
Alternative Flow:   
1. If the input inventory value is invalid or results in a negative quantity, the system displays an error message and prompts the administrator to correct the data.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the inventory update process.  
  
Use Case Name: Register Customer   
Use Case ID: UC-06   
Actors: Administrator, Customer, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to register a new customer. The system must have access to the Customer entity and the Plugin module is active and properly configured.   
Postconditions: A new Customer entity is successfully created in the system and is ready for interaction, such as placing an Order or receiving an Email notification.   
Main Flow:   
1. The administrator navigates to the customer management section via the system interface.   
2. The administrator selects the "Register Customer" option.   
3. The system displays a customer registration form.   
4. The administrator fills in the customer details (e.g., name, email, password, address, and contact information).   
5. The administrator submits the form.   
6. The system validates the input data (e.g., checks for valid email format and password strength).   
7. The system uses the Plugin module to perform any additional checks or actions (e.g., email verification or spam detection).   
8. The system creates a new Customer entity with the provided details.   
9. The system sends a confirmation Email to the customer.   
10. The system confirms the successful registration of the customer to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the customer registration process.   
3. If the Email notification fails to send, the system logs the error and notifies the administrator, but the customer registration may still proceed.  
  
Use Case Name: View Customer Profile   
Use Case ID: UC-07   
Actors: Administrator   
Preconditions: The administrator is logged into the system and has the necessary permissions to view customer profiles. The system contains at least one Customer entity.   
Postconditions: The administrator is able to view the detailed information of a selected customer.   
Main Flow:   
1. The administrator navigates to the customer management section via the system interface.   
2. The administrator selects the "View Customer Profile" option.   
3. The system displays a list of registered customers.   
4. The administrator selects a specific customer from the list.   
5. The system retrieves the Customer entity from the database.   
6. The system displays the customer's details (e.g., name, email, address, and order history).   
7. The administrator reviews the customer information.   
  
Alternative Flow:   
1. If the selected customer does not exist or has been removed, the system displays a message indicating that the customer profile is unavailable.   
2. If there is an error retrieving the customer data, the system logs the error and displays a generic error message to the administrator.  
  
Use Case Name: Update Customer Information   
Use Case ID: UC-08   
Actors: Administrator, Customer, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to update customer information. The Customer entity to be updated already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The selected customer's information is successfully updated in the system and reflects the new details in the Customer entity.   
Main Flow:   
1. The administrator navigates to the customer management section via the system interface.   
2. The administrator selects the "Update Customer Information" option.   
3. The system displays a list of registered customers.   
4. The administrator selects the customer they wish to update.   
5. The system displays the customer's details in an editable form.   
6. The administrator modifies the necessary information (e.g., name, email, address, or contact details).   
7. The administrator submits the updated form.   
8. The system validates the updated input data (e.g., checks for valid email format).   
9. The system uses the Plugin module to perform any additional checks or actions (e.g., email verification).   
10. The system updates the Customer entity with the new details.   
11. The system confirms the successful update of the customer information to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the update process.   
3. If the updated email is already associated with another Customer entity, the system displays an error message and prevents the update.  
  
Use Case Name: Process Customer Order   
Use Case ID: UC-09   
Actors: Customer, Administrator, Order, Payment, Email, Plugin   
Preconditions: The customer is logged into the system and has selected at least one Product to purchase. The system has access to the Order and Payment entities. The Plugin module is active and properly configured.   
Postconditions: A new Order entity is created and associated with the customer. The Payment is processed, and the customer is notified via Email of the order confirmation. Inventory levels are updated accordingly.   
Main Flow:   
1. The customer navigates to the shopping cart and selects "Proceed to Checkout."   
2. The system displays the order confirmation page with the selected products and total cost.   
3. The customer enters or confirms their shipping and billing information.   
4. The customer selects a payment method and submits the payment details.   
5. The system creates a new Order entity and associates it with the Customer.   
6. The system processes the Payment through the configured payment gateway.   
7. The system uses the Plugin module to perform any additional validations or actions (e.g., fraud detection).   
8. The system updates the inventory levels of the ordered Products.   
9. The system sends a confirmation Email to the customer.   
10. The system confirms the successful processing of the order to the customer.   
  
Alternative Flow:   
1. If the customer cancels the checkout process, the system returns to the shopping cart without creating an order.   
2. If the payment fails, the system displays an error message and prompts the customer to try a different payment method.   
3. If the Plugin module fails during processing, the system logs the error and notifies the customer, halting the order processing.   
4. If the inventory is insufficient for the ordered Products, the system displays an error message and prevents the order from being processed.   
5. If the Email notification fails to send, the system logs the error but continues with the order processing.  
  
Use Case Name: Delete Customer Account   
Use Case ID: UC-10   
Actors: Administrator   
Preconditions: The administrator is logged into the system and has the necessary permissions to delete a customer account. The Customer entity to be deleted already exists in the system.   
Postconditions: The selected customer account is successfully removed from the system and all associated data (e.g., order history and email records) is either deleted or archived, depending on system configuration.   
Main Flow:   
1. The administrator navigates to the customer management section via the system interface.   
2. The administrator selects the "Delete Customer Account" option.   
3. The system displays a list of registered customers.   
4. The administrator selects the customer they wish to delete.   
5. The system prompts the administrator to confirm the deletion.   
6. The administrator confirms the deletion.   
7. The system removes the selected Customer entity from the database.   
8. The system deletes or archives any associated Order and Email records.   
9. The system confirms the successful deletion of the customer account to the administrator.   
  
Alternative Flow:   
1. If the administrator cancels the deletion, the system returns to the customer list without making any changes.   
2. If the deletion involves legal or compliance restrictions, the system logs the attempt and displays a warning message, preventing the deletion.  
  
Use Case Name: Place Order   
Use Case ID: UC-11   
Actors: Customer, Administrator, Order, Payment, Email, Plugin   
Preconditions: The customer is logged into the system and has selected at least one Product to purchase. The system has access to the Order and Payment entities. The Plugin module is active and properly configured.   
Postconditions: A new Order entity is created and associated with the customer. The Payment is processed, and the customer is notified via Email of the order confirmation. Inventory levels are updated accordingly.   
Main Flow:   
1. The customer navigates to the shopping cart and selects "Proceed to Checkout."   
2. The system displays the order confirmation page with the selected products and total cost.   
3. The customer enters or confirms their shipping and billing information.   
4. The customer selects a payment method and submits the payment details.   
5. The system creates a new Order entity and associates it with the Customer.   
6. The system processes the Payment through the configured payment gateway.   
7. The system uses the Plugin module to perform any additional validations or actions (e.g., fraud detection).   
8. The system updates the inventory levels of the ordered Products.   
9. The system sends a confirmation Email to the customer.   
10. The system confirms the successful processing of the order to the customer.   
  
Alternative Flow:   
1. If the customer cancels the checkout process, the system returns to the shopping cart without creating an order.   
2. If the payment fails, the system displays an error message and prompts the customer to try a different payment method.   
3. If the Plugin module fails during processing, the system logs the error and notifies the customer, halting the order processing.   
4. If the inventory is insufficient for the ordered Products, the system displays an error message and prevents the order from being processed.   
5. If the Email notification fails to send, the system logs the error but continues with the order processing.  
  
Use Case Name: Confirm Order   
Use Case ID: UC-12   
Actors: Customer, Administrator, Order, Payment, Email, Plugin   
Preconditions: The customer has placed an Order and completed the Payment process. The system has access to the Order, Payment, and Plugin entities. The Plugin module is active and properly configured.   
Postconditions: The Order is confirmed by the system. The customer is notified via Email of the confirmation. Inventory levels are updated, and the order status is marked as confirmed.   
Main Flow:   
1. The system receives a completed Order with a successful Payment.   
2. The system checks the availability of the ordered Products.   
3. The system uses the Plugin module to validate the order and payment (if configured).   
4. The system updates the inventory levels of the ordered Products.   
5. The system marks the Order as confirmed.   
6. The system sends a confirmation Email to the customer.   
7. The system updates the order status in the database and displays a confirmation message to the customer.   
  
Alternative Flow:   
1. If the Plugin module fails during validation, the system logs the error and displays a message to the customer, halting the confirmation process.   
2. If the inventory is insufficient for the ordered Products, the system displays an error message, cancels the order, and prompts the customer for alternative options.   
3. If the Email notification fails to send, the system logs the error but continues with the order confirmation process.  
  
Use Case Name: Cancel Order   
Use Case ID: UC-13   
Actors: Customer, Administrator, Order, Plugin   
Preconditions: The customer or administrator is logged into the system and has the necessary permissions to cancel an order. The Order entity to be canceled already exists in the system and has not been shipped. The Plugin module is active and properly configured.   
Postconditions: The selected Order is successfully canceled in the system. The inventory levels of the ordered Products are updated, and the customer is notified via Email of the cancellation.   
Main Flow:   
1. The customer or administrator navigates to the order management section via the system interface.   
2. The system displays a list of existing orders.   
3. The user selects the order they wish to cancel.   
4. The system displays the order details and prompts the user to confirm the cancellation.   
5. The user confirms the cancellation.   
6. The system checks if the order is eligible for cancellation (e.g., not yet shipped).   
7. The system uses the Plugin module to perform any pre-cancellation checks or actions (if configured).   
8. The system updates the Order entity status to "Canceled."   
9. The system restores the inventory levels of the ordered Products.   
10. The system sends a cancellation confirmation Email to the customer.   
11. The system confirms the successful cancellation of the order to the user.   
  
Alternative Flow:   
1. If the user cancels the cancellation request, the system returns to the order list without making any changes.   
2. If the order has already been shipped, the system displays an error message and prevents the cancellation.   
3. If the Plugin module fails during processing, the system logs the error and notifies the user, halting the cancellation process.   
4. If the Email notification fails to send, the system logs the error but continues with the order cancellation process.  
  
Use Case Name: Track Order Status   
Use Case ID: UC-14   
Actors: Customer, Administrator, Order, Plugin   
Preconditions: The customer or administrator is logged into the system and has access to the order management section. The Order entity to be tracked already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The user is able to view the current status of the selected order. The system displays the relevant information, and any additional checks via the Plugin module are completed.   
  
Main Flow:   
1. The customer or administrator navigates to the order management section via the system interface.   
2. The system displays a list of orders associated with the user.   
3. The user selects a specific order from the list.   
4. The system retrieves the Order entity from the database.   
5. The system uses the Plugin module to fetch or verify the latest status information (if configured).   
6. The system displays the order details, including status, shipping information, and estimated delivery time.   
7. The user reviews the order status information.   
  
Alternative Flow:   
1. If the selected order does not exist or has been removed, the system displays a message indicating that the order is unavailable.   
2. If there is an error retrieving the order data, the system logs the error and displays a generic error message to the user.   
3. If the Plugin module fails to retrieve or verify the status, the system logs the error and displays a message to the user, showing the last known status of the order.  
  
Use Case Name: Manage Order History   
Use Case ID: UC-15   
Actors: Administrator, Customer, Order, Plugin   
Preconditions: The user (either an administrator or customer) is logged into the system and has the necessary permissions to view order history. The system contains at least one Order entity. The Plugin module is active and properly configured.   
Postconditions: The user is able to view and manage the history of their or a customer's orders. The system displays the relevant order details, and any additional checks via the Plugin module are completed.   
  
Main Flow:   
1. The user navigates to the order history section via the system interface.   
2. The system displays a list of past orders associated with the user or customer.   
3. The user selects a specific order to view or manage.   
4. The system retrieves the Order entity from the database.   
5. The system uses the Plugin module to fetch or verify additional order-related information (if configured).   
6. The system displays the order details (e.g., items, total cost, order status, and date).   
7. The user reviews the order information or performs actions such as cancellation or refund.   
8. The system updates the Order status accordingly if an action is taken.   
9. The system confirms the action to the user and logs the change.   
  
Alternative Flow:   
1. If the selected order does not exist or has been removed, the system displays a message indicating that the order is unavailable.   
2. If there is an error retrieving the order data, the system logs the error and displays a generic error message to the user.   
3. If the Plugin module fails during processing, the system logs the error and notifies the user, halting the action if applicable.   
4. If the user attempts to cancel or modify an order that is no longer eligible for such actions (e.g., already shipped), the system displays an error message and prevents the change.  
  
Use Case Name: Process Payment   
Use Case ID: UC-16   
Actors: Customer, Administrator, Payment, Plugin   
Preconditions: The customer has placed an Order and selected a payment method. The Payment module is active and properly configured. The Plugin module is active and properly configured.   
Postconditions: The Payment is successfully processed and associated with the Order. The system updates the order status and notifies the customer via Email. Inventory levels are adjusted accordingly.   
  
Main Flow:   
1. The customer or administrator initiates the payment process for an Order.   
2. The system validates the Order status to ensure it is eligible for payment processing.   
3. The system retrieves the Payment configuration and prompts the user to confirm the payment amount.   
4. The system uses the Plugin module to execute payment processing (e.g., payment gateway integration).   
5. The Plugin module returns a success status for the payment.   
6. The system updates the Payment entity with the transaction details.   
7. The system updates the Order status to "Paid."   
8. The system sends a payment confirmation Email to the customer.   
9. The system confirms the successful processing of the payment to the user.   
  
Alternative Flow:   
1. If the Order is not eligible for payment processing (e.g., already paid or canceled), the system displays an error message and prevents the action.   
2. If the Plugin module fails during payment processing, the system logs the error and displays a message to the user, halting the process.   
3. If the Payment confirmation fails to send via Email, the system logs the error but continues with the payment processing.   
4. If the payment is declined or fails, the system displays an error message and prompts the user to try a different payment method.  
  
Use Case Name: Verify Payment   
Use Case ID: UC-17   
Actors: Administrator, Payment, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to verify payments. The Payment entity to be verified already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The system confirms the validity of the Payment. The Payment status is updated accordingly, and the administrator is notified of the verification outcome.   
  
Main Flow:   
1. The administrator navigates to the payment management section via the system interface.   
2. The administrator selects the "Verify Payment" option.   
3. The system displays a list of pending or unverified Payment records.   
4. The administrator selects a specific Payment to verify.   
5. The system retrieves the Payment entity and its associated Order details.   
6. The system uses the Plugin module to validate the payment (e.g., checks with external payment gateways for confirmation).   
7. The Plugin module returns a verification status (e.g., valid, invalid, or failed).   
8. The system updates the Payment entity with the verification result.   
9. The system confirms the verification outcome to the administrator (e.g., success or failure).   
  
Alternative Flow:   
1. If the selected Payment does not exist or has already been verified, the system displays a message indicating the action is unnecessary.   
2. If the Plugin module fails during verification, the system logs the error and displays a message to the administrator, halting the verification process.   
3. If the verification result is invalid, the system updates the Payment status and prompts the administrator to take further action (e.g., investigate or contact the customer).  
  
Use Case Name: Refund Payment   
Use Case ID: UC-18   
Actors: Administrator, Customer, Payment, Plugin, Email   
Preconditions: The administrator is logged into the system and has the necessary permissions to process refunds. The Payment entity associated with an Order is in a "Paid" or "Processed" status. The Plugin module is active and properly configured. The customer has initiated or requested a refund.   
Postconditions: The Payment is successfully refunded, and the system updates the Order and Payment status. The customer is notified via Email of the refund confirmation. Inventory levels are restored if applicable.   
  
Main Flow:   
1. The customer or administrator initiates a refund request for a specific Order.   
2. The system displays the Order and associated Payment details for review.   
3. The system checks if the Order is eligible for a refund (e.g., not yet shipped).   
4. The system uses the Plugin module to process the refund through the configured payment gateway.   
5. The Plugin module returns a success status for the refund.   
6. The system updates the Payment entity to "Refunded" and adjusts the Order status accordingly.   
7. The system restores the inventory levels of the refunded Products.   
8. The system sends a refund confirmation Email to the customer.   
9. The system confirms the successful refund to the administrator or customer.   
  
Alternative Flow:   
1. If the Order is not eligible for a refund (e.g., already shipped or canceled), the system displays an error message and prevents the refund.   
2. If the Plugin module fails during refund processing, the system logs the error and displays a message to the administrator or customer, halting the refund process.   
3. If the Email notification fails to send, the system logs the error but continues with the refund processing.   
4. If the refund is declined or cannot be processed, the system updates the Payment status and notifies the administrator to take further action.  
  
Use Case Name: View Payment Details   
Use Case ID: UC-19   
Actors: Administrator, Customer, Payment, Plugin   
Preconditions: The user (either an administrator or customer) is logged into the system and has access to the payment history. The Payment entity to be viewed already exists in the system. The Plugin module is active and properly configured.   
Postconditions: The user is able to view the detailed information of a selected Payment. The system displays the relevant data, and any additional checks via the Plugin module are completed.   
  
Main Flow:   
1. The administrator or customer navigates to the payment history section via the system interface.   
2. The system displays a list of Payment records associated with the user.   
3. The user selects a specific Payment to view.   
4. The system retrieves the Payment entity from the database.   
5. The system uses the Plugin module to fetch or verify additional payment-related information (if configured).   
6. The system displays the Payment details (e.g., transaction ID, amount, date, status, and associated Order).   
7. The user reviews the payment information.   
  
Alternative Flow:   
1. If the selected Payment does not exist or has been removed, the system displays a message indicating the payment is unavailable.   
2. If there is an error retrieving the Payment data, the system logs the error and displays a generic error message to the user.   
3. If the Plugin module fails to retrieve or verify the payment details, the system logs the error and displays the last known Payment information to the user.  
  
Use Case Name: Send Order Confirmation Email   
Use Case ID: UC-20   
Actors: System, Email, Order, Plugin   
Preconditions: The system has access to an Order entity that has been successfully placed and processed. The Email module is active and properly configured. The Plugin module is active and properly configured.   
Postconditions: The customer receives an order confirmation Email. The Email is logged in the system, and the Plugin module has completed any necessary post-notification actions.   
  
Main Flow:   
1. The system triggers the "Send Order Confirmation Email" action after a successful order placement and payment processing.   
2. The system retrieves the Order entity and associated customer details (e.g., email address).   
3. The system generates the content of the confirmation Email, including order summary and expected delivery time.   
4. The system uses the Email module to send the confirmation Email to the customer.   
5. The system logs the Email in the database.   
6. The system uses the Plugin module to perform any post-notification actions (e.g., analytics tracking or CRM integration).   
7. The system confirms that the Email was successfully sent and processed.   
  
Alternative Flow:   
1. If the customer's Email address is invalid or missing, the system logs an error and displays a message to the administrator, halting the Email sending process.   
2. If the Email module fails to send the Email, the system logs the error and displays a message to the administrator, but the order confirmation process may still proceed.   
3. If the Plugin module fails during post-notification processing, the system logs the error and notifies the administrator, but the Email is still sent to the customer.  
  
Use Case Name: Send Order Update Email   
Use Case ID: UC-21   
Actors: System, Email, Order, Plugin   
Preconditions: The system has access to an Order entity that has been updated (e.g., status change or inventory adjustment). The Email module is active and properly configured. The Plugin module is active and properly configured.   
Postconditions: The customer receives an Email notification regarding the order update. The Email is logged in the system, and the Plugin module has completed any necessary post-notification actions.   
  
Main Flow:   
1. The system detects an update to an existing Order (e.g., status change to "Shipped" or "Processing").   
2. The system retrieves the updated Order entity and the associated customer's email address.   
3. The system generates the content of the update Email, including the new status, relevant details, and next steps.   
4. The system uses the Email module to send the update Email to the customer.   
5. The system logs the Email in the database for record-keeping.   
6. The system uses the Plugin module to perform any post-notification actions (e.g., updating CRM status or tracking user engagement).   
7. The system confirms that the Email was successfully sent and processed.   
  
Alternative Flow:   
1. If the customer's Email address is invalid or missing, the system logs an error and displays a message to the administrator, halting the Email sending process.   
2. If the Email module fails to send the Email, the system logs the error and displays a message to the administrator, but the order update process may still proceed.   
3. If the Plugin module fails during post-notification processing, the system logs the error and notifies the administrator, but the Email is still sent to the customer.  
  
Use Case Name: Manage Email Templates   
Use Case ID: UC-22   
Actors: Administrator, Email, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage email templates. The Email module is active and properly configured. The Plugin module is active and properly configured if additional actions are needed.   
Postconditions: The administrator can create, update, or delete email templates. The changes are reflected in the system and may affect future Email notifications sent to customers.   
  
Main Flow:   
1. The administrator navigates to the email management section via the system interface.   
2. The administrator selects the "Manage Email Templates" option.   
3. The system displays a list of existing email templates with their names and purposes.   
4. The administrator selects an action (Create, Update, or Delete) for a specific template.   
5. If creating or updating, the system displays a form for the email template (e.g., subject, content, placeholders).   
6. The administrator fills in or modifies the template details.   
7. The administrator submits the form.   
8. The system validates the input data.   
9. The system uses the Plugin module to perform any pre-processing or custom validation (if configured).   
10. The system saves or updates the Email Template entity in the database.   
11. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the email template list without making any changes.  
  
Use Case Name: Login as Administrator   
Use Case ID: UC-00   
Actors: Administrator, Plugin   
Preconditions: The system is accessible and operational. The Plugin module is active and properly configured. The administrator has valid login credentials (e.g., username and password).   
Postconditions: The administrator is successfully authenticated and logged into the system, gaining access to administrative features and functionalities.   
  
Main Flow:   
1. The administrator accesses the system login interface.   
2. The administrator enters their username and password.   
3. The system validates the credentials against the stored data.   
4. The system uses the Plugin module to perform additional security checks (e.g., two-factor authentication or session tracking).   
5. The Plugin module returns a success status.   
6. The system grants access to the administrator dashboard.   
7. The system logs the login activity.   
8. The administrator is notified of successful login.   
  
Alternative Flow:   
1. If the credentials are invalid or do not match any existing Administrator entity, the system displays an error message and prompts the administrator to try again.   
2. If the Plugin module fails during processing, the system logs the error and displays a message to the administrator, halting the login process.   
3. If the login attempt is flagged for security reasons (e.g., too many failed attempts), the system locks the account and notifies the administrator.  
  
Use Case Name: Manage Product Categories   
Use Case ID: UC-00   
Actors: Administrator, Product, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage product categories. The system contains at least one Product entity. The Plugin module is active and properly configured.   
Postconditions: The product categories are successfully created, updated, or deleted in the system. The changes are reflected in the Product entities and available for selection during product creation or updates.   
  
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Manage Product Categories" option.   
3. The system displays a list of existing product categories.   
4. The administrator selects an action (Create, Update, or Delete) for a category.   
5. If creating or updating, the system displays a form for the category (e.g., category name, description, or parent category).   
6. The administrator fills in or modifies the category details.   
7. The administrator submits the form.   
8. The system validates the input data.   
9. The system uses the Plugin module to perform any additional checks or actions (if configured).   
10. The system creates, updates, or deletes the Product Category entity in the database.   
11. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the category list without making any changes.  
  
Use Case Name: Manage Customer Accounts   
Use Case ID: UC-23   
Actors: Administrator, Customer, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage customer accounts. The system contains at least one Customer entity. The Plugin module is active and properly configured.   
Postconditions: The customer account is successfully created, updated, or deleted in the system. The changes are reflected in the Customer entity and associated data (e.g., orders and emails).   
  
Main Flow:   
1. The administrator navigates to the customer management section via the system interface.   
2. The administrator selects an action (Create, Update, or Delete) for a customer account.   
3. If creating or updating, the system displays a form for the customer account (e.g., name, email, address, or contact information).   
4. The administrator fills in or modifies the customer details as needed.   
5. The administrator submits the form.   
6. The system validates the input data (e.g., checks for valid email format and password strength).   
7. The system uses the Plugin module to perform any additional checks or actions (e.g., email verification or spam detection).   
8. The system creates, updates, or deletes the Customer entity in the database.   
9. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the customer list without making any changes.   
4. If the selected customer does not exist or has already been deleted, the system displays a message indicating that the account is unavailable.  
  
Use Case Name: Manage Orders   
Use Case ID: UC-24   
Actors: Administrator, Customer, Order, Plugin   
Preconditions: The user (either an administrator or customer) is logged into the system and has access to the order management section. The system contains at least one Order entity. The Plugin module is active and properly configured.   
Postconditions: The user is able to view, edit, or delete orders. The system updates the relevant order information, and any additional checks via the Plugin module are completed.   
  
Main Flow:   
1. The administrator or customer navigates to the order management section via the system interface.   
2. The system displays a list of orders associated with the user.   
3. The user selects a specific order to manage.   
4. The system retrieves the Order entity from the database.   
5. The system displays the order details (e.g., items, total cost, order status, and date).   
6. The user can choose to update the order status or cancel the order.   
7. If the user selects to update the status, the system validates the new status and uses the Plugin module to perform any necessary checks.   
8. The system updates the Order entity with the new status or cancellation.   
9. The system confirms the change to the user and logs the activity.   
  
Alternative Flow:   
1. If the selected order does not exist or has been removed, the system displays a message indicating that the order is unavailable.   
2. If there is an error retrieving the order data, the system logs the error and displays a generic error message to the user.   
3. If the Plugin module fails during processing, the system logs the error and notifies the user, halting the action if applicable.   
4. If the user cancels the action, the system returns to the order list without making any changes.   
5. If the order has already been shipped, the system displays an error message and prevents the cancellation.  
  
Use Case Name: Manage Payment Methods   
Use Case ID: UC-25   
Actors: Administrator, Payment, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage payment methods. The Payment module is active and properly configured. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The payment methods are successfully created, updated, or deleted in the system. The changes are reflected in the Payment configuration and are available for use during order processing.   
  
Main Flow:   
1. The administrator navigates to the payment management section via the system interface.   
2. The administrator selects an action (Create, Update, or Delete) for a payment method.   
3. If creating or updating, the system displays a form for the payment method (e.g., name, type, configuration details).   
4. The administrator fills in or modifies the payment method details.   
5. The administrator submits the form.   
6. The system validates the input data (e.g., checks for duplicate payment method names).   
7. The system uses the Plugin module to perform any additional checks or actions (if configured).   
8. The system creates, updates, or deletes the Payment Method entity in the database.   
9. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the payment method list without making any changes.   
4. If the selected payment method does not exist or has already been deleted, the system displays a message indicating that the method is unavailable.  
  
Use Case Name: Manage Email Settings   
Use Case ID: UC-26   
Actors: Administrator, Email, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage email settings. The Email module is active and properly configured. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The email settings are successfully configured or updated in the system. The changes are reflected in the Email module and may affect future Email notifications sent to customers.   
  
Main Flow:   
1. The administrator navigates to the email management section via the system interface.   
2. The administrator selects the "Manage Email Settings" option.   
3. The system displays the current email settings (e.g., SMTP server, port, sender address, and authentication details).   
4. The administrator modifies the necessary settings or adds new configurations.   
5. The administrator submits the updated email settings.   
6. The system validates the input data (e.g., checks for valid server address and port).   
7. The system uses the Plugin module to perform any additional validation or integration checks (if configured).   
8. The system updates the Email settings in the configuration database.   
9. The system confirms the successful update of the email settings to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation, the system logs the error and notifies the administrator, halting the update process.   
3. If the administrator cancels the update, the system returns to the email settings page without making any changes.   
4. If the updated settings prevent the Email module from functioning (e.g., invalid server credentials), the system logs the failure and displays a warning to the administrator.  
  
Use Case Name: Install Plugin   
Use Case ID: UC-27   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to install plugins. The Plugin module is accessible and supports installation of new plugins. The system is in a stable state and ready to accept plugin configurations.   
Postconditions: A new Plugin is successfully installed and activated in the system. The plugin is available for use in enhancing system functionality (e.g., email verification, payment processing, or fraud detection).   
  
Main Flow:   
1. The administrator navigates to the plugin management section via the system interface.   
2. The administrator selects the "Install Plugin" option.   
3. The system displays a plugin repository or allows the administrator to upload a plugin file.   
4. The administrator selects or uploads the Plugin to be installed.   
5. The system validates the plugin file or configuration for compatibility and security.   
6. The system installs the Plugin and configures it based on the provided settings.   
7. The system activates the Plugin and adds it to the available modules.   
8. The system confirms the successful installation of the plugin to the administrator.   
  
Alternative Flow:   
1. If the plugin file is invalid or incompatible, the system displays an error message and prompts the administrator to select a valid plugin.   
2. If the Plugin module fails during installation or activation, the system logs the error and notifies the administrator, halting the installation process.   
3. If the administrator cancels the installation, the system returns to the plugin management page without making any changes.  
  
Use Case Name: Update Plugin   
Use Case ID: UC-28   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to update plugins. The Plugin entity to be updated already exists in the system. The system is in a stable state and ready to accept plugin updates.   
Postconditions: The selected Plugin is successfully updated with new configurations or versions. The updated plugin is active and available for use in enhancing system functionality (e.g., email verification, payment processing, or fraud detection).   
  
Main Flow:   
1. The administrator navigates to the plugin management section via the system interface.   
2. The administrator selects the "Update Plugin" option.   
3. The system displays a list of installed plugins.   
4. The administrator selects the plugin they wish to update.   
5. The system displays the plugin's current configuration and provides an update form.   
6. The administrator modifies the plugin configuration or uploads a new version of the plugin.   
7. The administrator submits the update request.   
8. The system validates the new plugin version or configuration for compatibility and security.   
9. The system deactivates the existing plugin and applies the update.   
10. The system reactivates the updated plugin and confirms the update to the administrator.   
  
Alternative Flow:   
1. If the plugin file or configuration is invalid or incompatible, the system displays an error message and prompts the administrator to provide a valid plugin.   
2. If the Plugin module fails during deactivation or update, the system logs the error and notifies the administrator, halting the update process.   
3. If the administrator cancels the update, the system returns to the plugin management page without making any changes.  
  
Use Case Name: Uninstall Plugin   
Use Case ID: UC-29   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to uninstall plugins. The Plugin to be uninstalled is currently installed and active in the system. The system is in a stable state and supports plugin deactivation and removal.   
Postconditions: The selected Plugin is successfully uninstalled and removed from the system. The system no longer uses the plugin for any functionality, and any associated configurations are either deleted or archived.   
  
Main Flow:   
1. The administrator navigates to the plugin management section via the system interface.   
2. The administrator selects the "Uninstall Plugin" option.   
3. The system displays a list of installed plugins.   
4. The administrator selects the plugin they wish to uninstall.   
5. The system prompts the administrator to confirm the uninstallation.   
6. The administrator confirms the uninstallation.   
7. The system deactivates the Plugin and removes it from the active modules.   
8. The system deletes or archives the plugin's configuration data.   
9. The system confirms the successful uninstallation of the plugin to the administrator.   
  
Alternative Flow:   
1. If the administrator cancels the uninstallation, the system returns to the plugin management page without making any changes.   
2. If the Plugin is currently in use (e.g., required for an ongoing process), the system displays a warning message and prevents the uninstallation.   
3. If the Plugin module fails during deactivation or removal, the system logs the error and notifies the administrator, halting the uninstallation process.  
  
Use Case Name: Configure Plugin Settings   
Use Case ID: UC-30   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to configure plugins. The Plugin to be configured is already installed and active in the system. The system is in a stable state and supports plugin configuration.   
Postconditions: The selected Plugin's settings are successfully updated in the system. The plugin functions according to the new configuration and may affect system behavior (e.g., email verification, payment gateway integration, or inventory validation).   
  
Main Flow:   
1. The administrator navigates to the plugin management section via the system interface.   
2. The administrator selects a specific Plugin to configure.   
3. The system displays the current settings and configuration options for the selected Plugin.   
4. The administrator modifies the plugin settings as needed (e.g., API keys, activation status, or behavior rules).   
5. The administrator submits the updated configuration.   
6. The system validates the input data for the plugin settings.   
7. The system updates the Plugin entity in the configuration database.   
8. The system reactivates the plugin with the new settings.   
9. The system confirms the successful configuration update to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the configuration.   
2. If the Plugin module fails to apply the updated settings, the system logs the error and notifies the administrator, halting the configuration process.   
3. If the administrator cancels the configuration, the system returns to the plugin management page without making any changes.  
  
Use Case Name: View Plugin Information   
Use Case ID: UC-31   
Actors: Administrator, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to view plugin information. The Plugin module is installed and active in the system.   
Postconditions: The administrator is able to view detailed information about a specific Plugin, including its configuration, status, and description.   
  
Main Flow:   
1. The administrator navigates to the plugin management section via the system interface.   
2. The administrator selects the "View Plugin Information" option.   
3. The system displays a list of installed plugins.   
4. The administrator selects a specific Plugin to view.   
5. The system retrieves the Plugin entity from the configuration database.   
6. The system displays the plugin details (e.g., name, version, status, description, and configuration parameters).   
7. The administrator reviews the plugin information.   
  
Alternative Flow:   
1. If the selected Plugin does not exist or has been removed, the system displays a message indicating that the plugin is unavailable.   
2. If there is an error retrieving the Plugin data, the system logs the error and displays a generic error message to the administrator.  
  
Use Case Name: Manage Order Status   
Use Case ID: UC-32   
Actors: Administrator, Order, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage order statuses. The system contains at least one Order entity. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The order status is successfully updated in the system. The changes are reflected in the Order entity and may trigger related actions such as Email notifications or inventory updates, depending on the system configuration.   
  
Main Flow:   
1. The administrator navigates to the order management section via the system interface.   
2. The administrator selects the "Manage Order Status" option.   
3. The system displays a list of orders with their current status.   
4. The administrator selects an order and chooses a new status from available options (e.g., "Processing", "Shipped", "Delivered", "Canceled").   
5. The system validates the selected status change to ensure it is permissible according to business rules.   
6. The system uses the Plugin module to perform any pre-status change checks or actions (if configured).   
7. The system updates the Order Status entity in the database with the new status.   
8. If applicable, the system triggers an Email notification to the customer regarding the status change.   
9. If applicable, the system updates inventory levels based on the new status.   
10. The system confirms the successful status update to the administrator.   
  
Alternative Flow:   
1. If the selected status change is not allowed (e.g., attempting to mark an order as "Delivered" without first marking it as "Shipped"), the system displays an error message and prevents the change.   
2. If the Plugin module fails during pre-status change processing, the system logs the error and notifies the administrator, halting the status update process.   
3. If the Email notification fails to send, the system logs the error but continues with the status update process.   
4. If the inventory update fails (e.g., due to system error), the system logs the error and notifies the administrator, but the order status update may still proceed.   
5. If the administrator cancels the action, the system returns to the order list without making any changes.  
  
Use Case Name: Manage ProductCategory   
Use Case ID: UC-33   
Actors: Administrator, ProductCategory, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage product categories. The system contains at least one ProductCategory entity. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The product category is successfully created, updated, or deleted in the system. The changes are reflected in the ProductCategory entity and may affect product organization and filtering within the system.   
  
Main Flow:   
1. The administrator navigates to the product management section via the system interface.   
2. The administrator selects the "Manage ProductCategory" option.   
3. The system displays a list of existing product categories.   
4. The administrator selects an action (Create, Update, or Delete) for a category.   
5. If creating or updating, the system displays a form for the product category (e.g., name, description, and parent category).   
6. The administrator fills in or modifies the category details.   
7. The administrator submits the form.   
8. The system validates the input data (e.g., checks for duplicate category names).   
9. The system uses the Plugin module to perform any additional checks or actions (if configured).   
10. The system creates, updates, or deletes the ProductCategory entity in the database.   
11. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the category list without making any changes.   
4. If the selected product category does not exist or has already been deleted, the system displays a message indicating that the category is unavailable.  
  
Use Case Name: Manage PaymentMethod   
Use Case ID: UC-34   
Actors: Administrator, PaymentMethod, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage payment methods. The system contains at least one PaymentMethod entity. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The payment method is successfully created, updated, or deleted in the system. The changes are reflected in the PaymentMethod entity and may affect the available payment options during order processing.   
  
Main Flow:   
1. The administrator navigates to the payment management section via the system interface.   
2. The administrator selects the "Manage PaymentMethod" option.   
3. The system displays a list of existing payment methods.   
4. The administrator selects an action (Create, Update, or Delete) for a payment method.   
5. If creating or updating, the system displays a form for the payment method (e.g., name, type, description, and configuration details).   
6. The administrator fills in or modifies the payment method details.   
7. The administrator submits the form.   
8. The system validates the input data (e.g., checks for duplicate method names).   
9. The system uses the Plugin module to perform any additional validation or integration checks (if configured).   
10. The system creates, updates, or deletes the PaymentMethod entity in the database.   
11. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the payment method list without making any changes.   
4. If the selected payment method does not exist or has already been deleted, the system displays a message indicating that the method is unavailable.  
  
Use Case Name: Manage EmailTemplate   
Use Case ID: UC-35   
Actors: Administrator, EmailTemplate, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage email templates. The EmailTemplate entity is supported in the system. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The email template is successfully created, updated, or deleted in the system. The changes are reflected in the EmailTemplate entity and may affect the content and structure of future email notifications sent to customers.   
  
Main Flow:   
1. The administrator navigates to the email management section via the system interface.   
2. The administrator selects the "Manage EmailTemplate" option.   
3. The system displays a list of existing email templates.   
4. The administrator selects an action (Create, Update, or Delete) for an email template.   
5. If creating or updating, the system displays a form for the email template (e.g., subject, template content, placeholders, and associated event).   
6. The administrator fills in or modifies the template details.   
7. The administrator submits the form.   
8. The system validates the input data (e.g., checks for valid placeholders or template syntax).   
9. The system uses the Plugin module to perform any additional validation or custom processing (if configured).   
10. The system creates, updates, or deletes the EmailTemplate entity in the database.   
11. The system confirms the action (creation, update, or deletion) to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation or processing, the system logs the error and notifies the administrator, halting the action.   
3. If the administrator cancels the action, the system returns to the email template list without making any changes.   
4. If the selected email template does not exist or has already been deleted, the system displays a message indicating that the template is unavailable.  
  
Use Case Name: Manage Email Settings   
Use Case ID: UC-36   
Actors: Administrator, Email, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage email settings. The Email module is active and properly configured. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The email settings are successfully configured or updated in the system. The changes are reflected in the Email module and may affect future Email notifications sent to customers.   
  
Main Flow:   
1. The administrator navigates to the email settings management section via the system interface.   
2. The administrator selects the "Manage Email Settings" option.   
3. The system displays the current email settings (e.g., SMTP server, port, sender address, authentication details, and encryption settings).   
4. The administrator modifies the necessary settings or adds new configurations.   
5. The administrator submits the updated email settings.   
6. The system validates the input data (e.g., checks for valid server address, port, and authentication credentials).   
7. The system uses the Plugin module to perform any additional validation or integration checks (if configured).   
8. The system updates the Email settings in the configuration database.   
9. The system confirms the successful update of the email settings to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation, the system logs the error and notifies the administrator, halting the update process.   
3. If the administrator cancels the update, the system returns to the email settings page without making any changes.   
4. If the updated settings prevent the Email module from functioning (e.g., invalid server credentials), the system logs the failure and displays a warning to the administrator.  
  
Use Case Name: Manage Product Categories   
Use Case ID: UC-37   
Actors: Administrator, ProductCategory, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage product categories. The system contains at least one ProductCategory entity. The Plugin module is active and properly configured to support custom validation or processing if required.   
Postconditions: The product categories are successfully created, updated, or deleted in the system. The changes are reflected in the ProductCategory entity and may affect how products are organized and filtered in the product catalog.   
  
Main Flow:   
1. The administrator accesses the product management interface.   
2. The administrator selects the "Manage Product Categories" option.   
3. The system presents a list of existing product categories.   
4. The administrator chooses to either create a new category, update an existing one, or delete a category.   
5. If creating or updating, the system displays a form with fields such as category name, description, and parent category (if applicable).   
6. The administrator fills in the required information and submits the form.   
7. The system validates the input to ensure it meets formatting and logical constraints (e.g., no duplicate names, valid parent-child relationships).   
8. If the Plugin module is configured to perform additional checks (e.g., category usage in orders or inventory), the system invokes the Plugin to validate the action.   
9. The system updates the ProductCategory entity in the database accordingly.   
10. The system confirms the successful creation, update, or deletion of the product category to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation or processing, the system logs the error, displays a message to the administrator, and halts the action.   
3. If the administrator cancels the operation, the system returns to the product category list without applying any changes.   
4. If the selected product category does not exist or has already been deleted, the system displays a message indicating the category is unavailable.  
  
Use Case Name: Manage Payment Methods   
Use Case ID: UC-38   
Actors: Administrator, PaymentMethod, Plugin   
Preconditions: The administrator is logged into the system and has the necessary permissions to manage payment methods. The system contains at least one PaymentMethod entity. The Plugin module is active and properly configured if additional actions are required.   
Postconditions: The payment methods are successfully created, updated, or deleted in the system. The changes are reflected in the PaymentMethod entity and may affect the available payment options for customers during order processing.   
  
Main Flow:   
1. The administrator accesses the payment management interface.   
2. The administrator selects the "Manage Payment Methods" option.   
3. The system displays a list of existing payment methods.   
4. The administrator chooses to either create a new payment method, update an existing one, or delete a payment method.   
5. If creating or updating, the system displays a form with fields such as method name, type (e.g., credit card, PayPal, bank transfer), description, and configuration parameters (e.g., API keys, service endpoints).   
6. The administrator fills in the required information and submits the form.   
7. The system validates the input to ensure it meets system requirements (e.g., unique method name, valid configuration details).   
8. If the Plugin module is configured to perform additional checks (e.g., integration testing or validation of payment gateway credentials), the system invokes the Plugin for verification.   
9. The system updates the PaymentMethod entity in the database accordingly.   
10. The system confirms the successful creation, update, or deletion of the payment method to the administrator.   
  
Alternative Flow:   
1. If the input data is invalid or incomplete, the system displays an error message and prompts the administrator to correct the information.   
2. If the Plugin module fails during validation, the system logs the error, displays a message to the administrator, and halts the action.   
3. If the administrator cancels the operation, the system returns to the payment method list without applying any changes.   
4. If the selected payment method does not exist or has already been deleted, the system displays a message indicating the method is unavailable.