项目文档

# Functional Requirement

# 1. Functional Requirements  
  
1.1 Customer Registration Function   
Function ID: FR-01   
Description: Customers can create a new account by providing personal information. The registration must be approved by an administrator before being activated.   
Input: Customer's personal information (name, email, password)   
Output: New customer profile in the Customer data entity and confirmation email sent to the customer's email address  
  
1.2 Customer Login Function   
Function ID: FR-02   
Description: Authenticated customers can access their account by providing valid login credentials.   
Input: Customer's email and password   
Output: Active session in the Session data entity and redirection to the account dashboard  
  
1.3 Customer Logout Function   
Function ID: FR-03   
Description: Customers can terminate their active session and exit their account.   
Input: Active session in the Session data entity   
Output: Terminated session in the Session data entity and redirection to the login/home page  
  
1.4 View Product Details Function   
Function ID: FR-04   
Description: Customers can view detailed information about a specific product, including inventory status.   
Input: ProductID selected by the customer   
Output: Product details (name, description, price, category, inventory status) displayed to the customer  
  
1.5 Search Products by Category Function   
Function ID: FR-05   
Description: Customers can filter and view products based on selected categories.   
Input: CategoryID selected by the customer   
Output: Filtered list of products belonging to the selected category with their details and inventory status  
  
1.6 Add Product to Cart Function   
Function ID: FR-06   
Description: Customers can add products to their cart for future purchase.   
Input: ProductID and quantity selected by the customer   
Output: Updated cart in the Order data entity and inventory quantity updated in the Inventory data entity  
  
1.7 Checkout Function   
Function ID: FR-07   
Description: Customers can complete the purchase process by confirming their cart, providing shipping/billing information, and processing payment.   
Input: Customer's cart (OrderID), shipping/billing information, and selected payment method   
Output: Finalized order in the Order data entity, processed payment in the Payment data entity, and confirmation email sent to the customer  
  
1.8 Receive Order Confirmation Email Function   
Function ID: FR-08   
Description: Customers receive a confirmation email after completing the checkout process.   
Input: OrderID and customer's email address   
Output: Order confirmation email sent to the customer's email address  
  
1.9 Administrator Login Function   
Function ID: FR-09   
Description: Administrators can access the administrative dashboard by providing valid login credentials.   
Input: Administrator's email and password   
Output: Active session in the Session data entity and redirection to the administrative dashboard  
  
1.10 Administrator Logout Function   
Function ID: FR-10   
Description: Administrators can terminate their active session and exit the administrative dashboard.   
Input: Active session in the Session data entity   
Output: Terminated session in the Session data entity and redirection to the login/home page  
  
1.11 Manage Inventory Function   
Function ID: FR-11   
Description: Administrators can update inventory levels for products.   
Input: ProductID and updated inventory quantity   
Output: Updated inventory levels in the Inventory data entity and product availability status in the Product data entity  
  
1.12 Add Product Function   
Function ID: FR-12   
Description: Administrators can create new product records with associated inventory and category information.   
Input: Product details (name, description, price, CategoryID, InventoryID)   
Output: New product record in the Product data entity with valid inventory and category associations  
  
1.13 Update Product Function   
Function ID: FR-13   
Description: Administrators can modify product details including inventory and category associations.   
Input: ProductID and updated product information (name, description, price, CategoryID, InventoryID)   
Output: Updated product record in the Product data entity with modified inventory and category associations  
  
1.14 Delete Product Function   
Function ID: FR-14   
Description: Administrators can remove product records from the system.   
Input: ProductID to be deleted   
Output: Removed product record from the Product data entity with updated inventory and category associations  
  
1.15 Manage Product Categories Function   
Function ID: FR-15   
Description: Administrators can add, update, or delete product categories.   
Input: Category details (name, description, ParentCategoryID)   
Output: Updated category records in the Category data entity with appropriate product associations  
  
1.16 Add Category Function   
Function ID: FR-16   
Description: Administrators can create new product category records.   
Input: New category details (name, description, ParentCategoryID)   
Output: New category record in the Category data entity  
  
1.17 Update Category Function   
Function ID: FR-17   
Description: Administrators can modify existing product category records.   
Input: CategoryID and updated category information (name, description, ParentCategoryID)   
Output: Updated category record in the Category data entity  
  
1.18 Delete Category Function   
Function ID: FR-18   
Description: Administrators can remove product category records from the system.   
Input: CategoryID to be deleted   
Output: Removed category record from the Category data entity with updated product associations  
  
1.19 View Order Details Function   
Function ID: FR-19   
Description: Customers can view detailed information about their orders, including product and inventory status at the time of purchase.   
Input: OrderID selected by the customer   
Output: Order details (items, quantities, prices, total amount, status) displayed to the customer  
  
1.20 Manage Orders Function   
Function ID: FR-20   
Description: Administrators can view, update, or delete existing orders and their related information.   
Input: OrderID and selected action (view, update, or delete)   
Output: Updated order status and related information in the Order data entity  
  
1.21 Process Payment Function   
Function ID: FR-21   
Description: The system can handle the payment process for customer orders.   
Input: OrderID, payment method, and payment details   
Output: Payment record in the Payment data entity and updated order status in the Order data entity  
  
1.22 Manage Payment Information Function   
Function ID: FR-22   
Description: Customers can add, update, or delete saved payment methods.   
Input: Payment information (method, card details, billing address)   
Output: Updated payment records in the Payment data entity  
  
1.23 Manage Sessions Function   
Function ID: FR-23   
Description: Administrators can manage user sessions (create, update, or delete).   
Input: SessionID and selected action (create, update, or delete)   
Output: Updated session records in the Session data entity  
  
1.24 Install Plugin Function   
Function ID: FR-24   
Description: Administrators can install new plugins to extend system functionality.   
Input: Plugin file or configuration details   
Output: Installed plugin record in the Plugin data entity  
  
1.25 Uninstall Plugin Function   
Function ID: FR-25   
Description: Administrators can remove plugins from the system.   
Input: PluginID to be uninstalled   
Output: Removed plugin record from the Plugin data entity  
  
1.26 Update Plugin Function   
Function ID: FR-26   
Description: Administrators can update existing plugins with new versions or configurations.   
Input: PluginID and updated plugin file or configuration details   
Output: Updated plugin record in the Plugin data entity  
  
1.27 Configure Plugin Function   
Function ID: FR-27   
Description: Administrators can adjust plugin settings to customize functionality.   
Input: PluginID and new configuration settings   
Output: Updated plugin configuration in the Plugin data entity  
  
1.28 View Plugin Documentation Function   
Function ID: FR-28   
Description: Administrators can access plugin documentation for usage and configuration guidance.   
Input: PluginID selected by the administrator   
Output: Plugin documentation displayed to the administrator  
  
1.29 Manage Email Templates Function   
Function ID: FR-29   
Description: Administrators can add, update, or delete email templates used for notifications.   
Input: Email template details (subject, body, recipient type)   
Output: Updated email template records in the EmailTemplate data entity  
  
1.30 Send Order Confirmation Email Function   
Function ID: FR-30   
Description: The system can send confirmation emails to customers after order processing.   
Input: OrderID and customer's email address   
Output: Order confirmation email sent to the customer's email address

# External Description

2. External Interfaces  
  
2.1 User Interface Output   
The user interface of the system is designed to provide an intuitive and user-friendly experience for both customers and administrators. Key components include:   
  
- \*\*Customer Account Dashboard\*\*: Displays personalized information for the logged-in customer, including order history, saved payment methods, and profile details.   
- \*\*Product Detail Page\*\*: Shows comprehensive details about a specific product, including name, description, price, category, and inventory status.   
- \*\*Product Search Interface\*\*: Allows customers to filter products by category, displaying a list of products with their details and inventory status.   
- \*\*Shopping Cart Interface\*\*: Enables customers to view and manage their selected products for purchase, including quantities and total price.   
- \*\*Checkout Interface\*\*: Provides a step-by-step process for customers to confirm their cart, enter shipping/billing information, and select a payment method.   
- \*\*Administrative Dashboard\*\*: Offers administrators tools to manage products, inventory, categories, orders, sessions, and plugins, along with email templates and system settings.   
- \*\*Session Management Interface\*\*: Allows users to log in and log out, with clear prompts and confirmation messages for session termination.   
- \*\*Order Details Interface\*\*: Displays detailed information about a specific order, including items, quantities, prices, total amount, and status.   
  
Each interface is designed to provide clear visual feedback and interactive elements to ensure seamless user interaction and accurate data input/output.  
  
2.2 Hardware Interface Output   
The system does not directly interact with external hardware devices. However, it is designed to operate on standard computing hardware, including servers, desktops, laptops, and mobile devices, ensuring compatibility across a wide range of client-side hardware platforms.  
  
2.3 Software Interface Output   
The system interacts with the following software components and databases to fulfill its functional requirements:   
  
- \*\*Customer Data Entity\*\*: A database storing customer information (name, email, password) and managing customer profiles.   
- \*\*Session Data Entity\*\*: A database tracking active and terminated sessions for both customers and administrators.   
- \*\*Product Data Entity\*\*: A database containing records of all products, including name, description, price, and category associations.   
- \*\*Inventory Data Entity\*\*: A database storing inventory quantities and product availability status.   
- \*\*Order Data Entity\*\*: A database tracking order details, such as items, quantities, prices, total amount, and status.   
- \*\*Payment Data Entity\*\*: A database recording payment transactions and stored payment methods.   
- \*\*Category Data Entity\*\*: A database managing product categories, including hierarchical relationships (e.g., ParentCategoryID).   
- \*\*Plugin Data Entity\*\*: A database storing plugin configurations, installation status, and associated metadata.   
- \*\*EmailTemplate Data Entity\*\*: A database storing email templates for notifications, including subject, body, and recipient type.   
  
These software interfaces ensure data consistency, integrity, and accessibility across the system's functionalities. Developers must ensure that all database interactions are secure, efficient, and properly documented.  
  
2.4 Communication Interface Output   
The system utilizes the following communication interfaces to interact with external systems and users:   
  
- \*\*Email Notification Service\*\*:   
 - \*\*Role\*\*: Sends confirmation emails to customers after registration and checkout.   
 - \*\*Description\*\*: Integrates with an email server or third-party email service to deliver automated notifications.   
 - \*\*Inputs/Outputs\*\*:   
 - Input: OrderID, customer's email address, and template content (subject, body).   
 - Output: Confirmation email sent to the customer's email address.   
  
- \*\*Web Browser Communication\*\*:   
 - \*\*Role\*\*: Provides a web-based interface for customers and administrators to access system functionalities.   
 - \*\*Description\*\*: The system is accessed via HTTP/HTTPS protocols, ensuring compatibility with modern web browsers.   
 - \*\*Inputs/Outputs\*\*:   
 - Input: User actions (e.g., login, search, checkout) and data input (e.g., product details, payment methods).   
 - Output: Dynamic web pages and user interface updates based on user interactions and system responses.   
  
- \*\*Plugin Configuration and Documentation Access\*\*:   
 - \*\*Role\*\*: Enables administrators to configure plugins and access their documentation.   
 - \*\*Description\*\*: Plugins may interact with external APIs or internal system modules to extend functionality.   
 - \*\*Inputs/Outputs\*\*:   
 - Input: PluginID, configuration settings, or documentation requests.   
 - Output: Updated plugin configurations or documentation content displayed to the administrator.   
  
These communication interfaces ensure that the system can effectively interact with users and external services while maintaining data integrity and security. Developers should ensure robust handling of all communication protocols and provide clear error messages for failed interactions.

# Use Case

Use Case Name: Customer Registration   
Use Case ID: UC-01   
Actors: Customer, Administrator, System   
Preconditions:   
1. The system is operational and accessible.   
2. The customer is not already registered in the system.   
3. The administrator has the authority to approve or reject customer registration requests.   
  
Postconditions:   
1. A new customer profile is created in the system if the registration is approved.   
2. The customer receives a confirmation email if the registration is successful.   
3. The system logs the registration request and its status.   
  
Main Flow:   
1. The customer accesses the registration page of the system.   
2. The customer fills out the registration form with required personal information (e.g., name, email, password).   
3. The system validates the input data (e.g., checks for valid email format, password strength).   
4. The system sends the registration request to the administrator for approval.   
5. The administrator reviews the request and approves it.   
6. The system creates a new customer profile in the Customer data entity.   
7. The system sends a confirmation email to the customer’s provided email address.   
8. The customer receives the confirmation email and completes the registration process.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete input data, it displays an error message and prompts the customer to correct the information.   
2. If the customer’s email is already registered, the system displays an error message and prevents duplicate registration.   
3. If the administrator rejects the registration request, the system logs the rejection and informs the customer via email.   
4. If the system fails to send the confirmation email, it logs the error and allows the customer to request a resend.  
  
Use Case Name: Customer Login   
Use Case ID: UC-02   
Actors: Customer, System   
Preconditions:   
1. The system is operational and accessible.   
2. The customer has already been registered and approved.   
3. The customer has a valid session or has initiated the login process.   
  
Postconditions:   
1. The customer is authenticated and a session is created.   
2. The system logs the login attempt and its outcome.   
3. The customer is redirected to their account dashboard or the main page of the system.   
  
Main Flow:   
1. The customer accesses the login page of the system.   
2. The customer enters their registered email and password.   
3. The system validates the credentials against the Customer data entity.   
4. If the credentials are valid, the system creates a new session in the Session data entity.   
5. The system redirects the customer to their account dashboard.   
6. The system logs the successful login.   
  
Alternative Flow:   
1. If the email or password is incorrect, the system displays an error message and prompts the customer to retry.   
2. If the customer account is locked or disabled, the system denies access and informs the customer.   
3. If the system detects a suspicious login attempt (e.g., multiple failed attempts), it may send a verification email to the customer’s registered email address.   
4. If the verification email is sent, the customer must confirm their identity via the email link to proceed with login.  
  
Use Case Name: Customer Logout   
Use Case ID: UC-03   
Actors: Customer, System   
Preconditions:   
1. The system is operational and accessible.   
2. The customer is currently logged in and has an active session.   
  
Postconditions:   
1. The customer's session is terminated.   
2. The system logs the logout event.   
3. The customer is redirected to the login or home page.   
  
Main Flow:   
1. The customer navigates to the account settings or logout option on the system interface.   
2. The customer selects the "Logout" option.   
3. The system verifies the existence of an active session for the customer.   
4. The system terminates the session by removing the session record from the Session data entity.   
5. The system logs the logout event in the system log.   
6. The system redirects the customer to the login or home page.   
  
Alternative Flow:   
1. If the system cannot find an active session for the customer, it displays an error message indicating that the user is not logged in.   
2. If the session termination fails due to a system error, the system logs the error and displays a message to the customer requesting a retry.   
3. If the customer closes the browser or navigates away before the logout is confirmed, the system may automatically terminate the session after a timeout.  
  
Use Case Name: View Product Details   
Use Case ID: UC-04   
Actors: Customer, System, Product, Inventory, Category   
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The product to be viewed exists in the Product data entity.   
4. The product is associated with a valid inventory and category.   
  
Postconditions:   
1. The customer is presented with the detailed information of the selected product.   
2. The system logs the product view event.   
3. If the product is out of stock, the system displays a stock status message.   
  
Main Flow:   
1. The customer browses the product catalog or searches for a product.   
2. The customer selects a product to view its details.   
3. The system retrieves the product information from the Product data entity.   
4. The system checks the product's inventory status from the Inventory data entity.   
5. The system displays the product details, including price, description, stock status, and category.   
6. The system logs the customer's view of the product.   
  
Alternative Flow:   
1. If the product does not exist or is not found, the system displays an error message and suggests similar products or categories.   
2. If the inventory data is not available, the system displays a message indicating that the stock information is currently unavailable.   
3. If the product belongs to a restricted category, the system checks the customer's permissions and may restrict access accordingly.   
4. If the system fails to load product details, it logs the error and displays a message prompting the customer to try again or contact support.  
  
Use Case Name: Search Products by Category   
Use Case ID: UC-05   
Actors: Customer, System, Product, Category, Inventory   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The Product, Category, and Inventory data entities contain valid and up-to-date information.   
  
Postconditions:   
1. The customer is presented with a list of products that belong to the selected category.   
2. The system logs the search action.   
3. If no products are found in the selected category, the system informs the customer.   
  
Main Flow:   
1. The customer navigates to the product catalog or category selection page.   
2. The customer selects a category from the list of available categories.   
3. The system queries the Category data entity to retrieve the selected category details.   
4. The system filters the Product data entity based on the selected category.   
5. The system retrieves the inventory status for each product from the Inventory data entity.   
6. The system displays the filtered list of products, including their details and stock status.   
7. The system logs the search event in the system log.   
  
Alternative Flow:   
1. If the selected category does not exist or is invalid, the system displays an error message and prompts the customer to select a valid category.   
2. If the system cannot retrieve product or inventory data due to an error, it displays a message indicating the issue and suggests trying again later.   
3. If no products are found in the selected category, the system informs the customer and may suggest related categories or products.   
4. If the customer clicks on a product to view details, the system initiates the "View Product Details" use case.  
  
Use Case Name: Add Product to Cart   
Use Case ID: UC-06   
Actors: Customer, System, Product, Inventory, Order   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The product to be added exists in the Product data entity.   
4. The product has an available inventory in the Inventory data entity.   
  
Postconditions:   
1. The selected product is added to the customer's current order in the Order data entity.   
2. The system updates the inventory quantity for the product.   
3. The system logs the addition of the product to the cart.   
4. The customer receives a confirmation message that the product has been added.   
  
Main Flow:   
1. The customer views a product or browses the product catalog.   
2. The customer selects the "Add to Cart" option for the desired product.   
3. The system checks the product's availability in the Inventory data entity.   
4. The system adds the product to the customer's active order in the Order data entity.   
5. The system updates the inventory quantity by decrementing the stock count.   
6. The system displays a confirmation message to the customer.   
7. The system logs the action in the system log.   
  
Alternative Flow:   
1. If the product is out of stock, the system displays a message indicating that the product is unavailable.   
2. If the system fails to update the inventory, it logs the error and displays a message to the customer.   
3. If the customer attempts to add more units than are available in inventory, the system displays an error message and restricts the addition.   
4. If the customer is not logged in, the system prompts them to log in or register before proceeding.   
5. If the system cannot find the product in the Product data entity, it displays an error message and suggests similar products or categories.  
  
Use Case Name: Checkout   
Use Case ID: UC-07   
Actors: Customer, System, Product, Inventory, Order, Payment, Email   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one product in their cart (Order data entity).   
4. The product(s) in the cart have valid inventory quantities.   
5. The customer has provided valid shipping and billing information.   
  
Postconditions:   
1. The order is finalized and processed.   
2. The inventory quantity for the purchased product(s) is updated.   
3. A confirmation email is sent to the customer.   
4. The system logs the checkout and payment process.   
5. The customer's cart is cleared of the purchased items.   
  
Main Flow:   
1. The customer navigates to the cart or checkout page.   
2. The system displays the list of products in the customer's cart, along with total price and inventory status.   
3. The customer reviews the cart contents and confirms the purchase.   
4. The system prompts the customer to enter or confirm shipping and billing information.   
5. The customer selects a payment method and initiates the payment process.   
6. The system processes the payment via the Payment data entity.   
7. The system updates the inventory for each product in the order by decrementing the stock.   
8. The system creates a new order record in the Order data entity.   
9. The system sends a confirmation email to the customer via the Email data entity.   
10. The system logs the checkout event and displays a confirmation page to the customer.   
  
Alternative Flow:   
1. If the customer’s cart is empty, the system displays a message and redirects them to the product catalog.   
2. If the inventory for any product in the cart is insufficient, the system displays a warning and prevents checkout until the issue is resolved.   
3. If the payment fails, the system displays an error message and allows the customer to retry or choose a different payment method.   
4. If the system fails to send the confirmation email, it logs the error and allows the customer to request a resend.   
5. If the customer cancels the checkout process, the system retains the cart items and logs the cancellation.   
6. If the customer is not logged in, the system prompts them to log in or register before proceeding with checkout.  
  
Use Case Name: Receive Order Confirmation Email   
Use Case ID: UC-08   
Actors: Customer, System, Email, Order   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer has completed the checkout process.   
3. The order has been successfully processed and recorded in the Order data entity.   
4. The customer has provided a valid email address during registration or checkout.   
  
Postconditions:   
1. The customer receives an order confirmation email.   
2. The system logs the email sending event.   
3. The customer is informed of the order status and delivery details.   
  
Main Flow:   
1. The system processes the customer’s order and confirms the purchase.   
2. The system generates an order confirmation email with details such as order number, items purchased, total amount, and estimated delivery time.   
3. The system sends the email to the customer’s registered email address using the Email data entity.   
4. The system logs the successful sending of the email.   
5. The customer receives and views the confirmation email.   
  
Alternative Flow:   
1. If the customer’s email address is invalid or missing, the system logs the error and displays a message prompting the customer to update their email information.   
2. If the system fails to send the email, it logs the error and provides an option for the customer to request a resend.   
3. If the customer does not receive the email within a reasonable time, the system allows the customer to manually trigger a resend.   
4. If the system is unable to retrieve order details to include in the email, it logs the error and sends a generic confirmation message with a link to view the order online.  
  
Use Case Name: Administrator Login   
Use Case ID: UC-09   
Actors: Administrator, System, Session   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator has a valid account in the system.   
3. The administrator has not previously logged in and has no active session.   
  
Postconditions:   
1. The administrator is authenticated and a session is created in the Session data entity.   
2. The system logs the login attempt and its outcome.   
3. The administrator is redirected to the administrative dashboard.   
  
Main Flow:   
1. The administrator accesses the administrator login page of the system.   
2. The administrator enters their registered email and password.   
3. The system validates the credentials against the Administrator data entity.   
4. If the credentials are valid, the system creates a new session in the Session data entity.   
5. The system redirects the administrator to the administrative dashboard.   
6. The system logs the successful login.   
  
Alternative Flow:   
1. If the email or password is incorrect, the system displays an error message and prompts the administrator to retry.   
2. If the administrator account is locked or disabled, the system denies access and informs the administrator.   
3. If the system detects a suspicious login attempt (e.g., multiple failed attempts), it may send a verification email to the administrator’s registered email address.   
4. If the verification email is sent, the administrator must confirm their identity via the email link to proceed with login.  
  
Use Case Name: Administrator Logout   
Use Case ID: UC-10   
Actors: Administrator, System, Session   
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is currently logged in and has an active session.   
  
Postconditions:   
1. The administrator's session is terminated.   
2. The system logs the logout event.   
3. The administrator is redirected to the login or home page.   
  
Main Flow:   
1. The administrator navigates to the account settings or logout option on the administrative interface.   
2. The administrator selects the "Logout" option.   
3. The system verifies the existence of an active session for the administrator.   
4. The system terminates the session by removing the session record from the Session data entity.   
5. The system logs the logout event in the system log.   
6. The system redirects the administrator to the login or home page.   
  
Alternative Flow:   
1. If the system cannot find an active session for the administrator, it displays an error message indicating that the user is not logged in.   
2. If the session termination fails due to a system error, the system logs the error and displays a message to the administrator requesting a retry.   
3. If the administrator closes the browser or navigates away before the logout is confirmed, the system may automatically terminate the session after a timeout.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-11   
Actors: Administrator, System, Product, Inventory   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to manage inventory.   
4. The Product and Inventory data entities contain valid and up-to-date information.   
  
Postconditions:   
1. The inventory levels for selected products are updated in the Inventory data entity.   
2. The system logs the inventory management action.   
3. The administrator receives a confirmation message that the inventory has been successfully managed.   
4. Product availability status is updated based on inventory changes.   
  
Main Flow:   
1. The administrator accesses the inventory management section of the administrative dashboard.   
2. The system displays a list of products along with their current inventory levels.   
3. The administrator selects a product to modify its inventory.   
4. The system retrieves the product details from the Product data entity.   
5. The administrator updates the inventory quantity (e.g., adds stock or marks items as out of stock).   
6. The system validates the input and updates the Inventory data entity with the new quantity.   
7. The system logs the inventory change and displays a confirmation message to the administrator.   
8. If the product's availability status changes (e.g., from in stock to out of stock), the system updates the status in the Product data entity.   
  
Alternative Flow:   
1. If the product is not found in the system, the administrator receives an error message, and the system suggests similar products or categories.   
2. If the administrator enters an invalid inventory quantity (e.g., negative number), the system displays an error message and prompts for correction.   
3. If the system fails to update the inventory due to an error, it logs the failure and displays a message to the administrator.   
4. If the administrator cancels the inventory update, the system reverts to the inventory management list and logs the cancellation.  
  
Use Case Name: Add Product   
Use Case ID: UC-12   
Actors: Administrator, System, Product, Inventory, Category   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to add products.   
4. The Category and Inventory data entities contain valid and up-to-date information.   
  
Postconditions:   
1. A new product is added to the Product data entity.   
2. The product is associated with a valid inventory and category.   
3. The system logs the product addition action.   
4. The administrator receives a confirmation message that the product was successfully added.   
  
Main Flow:   
1. The administrator navigates to the product management section of the administrative dashboard.   
2. The administrator selects the "Add Product" option.   
3. The system displays a form for entering product details (e.g., name, description, price, category, inventory quantity).   
4. The administrator fills out the form with the required information.   
5. The system validates the input data (e.g., checks for valid price format, ensures category and inventory are selected).   
6. The system creates a new product record in the Product data entity.   
7. The system associates the product with the selected category and inventory.   
8. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete input data, it displays an error message and prompts the administrator to correct the information.   
2. If the selected category is invalid or not found, the system displays an error message and prompts the administrator to select a valid category.   
3. If the system fails to add the product due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system discards the entered data and logs the cancellation.  
  
Use Case Name: Update Product   
Use Case ID: UC-13   
Actors: Administrator, System, Product, Inventory, Category   
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to update product details.   
4. The product to be updated exists in the Product data entity.   
5. The product is associated with valid inventory and category data.   
  
Postconditions:   
1. The product details are updated in the Product data entity.   
2. The inventory and category associations are updated if necessary.   
3. The system logs the product update action.   
4. The administrator receives a confirmation message that the product was successfully updated.   
  
Main Flow:   
1. The administrator navigates to the product management section of the administrative dashboard.   
2. The administrator selects a product to update.   
3. The system retrieves the product details from the Product data entity.   
4. The system displays a form pre-filled with the current product information.   
5. The administrator modifies the product details (e.g., name, price, description, category, inventory quantity).   
6. The system validates the input data (e.g., checks for valid price format, ensures category and inventory are selected).   
7. The system updates the Product data entity with the new information.   
8. The system updates the Inventory and Category data entities if changes are made.   
9. The system logs the update and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete input data, it displays an error message and prompts the administrator to correct the information.   
2. If the selected category is invalid or not found, the system displays an error message and prompts the administrator to select a valid category.   
3. If the system fails to update the product due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system reverts to the product management list and logs the cancellation.  
  
Use Case Name: Delete Product   
Use Case ID: UC-14   
Actors: Administrator, System, Product, Inventory, Category   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to delete products.   
4. The product to be deleted exists in the Product data entity.   
5. The product is associated with valid inventory and category data.   
  
Postconditions:   
1. The product is removed from the Product data entity.   
2. The inventory and category associations are updated or removed accordingly.   
3. The system logs the product deletion action.   
4. The administrator receives a confirmation message that the product was successfully deleted.   
  
Main Flow:   
1. The administrator navigates to the product management section of the administrative dashboard.   
2. The administrator selects a product to delete.   
3. The system retrieves the product details from the Product data entity.   
4. The system confirms the deletion with the administrator (e.g., prompts for confirmation).   
5. The administrator confirms the deletion.   
6. The system removes the product record from the Product data entity.   
7. The system updates or removes the inventory and category associations for the deleted product.   
8. The system logs the deletion and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the product is not found in the system, the administrator receives an error message, and the system suggests similar products or categories.   
2. If the system detects that the product is currently in an active order, it displays a warning and prevents deletion until the issue is resolved.   
3. If the system fails to delete the product due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the deletion, the system reverts to the product management list and logs the cancellation.  
  
Use Case Name: Manage Product Categories   
Use Case ID: UC-15   
Actors: Administrator, System, Category, Product   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to manage product categories.   
4. The Category data entity contains valid and up-to-date category information.   
5. The Product data entity contains valid product records.   
  
Postconditions:   
1. The product categories are added, updated, or deleted in the Category data entity.   
2. Products are associated with the updated or new categories.   
3. The system logs the category management actions.   
4. The administrator receives a confirmation message for each action performed.   
  
Main Flow:   
1. The administrator navigates to the category management section of the administrative dashboard.   
2. The system displays a list of existing product categories.   
3. The administrator selects an option to add, update, or delete a category.   
4. The system validates the selected action and displays a form accordingly.   
5. The administrator fills in the required information (e.g., category name, description, parent category).   
6. The system validates the input data and checks for conflicts (e.g., duplicate category names).   
7. The system performs the selected action on the Category data entity (add, update, or delete).   
8. The system updates product associations if a category is renamed or moved.   
9. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the category name is already in use, the system displays an error message and prompts the administrator to choose a different name.   
2. If the system detects that a category is referenced by active products, it displays a warning before deletion and allows the administrator to proceed or cancel.   
3. If the administrator enters invalid or incomplete data, the system displays an error message and prompts for corrections.   
4. If the system fails to update or delete a category due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
5. If the administrator cancels the operation, the system reverts to the category management list and logs the cancellation.  
  
Use Case Name: Add Category   
Use Case ID: UC-16   
Actors: Administrator, System, Category, Product   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to add categories.   
4. The Category data entity is accessible and ready for updates.   
5. The system allows for new category creation.   
  
Postconditions:   
1. A new category is added to the Category data entity.   
2. The system logs the category addition action.   
3. The administrator receives a confirmation message that the category was successfully added.   
4. Products can be associated with the new category if needed.   
  
Main Flow:   
1. The administrator navigates to the category management section of the administrative dashboard.   
2. The administrator selects the "Add Category" option.   
3. The system displays a form for entering category details (e.g., name, description, parent category).   
4. The administrator fills out the form with the required information.   
5. The system validates the input data (e.g., checks for valid name format, ensures no duplicate category).   
6. The system adds the new category to the Category data entity.   
7. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects duplicate or invalid category name, it displays an error message and prompts the administrator to correct the information.   
2. If the administrator enters incomplete or missing data, the system displays an error message and prompts for corrections.   
3. If the system fails to add the category due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system reverts to the category management list and logs the cancellation.  
  
Use Case Name: Update Category   
Use Case ID: UC-17   
Actors: Administrator, System, Category, Product   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to update categories.   
4. The category to be updated exists in the Category data entity.   
5. The system allows for category updates.   
  
Postconditions:   
1. The category details are updated in the Category data entity.   
2. Products associated with the category are updated if necessary.   
3. The system logs the category update action.   
4. The administrator receives a confirmation message that the category was successfully updated.   
  
Main Flow:   
1. The administrator navigates to the category management section of the administrative dashboard.   
2. The administrator selects a category to update.   
3. The system retrieves the category details from the Category data entity.   
4. The system displays a form pre-filled with the current category information.   
5. The administrator modifies the category details (e.g., name, description, parent category).   
6. The system validates the input data (e.g., checks for valid name format, ensures no duplicate category name).   
7. The system updates the Category data entity with the new information.   
8. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects duplicate or invalid category name, it displays an error message and prompts the administrator to correct the information.   
2. If the administrator enters incomplete or missing data, the system displays an error message and prompts for corrections.   
3. If the system fails to update the category due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system reverts to the category management list and logs the cancellation.  
  
Use Case Name: Delete Category   
Use Case ID: UC-18   
Actors: Administrator, System, Category, Product   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to delete categories.   
4. The category to be deleted exists in the Category data entity.   
5. The system allows for category deletion.   
  
Postconditions:   
1. The selected category is removed from the Category data entity.   
2. Products previously associated with the deleted category are updated or reassigned.   
3. The system logs the category deletion action.   
4. The administrator receives a confirmation message that the category was successfully deleted.   
  
Main Flow:   
1. The administrator navigates to the category management section of the administrative dashboard.   
2. The administrator selects a category to delete.   
3. The system retrieves the category details from the Category data entity.   
4. The system confirms the deletion with the administrator (e.g., prompts for confirmation).   
5. The administrator confirms the deletion.   
6. The system removes the category record from the Category data entity.   
7. The system updates or reassigns product associations if necessary.   
8. The system logs the deletion and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects that the category is referenced by active products, it displays a warning before deletion and allows the administrator to proceed or cancel.   
2. If the category is not found in the system, the administrator receives an error message, and the system suggests similar categories.   
3. If the administrator cancels the deletion, the system reverts to the category management list and logs the cancellation.   
4. If the system fails to delete the category due to an error (e.g., database issue), it logs the error and displays a message to the administrator.  
  
Use Case Name: View Order Details   
Use Case ID: UC-04   
Actors: Customer, System, Order, Product, Inventory   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one order in the Order data entity.   
4. The order contains valid product and inventory references.   
  
Postconditions:   
1. The customer is presented with the detailed information of their selected order.   
2. The system logs the order view event.   
3. The customer is informed of the order status, product details, and inventory status at the time of the order.   
  
Main Flow:   
1. The customer navigates to their account dashboard or order history section.   
2. The customer selects an order to view its details.   
3. The system retrieves the order information from the Order data entity.   
4. The system fetches the associated product details from the Product data entity.   
5. The system checks the inventory status of the products at the time of the order.   
6. The system displays the order details, including items, quantities, prices, total amount, and order status.   
7. The system logs the customer's view of the order.   
  
Alternative Flow:   
1. If the selected order does not exist, the system displays an error message and suggests reviewing the order history.   
2. If the system cannot retrieve product or inventory details for the order, it displays a message indicating the issue and suggests trying again later.   
3. If the customer does not have permission to view the selected order, the system denies access and logs the unauthorized attempt.   
4. If the system fails to load the order details due to an error, it logs the error and displays a message prompting the customer to retry or contact support.  
  
Use Case Name: Manage Orders   
Use Case ID: UC-19   
Actors: Administrator, System, Order, Customer, Product, Inventory, Payment   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to manage orders.   
4. The Order data entity contains valid and up-to-date order records.   
5. The Customer, Product, Inventory, and Payment data entities are accessible and contain relevant data.   
  
Postconditions:   
1. The administrator can view, update, or delete existing orders.   
2. Order status and related data (e.g., payment, inventory) are updated as needed.   
3. The system logs the order management actions.   
4. The administrator receives confirmation messages for each action performed.   
  
Main Flow:   
1. The administrator accesses the order management section of the administrative dashboard.   
2. The system displays a list of all orders or filters them based on criteria (e.g., status, date).   
3. The administrator selects an order to view or modify.   
4. The system retrieves the order details from the Order data entity.   
5. The system displays the order information, including customer details, product list, total amount, payment status, and order status.   
6. The administrator selects an action (e.g., update status, delete order).   
7. The system validates the action and performs the necessary updates (e.g., changes order status to "Shipped").   
8. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the selected order does not exist, the system displays an error message and suggests reviewing the order list.   
2. If the system cannot retrieve order details due to an error, it logs the error and displays a message to the administrator.   
3. If the administrator attempts to delete an order that is still pending or has partial fulfillment, the system displays a warning and prevents the deletion until the issue is resolved.   
4. If the administrator cancels the action, the system reverts to the order management list and logs the cancellation.  
  
Use Case Name: Process Payment   
Use Case ID: UC-20   
Actors: Customer, System, Order, Payment, Email   
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one product in their cart (Order data entity).   
4. The customer has provided valid shipping and billing information.   
5. The inventory for the selected products is sufficient for the order.   
  
Postconditions:   
1. The payment is processed successfully and recorded in the Payment data entity.   
2. The order status is updated to reflect completion.   
3. A confirmation email is sent to the customer.   
4. The system logs the payment and order processing event.   
5. The customer’s cart is cleared of the purchased items.   
  
Main Flow:   
1. The customer navigates to the checkout page and selects the "Proceed to Payment" option.   
2. The system displays the total amount due and available payment methods.   
3. The customer selects a payment method and enters the required payment details (e.g., credit card information, PayPal account).   
4. The system validates the payment details for accuracy and format.   
5. The system initiates the payment processing via the Payment data entity.   
6. The system confirms the successful payment and updates the order status in the Order data entity.   
7. The system sends a confirmation email to the customer via the Email data entity.   
8. The system logs the completed payment and order processing.   
9. The customer is redirected to a confirmation page, and their cart is cleared.   
  
Alternative Flow:   
1. If the payment details are invalid or incomplete, the system displays an error message and prompts the customer to correct the information.   
2. If the payment fails during processing, the system displays an error message and allows the customer to retry or choose a different payment method.   
3. If the system is unable to send the confirmation email, it logs the error and provides an option for the customer to request a resend.   
4. If the customer cancels the payment process, the system retains the cart items and logs the cancellation.   
5. If the system detects a problem updating the order status, it logs the error and displays a message to the customer and administrator.  
  
Use Case Name: Manage Payment Information   
Use Case ID: UC-21   
Actors: Customer, System, Administrator, Payment, Email   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one order in the Order data entity.   
4. The Payment data entity is accessible and supports payment information management.   
5. The administrator has the necessary permissions to update or delete payment records if required.   
  
Postconditions:   
1. The customer’s payment information is updated, deleted, or viewed in the Payment data entity.   
2. The system logs the payment information management action.   
3. If changes are made, the system may notify the customer via email.   
4. The administrator can manage payment records if authorized.   
  
Main Flow:   
1. The customer navigates to the account settings or payment information section of their profile.   
2. The system retrieves the customer’s existing payment information from the Payment data entity.   
3. The system displays the saved payment methods (e.g., credit card, PayPal) to the customer.   
4. The customer selects an option to add, update, or delete a payment method.   
5. The system validates the selected action and displays a form accordingly.   
6. The customer enters or modifies the payment details (e.g., card number, expiration date, billing address).   
7. The system validates the input data (e.g., checks for valid card format, expiration date).   
8. The system updates or adds the payment information in the Payment data entity.   
9. The system logs the action and displays a confirmation message to the customer.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete payment information, it displays an error message and prompts the customer to correct the data.   
2. If the customer attempts to delete a payment method that is associated with an active or pending order, the system displays a warning and prevents the deletion until the issue is resolved.   
3. If the system fails to update or delete a payment record due to an error (e.g., database issue), it logs the error and displays a message to the customer.   
4. If the customer cancels the operation, the system reverts to the payment information list and logs the cancellation.   
5. If the administrator is involved in managing payment records, they must have the appropriate permissions, and all actions are logged for audit purposes.  
  
Use Case Name: Manage Sessions   
Use Case ID: UC-22  
Actors: Administrator, System, Customer, Session   
Preconditions:   
1. The system is operational and accessible.   
2. The administrator or customer has a valid account in the system.   
3. The Session data entity is accessible and ready for session management.   
4. The administrator has the necessary permissions to manage sessions for users.   
  
Postconditions:   
1. The session for a user is created, updated, or deleted in the Session data entity.   
2. The system logs all session management actions.   
3. The administrator or customer is informed of the outcome of the session management action.   
4. The user's access to the system is appropriately enabled or disabled based on session changes.   
  
Main Flow:   
1. The administrator navigates to the session management section of the administrative dashboard.   
2. The system displays a list of active or historical sessions.   
3. The administrator selects a session to view, modify, or delete.   
4. The system retrieves the session details from the Session data entity.   
5. The administrator performs the selected action (e.g., extend session, terminate session).   
6. The system updates the Session data entity accordingly.   
7. The system logs the session management action.   
8. The system provides a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the selected session does not exist, the system displays an error message and suggests reviewing the session list.   
2. If the administrator attempts to modify or delete a session without proper permissions, the system denies access and logs the unauthorized attempt.   
3. If the system fails to update or delete the session due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the session management action, the system reverts to the session list and logs the cancellation.   
5. If the system detects a session timeout or automatic logout due to inactivity, it updates the Session data entity and logs the session end.  
  
Use Case Name: Install Plugin   
Use Case ID: UC-23   
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to install plugins.   
4. The Plugin data entity is accessible and supports the installation process.   
5. A valid plugin file or configuration is available for installation.   
  
Postconditions:   
1. The plugin is successfully installed and registered in the Plugin data entity.   
2. The system logs the plugin installation action.   
3. The administrator receives a confirmation message that the plugin was installed.   
4. The installed plugin becomes available for use within the system.   
  
Main Flow:   
1. The administrator navigates to the plugin management section of the administrative dashboard.   
2. The administrator selects the "Install Plugin" option.   
3. The system displays a form or interface for uploading or configuring the plugin.   
4. The administrator uploads the plugin file or provides the necessary configuration details.   
5. The system validates the plugin file or configuration (e.g., checks for compatibility, correct format).   
6. The system installs the plugin and registers it in the Plugin data entity.   
7. The system logs the installation action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the plugin file is invalid or incompatible with the system, the system displays an error message and prompts the administrator to upload a valid file.   
2. If the system fails to install the plugin due to an error (e.g., file corruption, missing dependencies), it logs the error and displays a message to the administrator.   
3. If the administrator cancels the installation process, the system reverts to the plugin management list and logs the cancellation.   
4. If the system detects that the plugin is already installed, it displays a warning message and prevents redundant installation.  
  
Use Case Name: Uninstall Plugin   
Use Case ID: UC-24   
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to uninstall plugins.   
4. The Plugin data entity contains valid and up-to-date plugin records.   
5. The plugin to be uninstalled is currently installed in the system.   
  
Postconditions:   
1. The selected plugin is removed from the system and unregistered in the Plugin data entity.   
2. The system logs the plugin uninstallation action.   
3. The administrator receives a confirmation message that the plugin was successfully uninstalled.   
4. The plugin is no longer available for use within the system.   
  
Main Flow:   
1. The administrator navigates to the plugin management section of the administrative dashboard.   
2. The system displays a list of currently installed plugins.   
3. The administrator selects a plugin to uninstall.   
4. The system retrieves the plugin details from the Plugin data entity.   
5. The system confirms the uninstallation with the administrator (e.g., prompts for confirmation).   
6. The administrator confirms the uninstallation.   
7. The system removes the plugin from the system and updates the Plugin data entity.   
8. The system logs the uninstallation action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects that the plugin is in use or has dependencies, it displays a warning and prevents uninstallation until the issue is resolved.   
2. If the selected plugin is not found in the Plugin data entity, the system displays an error message and suggests reviewing the plugin list.   
3. If the system fails to uninstall the plugin due to an error (e.g., file lock, missing permissions), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the uninstallation, the system reverts to the plugin management list and logs the cancellation.  
  
Use Case Name: Update Plugin   
Use Case ID: UC-25   
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to update plugins.   
4. The plugin to be updated exists in the Plugin data entity.   
5. A valid updated plugin file or configuration is available for upload.   
  
Postconditions:   
1. The plugin is updated in the Plugin data entity with the new version or configuration.   
2. The system logs the plugin update action.   
3. The administrator receives a confirmation message that the plugin was successfully updated.   
4. The updated plugin becomes active and available for use within the system.   
  
Main Flow:   
1. The administrator navigates to the plugin management section of the administrative dashboard.   
2. The system displays a list of installed plugins along with their current versions.   
3. The administrator selects a plugin to update.   
4. The system retrieves the plugin details from the Plugin data entity.   
5. The system displays an interface for uploading the updated plugin file or entering new configuration details.   
6. The administrator uploads the updated plugin file or modifies the configuration.   
7. The system validates the uploaded file or configuration for compatibility and correctness.   
8. The system updates the Plugin data entity with the new version or configuration.   
9. The system logs the update action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects that the uploaded file is invalid or incompatible, it displays an error message and prompts the administrator to upload a valid file.   
2. If the system fails to update the plugin due to an error (e.g., file corruption, missing dependencies), it logs the error and displays a message to the administrator.   
3. If the administrator cancels the update process, the system reverts to the plugin management list and logs the cancellation.   
4. If the system detects that the plugin is currently in use and cannot be updated without causing disruption, it displays a warning and allows the administrator to proceed or cancel.  
  
Use Case Name: Configure Plugin   
Use Case ID: UC-26  
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to configure plugins.   
4. The plugin to be configured is already installed and registered in the Plugin data entity.   
5. The plugin has configurable settings or parameters.   
  
Postconditions:   
1. The plugin's configuration is updated in the Plugin data entity.   
2. The system logs the plugin configuration action.   
3. The administrator receives a confirmation message that the plugin configuration was successfully updated.   
4. The plugin operates based on the new configuration settings.   
  
Main Flow:   
1. The administrator navigates to the plugin management section of the administrative dashboard.   
2. The system displays a list of installed plugins with the option to configure.   
3. The administrator selects a plugin to configure.   
4. The system retrieves the plugin details and available configuration options from the Plugin data entity.   
5. The system displays a configuration form or interface with the current settings.   
6. The administrator modifies the plugin configuration as needed.   
7. The system validates the new configuration for correctness and compatibility.   
8. The system updates the Plugin data entity with the new configuration.   
9. The system logs the configuration change and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects invalid or incompatible configuration settings, it displays an error message and prompts the administrator to correct the input.   
2. If the plugin does not support configuration or no settings are available, the system displays a message indicating that no configuration is needed.   
3. If the system fails to update the plugin configuration due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the configuration process, the system reverts to the plugin management list and logs the cancellation.  
  
Use Case Name: View Plugin Documentation   
Use Case ID: UC-27   
Actors: Administrator, System, Plugin   
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to view plugin documentation.   
4. The plugin exists in the Plugin data entity and has associated documentation.   
  
Postconditions:   
1. The administrator is presented with the plugin's documentation.   
2. The system logs the documentation view event.   
3. The administrator is informed of the plugin's functionality, configuration options, and usage instructions.   
  
Main Flow:   
1. The administrator navigates to the plugin management section of the administrative dashboard.   
2. The administrator selects a plugin to view its documentation.   
3. The system retrieves the plugin details from the Plugin data entity.   
4. The system fetches the associated documentation (e.g., from a file or database).   
5. The system displays the documentation to the administrator.   
6. The system logs the documentation view event.   
  
Alternative Flow:   
1. If the selected plugin does not exist, the system displays an error message and suggests reviewing the plugin list.   
2. If the plugin does not have associated documentation, the system informs the administrator and provides an option to upload or create it.   
3. If the system fails to load the documentation due to an error, it logs the failure and displays a message prompting the administrator to try again or contact support.   
4. If the administrator cancels the view, the system reverts to the plugin management list and logs the cancellation.  
  
Use Case Name: Manage Email Templates   
Use Case ID: UC-28   
Actors: Administrator, System, Email   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to manage email templates.   
4. The Email data entity is accessible and contains existing email templates or allows for new ones to be added.   
  
Postconditions:   
1. Email templates are added, updated, or deleted in the Email data entity.   
2. The system logs the email template management actions.   
3. The administrator receives confirmation messages for each action performed.   
4. The updated email templates are available for use in the system (e.g., for order confirmations, registration emails).   
  
Main Flow:   
1. The administrator navigates to the email template management section of the administrative dashboard.   
2. The system displays a list of existing email templates with options to add, edit, or delete.   
3. The administrator selects an action (add, update, or delete an email template).   
4. The system retrieves the relevant email template details (if updating or deleting).   
5. The administrator fills in or modifies the email template information (e.g., subject, body, recipient type).   
6. The system validates the input data (e.g., ensures required fields are present, checks for valid formatting).   
7. The system updates, adds, or deletes the email template in the Email data entity.   
8. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete input data, it displays an error message and prompts the administrator to correct the information.   
2. If the administrator attempts to delete an email template that is currently in use, the system displays a warning and prevents deletion until the issue is resolved.   
3. If the system fails to update, add, or delete the email template due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system reverts to the email template list and logs the cancellation.  
  
Use Case Name: Send Order Confirmation Email   
Use Case ID: UC-29   
Actors: Customer, System, Order, Email   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer has completed the checkout process.   
3. The order has been successfully processed and recorded in the Order data entity.   
4. The customer has provided a valid email address during registration or checkout.   
  
Postconditions:   
1. The customer receives an order confirmation email.   
2. The system logs the email sending event.   
3. The customer is informed of the order status and delivery details.   
  
Main Flow:   
1. The system processes the customer’s order and confirms the purchase.   
2. The system generates an order confirmation email with details such as order number, items purchased, total amount, and estimated delivery time.   
3. The system sends the email to the customer’s registered email address using the Email data entity.   
4. The system logs the successful sending of the email.   
5. The customer receives and views the confirmation email.   
  
Alternative Flow:   
1. If the customer’s email address is invalid or missing, the system logs the error and displays a message prompting the customer to update their email information.   
2. If the system fails to send the email, it logs the error and provides an option for the customer to request a resend.   
3. If the customer does not receive the email within a reasonable time, the system allows the customer to manually trigger a resend.   
4. If the system is unable to retrieve order details to include in the email, it logs the error and sends a generic confirmation message with a link to view the order online.  
  
Use Case Name: Add to Cart   
Use Case ID: UC-30   
Actors: Customer, System, Product, Inventory, Order   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The product to be added exists in the Product data entity.   
4. The product has an available inventory in the Inventory data entity.   
5. The customer has a valid or newly created order in the Order data entity.   
  
Postconditions:   
1. The selected product is added to the customer's cart in the Order data entity.   
2. The inventory quantity for the product is updated to reflect the addition.   
3. The system logs the cart update action.   
4. The customer receives a confirmation message that the product has been added to their cart.   
  
Main Flow:   
1. The customer views a product or browses the product catalog.   
2. The customer selects the "Add to Cart" option for the desired product.   
3. The system checks the product’s availability in the Inventory data entity.   
4. The system verifies the customer’s active session and ensures an order exists for them.   
5. The system adds the product to the customer's active order in the Order data entity.   
6. The system updates the inventory quantity by decrementing the stock count for the selected product.   
7. The system displays a confirmation message to the customer, indicating the product has been successfully added to the cart.   
8. The system logs the action in the system log.   
  
Alternative Flow:   
1. If the product is out of stock, the system displays a message indicating that the product is unavailable.   
2. If the system fails to update the inventory, it logs the error and displays a message to the customer.   
3. If the customer attempts to add more units than are available in inventory, the system displays an error message and restricts the addition.   
4. If the customer is not logged in, the system prompts them to log in or register before proceeding.   
5. If the system cannot find the product in the Product data entity, it displays an error message and suggests similar products or categories.  
  
Use Case Name: Manage Cart   
Use Case ID: UC-31  
Actors: Customer, System, Product, Inventory, Order   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one product in their cart (Order data entity).   
4. The Product and Inventory data entities contain valid and up-to-date information.   
5. The cart is associated with the customer’s active order.   
  
Postconditions:   
1. The customer’s cart is updated (products added, removed, or quantities modified).   
2. The inventory quantity for affected products is updated accordingly.   
3. The system logs the cart management action.   
4. The customer receives a confirmation message for the cart changes.   
  
Main Flow:   
1. The customer navigates to the cart or order preview page.   
2. The system displays the list of products currently in the cart, including quantities, prices, and total amount.   
3. The customer selects an option to modify the cart (e.g., increase/decrease quantity, remove a product).   
4. The system validates the requested action (e.g., checks if the new quantity is within available inventory).   
5. The system updates the cart in the Order data entity based on the customer's input.   
6. If the quantity is modified, the system updates the inventory for the affected product(s) in the Inventory data entity.   
7. The system displays the updated cart with the new total and product list.   
8. The system logs the cart modification action.   
9. The customer receives a confirmation message indicating the cart has been successfully updated.   
  
Alternative Flow:   
1. If the customer attempts to add a quantity that exceeds available inventory, the system displays an error message and restricts the change.   
2. If the system fails to update the cart or inventory due to an error (e.g., database issue), it logs the error and displays a message to the customer.   
3. If the customer removes all items from the cart, the system clears the cart and logs the action.   
4. If the customer cancels the modification process, the system reverts to the previous cart state and logs the cancellation.   
5. If the system cannot retrieve cart data due to an error, it logs the issue and displays a message to the customer, suggesting a retry or contacting support.  
  
Use Case Name: View Order Details   
Use Case ID: UC-32  
Actors: Customer, System, Order, Product, Inventory   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has at least one order in the Order data entity.   
4. The order contains valid product and inventory references.   
  
Postconditions:   
1. The customer is presented with the detailed information of their selected order.   
2. The system logs the order view event.   
3. The customer is informed of the order status, product details, and inventory status at the time of the order.   
  
Main Flow:   
1. The customer navigates to their account dashboard or order history section.   
2. The customer selects an order to view its details.   
3. The system retrieves the order information from the Order data entity.   
4. The system fetches the associated product details from the Product data entity.   
5. The system checks the inventory status of the products at the time of the order.   
6. The system displays the order details, including items, quantities, prices, total amount, and order status.   
7. The system logs the customer's view of the order.   
  
Alternative Flow:   
1. If the selected order does not exist, the system displays an error message and suggests reviewing the order history.   
2. If the system cannot retrieve product or inventory details for the order, it displays a message indicating the issue and suggests trying again later.   
3. If the customer does not have permission to view the selected order, the system denies access and logs the unauthorized attempt.   
4. If the system fails to load the order details due to an error, it logs the error and displays a message prompting the customer to retry or contact support.  
  
Use Case Name: Manage Customer   
Use Case ID: UC-33  
Actors: Administrator, System, Customer   
  
Preconditions:   
1. The system is operational and accessible.   
2. The administrator is logged in and has an active session.   
3. The administrator has the necessary permissions to manage customer accounts.   
4. The Customer data entity contains valid and up-to-date customer records.   
5. The system provides an interface for managing customer profiles.   
  
Postconditions:   
1. The customer profile is added, updated, or deleted in the Customer data entity.   
2. The system logs the customer management action.   
3. The administrator receives a confirmation message for each action performed.   
4. If a customer is updated or deleted, the associated data (e.g., orders, cart, payment info) may be adjusted or restricted accordingly.   
  
Main Flow:   
1. The administrator navigates to the customer management section of the administrative dashboard.   
2. The system displays a list of existing customers with options to add, edit, or delete a customer profile.   
3. The administrator selects an action (e.g., add a new customer, update an existing one, or delete a customer).   
4. The system retrieves the relevant customer details (if updating or deleting).   
5. The administrator fills in or modifies the customer information (e.g., name, email, account status, role).   
6. The system validates the input data (e.g., ensures valid email format, checks for duplicate entries).   
7. The system performs the selected action on the Customer data entity (e.g., creates a new customer, updates the profile, or deletes the account).   
8. The system logs the action and displays a confirmation message to the administrator.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete input data, it displays an error message and prompts the administrator to correct the information.   
2. If the administrator attempts to delete a customer account that is associated with active orders or payments, the system displays a warning and prevents deletion until the issue is resolved.   
3. If the system fails to update, add, or delete the customer record due to an error (e.g., database issue), it logs the error and displays a message to the administrator.   
4. If the administrator cancels the operation, the system reverts to the customer management list and logs the cancellation.   
5. If the administrator selects a customer that does not exist, the system displays an error message and suggests reviewing the customer list.  
  
Use Case Name: Manage Payment Information   
Use Case ID: UC-34  
Actors: Customer, System, Payment, Administrator   
  
Preconditions:   
1. The system is operational and accessible.   
2. The customer is logged in and has an active session.   
3. The customer has a valid Payment data entity associated with their account.   
4. The system provides an interface for managing saved payment information.   
5. The administrator has the necessary permissions to view or modify customer payment records if required for system maintenance or support.   
  
Postconditions:   
1. The customer's payment information is updated, deleted, or viewed in the Payment data entity.   
2. The system logs the payment information management action.   
3. The customer is informed of the outcome of the payment information management action.   
4. If changes are made, the system may update related order or transaction records to reflect the new payment information.   
  
Main Flow:   
1. The customer navigates to the account settings or payment information section of their profile.   
2. The system retrieves the customer's existing payment information from the Payment data entity.   
3. The system displays the saved payment methods (e.g., credit card, PayPal) to the customer.   
4. The customer selects an option to add, update, or delete a payment method.   
5. The system validates the selected action and displays a form accordingly.   
6. The customer enters or modifies the payment details (e.g., card number, expiration date, billing address).   
7. The system validates the input data (e.g., checks for valid card format, expiration date).   
8. The system updates, adds, or deletes the payment information in the Payment data entity.   
9. The system logs the action and displays a confirmation message to the customer.   
  
Alternative Flow:   
1. If the system detects invalid or incomplete payment information, it displays an error message and prompts the customer to correct the data.   
2. If the customer attempts to delete a payment method that is associated with an active or pending order, the system displays a warning and prevents deletion until the issue is resolved.   
3. If the system fails to update or delete a payment record due to an error (e.g., database issue), it logs the error and displays a message to the customer.   
4. If the customer cancels the operation, the system reverts to the payment information list and logs the cancellation.   
5. If the administrator is involved in managing payment records (e.g., for system maintenance or support), they must have the appropriate permissions, and all actions are logged for audit purposes.