项目文档

# Functional Requirement

\*\*Chapter 1: Functional Requirements\*\*   
\*System Name: GAMMA-J Web Store\*  
  
---  
  
### 1.1 Product Management Function   
\*\*Function ID:\*\* FR-01   
\*\*Description:\*\* Administrators and plugins can create, update, and delete products. Each product is associated with a category and can be part of an order.   
\*\*Input:\*\*   
- Product details (Name, Description, Price, Stock, CategoryID)   
- Administrator credentials and permissions   
- Plugin configuration (if applicable)   
\*\*Output:\*\*   
- A new or updated product record in the database   
- A notification to the administrator confirming the action   
- Error message (if validation or technical failure occurs)  
  
---  
  
### 1.2 Product View Function   
\*\*Function ID:\*\* FR-02   
\*\*Description:\*\* Administrators and customers can view the details of a specific product. This function retrieves and displays product information such as name, description, price, and stock status.   
\*\*Input:\*\*   
- ProductID or product name for search   
- User credentials (Administrator or Customer)   
\*\*Output:\*\*   
- Displayed product details   
- Error message if product not found or system failure occurs  
  
---  
  
### 1.3 Order Management Function   
\*\*Function ID:\*\* FR-03   
\*\*Description:\*\* Administrators can create, update, and delete orders. Orders are associated with a customer, and each order contains one or more order items.   
\*\*Input:\*\*   
- Order details (CustomerID, OrderDate, TotalAmount, Status)   
- OrderItems (ProductID, Quantity, Subtotal)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- A new or updated order record in the database   
- A notification to the administrator confirming the action   
- Error message if order or order item is invalid, or if system or plugin fails  
  
---  
  
### 1.4 Order View Function   
\*\*Function ID:\*\* FR-04   
\*\*Description:\*\* Administrators and customers can view the details of a specific order, including the associated products, payment status, and order status.   
\*\*Input:\*\*   
- OrderID   
- User credentials (Administrator or Customer)   
\*\*Output:\*\*   
- Displayed order details   
- Error message if order not found or system failure occurs  
  
---  
  
### 1.5 Order Status Update Function   
\*\*Function ID:\*\* FR-05   
\*\*Description:\*\* Administrators can update the status of an order (e.g., Processing, Shipped, Cancelled). The system validates the status and updates related records, including inventory and notifications.   
\*\*Input:\*\*   
- OrderID   
- New status (e.g., "Shipped")   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- Updated order status in the database   
- Notifications to customer and administrator   
- Error message if invalid status or system/plugin failure occurs  
  
---  
  
### 1.6 Order Item Management Function   
\*\*Function ID:\*\* FR-06   
\*\*Description:\*\* Administrators can manage order items by editing or removing them from an order. This function updates the order total and product inventory accordingly.   
\*\*Input:\*\*   
- OrderItemID   
- Updated quantity or price (if editing)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- Updated or removed order item in the database   
- Updated order total and inventory   
- Notification to the administrator   
- Error message if order is finalized or system/plugin failure occurs  
  
---  
  
### 1.7 Customer Management Function   
\*\*Function ID:\*\* FR-07   
\*\*Description:\*\* Administrators can create, update, and delete customer records. A customer can also view their own profile.   
\*\*Input:\*\*   
- Customer details (Name, Email, PhoneNumber, Address)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- A new or updated customer record in the database   
- A notification to the administrator confirming the action   
- Error message if invalid data or dependencies exist  
  
---  
  
### 1.8 Customer View Function   
\*\*Function ID:\*\* FR-08   
\*\*Description:\*\* Customers and administrators can view a customer's profile and order history.   
\*\*Input:\*\*   
- CustomerID   
- User credentials (Administrator or Customer)   
\*\*Output:\*\*   
- Displayed customer profile and order history   
- Error message if customer not found or system failure occurs  
  
---  
  
### 1.9 Payment Processing Function   
\*\*Function ID:\*\* FR-09   
\*\*Description:\*\* The system processes payments for orders using a selected payment method. This function updates the payment and order status and sends a confirmation notification.   
\*\*Input:\*\*   
- OrderID   
- PaymentMethodID   
- Payment amount and date   
- User credentials (Customer or Administrator)   
\*\*Output:\*\*   
- A new or updated payment record in the database   
- Updated order status to "Paid"   
- Notification to customer and administrator   
- Error message if payment fails, order is invalid, or system/plugin failure occurs  
  
---  
  
### 1.10 Payment View Function   
\*\*Function ID:\*\* FR-10   
\*\*Description:\*\* Administrators and customers can view the details of a specific payment, including transaction ID, amount, method, and status.   
\*\*Input:\*\*   
- PaymentID   
- User credentials (Administrator or Customer)   
\*\*Output:\*\*   
- Displayed payment details   
- Error message if payment not found or system failure occurs  
  
---  
  
### 1.11 Notification Management Function   
\*\*Function ID:\*\* FR-11   
\*\*Description:\*\* The system can send and view notifications to users (customers and administrators). Notifications may be sent via plugins (e.g., email, SMS).   
\*\*Input:\*\*   
- Notification content (Message)   
- Recipient details (CustomerID or AdminID)   
- Plugin configuration (if applicable)   
\*\*Output:\*\*   
- Notification record in the database   
- Delivered notification to the user   
- Error message if recipient is invalid or plugin fails  
  
---  
  
### 1.12 Notification View Function   
\*\*Function ID:\*\* FR-12   
\*\*Description:\*\* Users (administrators and customers) can view their notifications. The system marks notifications as viewed upon display.   
\*\*Input:\*\*   
- UserID (Administrator or Customer)   
- User credentials   
\*\*Output:\*\*   
- List of notifications with content and timestamp   
- Marked notifications as viewed   
- Error message if no notifications or system failure occurs  
  
---  
  
### 1.13 Plugin Management Function   
\*\*Function ID:\*\* FR-13   
\*\*Description:\*\* Administrators can manage plugins, including adding, removing, and updating plugin configurations. Plugins can extend system functionality for notifications, payments, and order processing.   
\*\*Input:\*\*   
- Plugin details (Name, Description, Version, Configuration)   
- Plugin file (if adding a new plugin)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- Plugin record in the database   
- Plugin status updated (activated, deactivated, or removed)   
- Notification to the administrator   
- Error message if plugin is incompatible, fails to initialize, or in use  
  
---  
  
### 1.14 Product Category Management Function   
\*\*Function ID:\*\* FR-14   
\*\*Description:\*\* Administrators can manage product categories by creating, updating, or deleting them. This function ensures that products are correctly categorized and that dependencies are handled.   
\*\*Input:\*\*   
- Category details (Name, Description, ParentCategory)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- Updated or new product category record in the database   
- Notification to the administrator   
- Error message if category is in use or system/plugin failure occurs  
  
---  
  
### 1.15 Payment Method Management Function   
\*\*Function ID:\*\* FR-15   
\*\*Description:\*\* Administrators can manage available payment methods by creating, updating, or deleting them. This function ensures that payment methods are compatible with the system and not currently in use.   
\*\*Input:\*\*   
- Payment method details (Name, Description, Configuration)   
- Administrator credentials and permissions   
\*\*Output:\*\*   
- Updated or new payment method record in the database   
- Notification to the administrator   
- Error message if method is in use or system/plugin failure occurs  
  
---  
  
This concludes \*\*Chapter 1: Functional Requirements\*\* of the \*\*GAMMA-J Web Store Software Requirement Specification (SRS)\*\*. Each function is aligned with the provided use cases and data model, ensuring feasibility, completeness, and consistency.

# External Description

# \*\*Chapter 2: External Interfaces\*\*   
\*\*System Name: GAMMA-J Web Store\*\*  
  
This chapter defines the external interfaces that the GAMMA-J Web Store system must interact with to fulfill its functional requirements. These interfaces include user interfaces, hardware interfaces, software interfaces, and communication interfaces. Each interface is described in detail, including its purpose, interaction method, and relevant input/output data. The goal is to ensure that developers and stakeholders understand the external dependencies and communication channels required for the system to operate effectively.  
  
---  
  
## \*\*2.1 User Interface\*\*  
  
The GAMMA-J Web Store interacts with end users through a web-based user interface (UI) that is accessible via modern web browsers. The UI is designed to be responsive and user-friendly, supporting both administrators and customers in performing their respective tasks.  
  
### \*\*2.1.1 Administrator Interface\*\*  
- \*\*Purpose:\*\* Provides administrators with tools to manage products, orders, customers, payment methods, and plugins.  
- \*\*Interaction Method:\*\* Web-based dashboard with forms, buttons, and interactive elements.  
- \*\*Inputs:\*\*  
 - Product details (Name, Description, Price, Stock, CategoryID)  
 - Order details (CustomerID, OrderDate, TotalAmount, Status)  
 - OrderItems (ProductID, Quantity, Subtotal)  
 - Customer details (Name, Email, PhoneNumber, Address)  
 - Payment method details (Name, Description, Configuration)  
 - Plugin details (Name, Description, Version, Configuration)  
 - User credentials for authentication and authorization  
- \*\*Outputs:\*\*  
 - Confirmation messages for successful actions (e.g., "Product created successfully")  
 - Error messages for invalid input or system failures  
 - Updated displays of product, order, and customer data  
 - Notifications (e.g., success, warning, or error messages)  
 - Plugin status updates (activated, deactivated, removed)  
  
### \*\*2.1.2 Customer Interface\*\*  
- \*\*Purpose:\*\* Allows customers to view product details, place and view orders, check order status, and review their payment and notification history.  
- \*\*Interaction Method:\*\* Web-based front-end with search functionality and order management panels.  
- \*\*Inputs:\*\*  
 - ProductID or product name for search  
 - OrderID for viewing or managing an order  
 - CustomerID for viewing their profile and order history  
 - PaymentMethodID for selecting a payment method  
 - User credentials for authentication and authorization  
- \*\*Outputs:\*\*  
 - Displayed product details (name, description, price, stock status)  
 - Displayed order details (products, status, payment status)  
 - Displayed customer profile and order history  
 - Displayed payment details (transaction ID, amount, method, status)  
 - Notifications (e.g., order status updates, payment confirmations)  
 - Error messages for invalid inputs or system failures  
  
---  
  
## \*\*2.2 Hardware Interface\*\*  
  
The GAMMA-J Web Store does not have direct dependencies on specific hardware components beyond the standard infrastructure required for web application deployment and user access.  
  
### \*\*2.2.1 Web Browsers\*\*  
- \*\*Purpose:\*\* Enable user interaction with the system via graphical interfaces.  
- \*\*Description:\*\* The system must be compatible with major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.  
- \*\*Supported Devices:\*\* Desktop computers, laptops, tablets, and mobile devices.  
- \*\*Requirements:\*\*  
 - Support for HTML5, CSS3, and modern JavaScript frameworks  
 - Responsive design for cross-device compatibility  
 - No special hardware requirements beyond a standard internet-connected device  
  
---  
  
## \*\*2.3 Software Interface\*\*  
  
The GAMMA-J Web Store interacts with several software components, including databases, plugins, and external tools for payment processing, notification delivery, and system management.  
  
### \*\*2.3.1 Database Interface\*\*  
- \*\*Purpose:\*\* Store and retrieve product, customer, order, payment, and notification data.  
- \*\*Description:\*\* The system interacts with a relational database (e.g., MySQL, PostgreSQL) to manage structured data.  
- \*\*Interaction Method:\*\* SQL queries or an ORM (Object-Relational Mapping) layer for database operations.  
- \*\*Inputs:\*\*  
 - Product, order, customer, payment, and notification data  
 - Queries for data retrieval (e.g., by ProductID, OrderID, CustomerID)  
- \*\*Outputs:\*\*  
 - Updated or newly inserted records in the database  
 - Retrieved records for display or further processing  
 - Error messages if database operations fail (e.g., invalid query, connection failure)  
  
### \*\*2.3.2 Plugin Interface\*\*  
- \*\*Purpose:\*\* Allow administrators to extend system functionality via third-party or custom plugins.  
- \*\*Description:\*\* Plugins are integrated into the system to support notifications, payment processing, and order management.  
- \*\*Interaction Method:\*\* Plugin files are uploaded and configured through the system’s Plugin Management UI. Each plugin must conform to a defined API or SDK to interact with the system.  
- \*\*Inputs:\*\*  
 - Plugin configuration data (Name, Description, Version, Configuration)  
 - Plugin files (e.g., .zip or .jar)  
 - Trigger events from the system (e.g., "Order Created", "Payment Success")  
- \*\*Outputs:\*\*  
 - Plugin status updates (activated, deactivated, removed)  
 - Notifications sent to users (via email, SMS, etc.)  
 - Payment processing results (success, failure, pending)  
 - Error messages if the plugin is incompatible or fails to initialize  
  
### \*\*2.3.3 Payment Gateway Interface\*\*  
- \*\*Purpose:\*\* Interface with external payment gateways to process customer payments.  
- \*\*Description:\*\* The system must support integration with various payment gateways (e.g., PayPal, Stripe, Alipay) via their APIs.  
- \*\*Interaction Method:\*\* RESTful API calls to the external payment gateway with appropriate authentication (e.g., API keys).  
- \*\*Inputs:\*\*  
 - OrderID and payment amount  
 - PaymentMethodID to determine the correct gateway  
 - Transaction data (e.g., customer information, payment method details)  
- \*\*Outputs:\*\*  
 - Payment confirmation or failure status  
 - Transaction ID and timestamp  
 - Updated payment and order status in the system  
 - Error messages for failed transactions or invalid inputs  
  
---  
  
## \*\*2.4 Communication Interface\*\*  
  
The system communicates with external entities using standard communication protocols and methods, including email, web services, and message queues.  
  
### \*\*2.4.1 Email Communication Interface\*\*  
- \*\*Purpose:\*\* Send notifications to users via email.  
- \*\*Description:\*\* The system sends emails to administrators and customers to confirm actions such as order updates, payment success, and plugin changes.  
- \*\*Interaction Method:\*\* SMTP (Simple Mail Transfer Protocol) or REST API integration with email service providers (e.g., SendGrid, Mailgun).  
- \*\*Inputs:\*\*  
 - Recipient email address (derived from user account)  
 - Notification message content (text or HTML)  
 - Timestamp of the event  
- \*\*Outputs:\*\*  
 - Email sent to the user  
 - Notification record stored in the database  
 - Error messages if the email fails to send  
  
### \*\*2.4.2 SMS Communication Interface (Optional)\*\*  
- \*\*Purpose:\*\* Send text notifications to users via SMS.  
- \*\*Description:\*\* The system may support SMS delivery for real-time notifications (e.g., order status changes, payment confirmations).  
- \*\*Interaction Method:\*\* REST API integration with SMS service providers (e.g., Twilio, Nexmo).  
- \*\*Inputs:\*\*  
 - Recipient phone number (from user account)  
 - Notification message content  
 - Timestamp of the event  
- \*\*Outputs:\*\*  
 - SMS message sent to the user  
 - Notification record stored in the database  
 - Error messages if the SMS fails to send  
  
### \*\*2.4.3 Web Communication Interface\*\*  
- \*\*Purpose:\*\* Enable interaction between the system and external web services or APIs.  
- \*\*Description:\*\* The system uses HTTP/HTTPS for communication with external systems, such as payment gateways and notification plugins.  
- \*\*Interaction Method:\*\* RESTful API calls with JSON or XML payloads.  
- \*\*Inputs:\*\*  
 - API request parameters (e.g., OrderID, PaymentMethodID, Plugin configuration)  
 - Authentication tokens or API keys  
- \*\*Outputs:\*\*  
 - API response data (e.g., payment confirmation, plugin status)  
 - Error responses if the external service is unavailable or returns an error  
  
### \*\*2.4.4 Notification Push Interface (Optional)\*\*  
- \*\*Purpose:\*\* Deliver real-time notifications to users via push messages (e.g., web push, mobile push).  
- \*\*Description:\*\* This interface allows the system to send instant alerts to users who have opted-in.  
- \*\*Interaction Method:\*\* Integration with push notification services (e.g., Firebase Cloud Messaging, Apple Push Notification Service).  
- \*\*Inputs:\*\*  
 - User device token  
 - Notification message content  
 - Timestamp and priority level  
- \*\*Outputs:\*\*  
 - Push notification delivered to the user's device  
 - Notification record stored in the database  
 - Error messages if the push fails (e.g., invalid token)  
  
---  
  
## \*\*2.5 External Data Sources\*\*  
  
All external data sources referenced in the functional requirements are covered under the \*\*Software Interface\*\* (Section 2.3), particularly the \*\*Database Interface\*\* and \*\*Plugin Interface\*\*. The system does not rely on external hardware data sources directly but may access external APIs for data processing and delivery.  
  
- \*\*Database:\*\* Used to store and retrieve all core system data including products, orders, customers, and notifications.  
- \*\*Plugins:\*\* Extend the system's functionality and may interface with external services such as payment gateways or notification channels.  
- \*\*Payment Gateways:\*\* External services used to process transactions and validate payment methods.  
  
---  
  
## \*\*2.6 Interface Summary\*\*  
  
| Interface Type | Description | Interaction Method |  
|-----------------------|-----------------------------------------------------------------------------|-----------------------------------------------------|  
| User Interface | Web-based UI for administrators and customers | Web forms, buttons, and interactive dashboards |  
| Web Browsers | Standard web browser interaction | HTML5, CSS3, JavaScript |  
| Database Interface | Relational database for storing and retrieving system data | SQL or ORM (e.g., Hibernate, Sequelize) |  
| Plugin Interface | Integration with third-party or custom plugins | REST API or system-defined plugin SDK |  
| Payment Gateway | External services for processing customer payments | RESTful API calls (e.g., Stripe, PayPal) |  
| Email Communication | Sending notifications via email | SMTP or REST API (e.g., SendGrid) |  
| SMS Communication | Optional interface for sending SMS notifications | REST API (e.g., Twilio) |  
| Notification Push | Optional interface for real-time push notifications | Push notification services (e.g., Firebase) |  
| Web Communication | General HTTP/HTTPS communication for API integrations | RESTful API calls with JSON/XML payloads |  
  
---  
  
This chapter concludes the definition of \*\*Chapter 2: External Interfaces\*\* in the \*\*GAMMA-J Web Store Software Requirements Specification (SRS)\*\*. It provides a comprehensive overview of how the system interacts with users, hardware, software, and external communication channels. Developers can use this section to understand the required integrations and ensure that the system is built with compatibility and scalability in mind.

# Use Case

Use Case Name: Create Product   
Use Case ID: UC-01   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to create a new Product. The system must have access to the Product data entity and its associated fields.   
Postconditions: A new Product is successfully created and stored in the system. A Notification is sent to the Administrator confirming the creation.   
  
Main Flow:   
1. The Administrator navigates to the Product management section of the system.   
2. The Administrator selects the "Create Product" option.   
3. The system displays a form for entering Product details (e.g., name, description, price, inventory).   
4. The Administrator fills in the required information for the new Product.   
5. The Administrator submits the form.   
6. The system validates the input data.   
7. The system creates a new Product record in the database.   
8. The system generates a Notification indicating that the Product was successfully created.   
9. The Administrator is redirected to the Product list, where the new Product appears.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, incorrect format), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during Product creation, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the creation process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Update Product Information   
Use Case ID: UC-02   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to update Product information. The system must have access to the Product data entity and its current record.   
Postconditions: The Product information is successfully updated in the system. A Notification is sent to the Administrator confirming the update.   
  
Main Flow:   
1. The Administrator navigates to the Product management section of the system.   
2. The Administrator selects an existing Product from the list.   
3. The system displays the current Product details in an editable form.   
4. The Administrator modifies the necessary fields (e.g., name, description, price, inventory).   
5. The Administrator submits the updated form.   
6. The system validates the input data.   
7. The system updates the Product record in the database.   
8. The system generates a Notification indicating that the Product was successfully updated.   
9. The Administrator is redirected to the Product list, where the updated Product details are visible.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, incorrect format), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the update process, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the update, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Delete Product   
Use Case ID: UC-03   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to delete a Product. The system must have access to the Product data entity and the specific record to be deleted.   
Postconditions: The selected Product is successfully removed from the system. A Notification is sent to the Administrator confirming the deletion.   
  
Main Flow:   
1. The Administrator navigates to the Product management section of the system.   
2. The Administrator selects an existing Product from the list and chooses the "Delete Product" option.   
3. The system prompts the Administrator to confirm the deletion.   
4. The Administrator confirms the deletion.   
5. The system checks for any associated Orders or Payments linked to the Product.   
6. If no dependencies are found, the system deletes the Product record from the database.   
7. The system generates a Notification indicating that the Product was successfully deleted.   
8. The Administrator is redirected to the Product list, where the deleted Product no longer appears.   
  
Alternative Flow:   
1. If there are associated Orders or Payments, the system displays an error message and prevents deletion.   
2. If the Administrator cancels the deletion confirmation, the system returns to the Product list without making any changes.   
3. If the system encounters a technical error during deletion, it displays an error message and logs the issue for further investigation.   
4. If the Plugin is involved and fails during the deletion process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: View Product Details   
Use Case ID: UC-04   
Actors: Administrator, Customer, Plugin   
Preconditions: The Administrator or Customer is logged in and has access to the Product catalog. The system must have access to the Product data entity and the specific Product record to be viewed.   
Postconditions: The requested Product details are displayed to the user. No changes are made to the system data.   
  
Main Flow:   
1. The Administrator or Customer navigates to the Product catalog.   
2. The user selects a specific Product from the list.   
3. The system retrieves the Product details from the database.   
4. The system displays the Product information (e.g., name, description, price, inventory status).   
5. The user reviews the information and may return to the catalog or take further actions (e.g., add to cart).   
  
Alternative Flow:   
1. If the requested Product does not exist or cannot be found, the system displays an error message.   
2. If the system encounters a technical error while retrieving Product details, it displays an error message and logs the issue.   
3. If the Plugin is involved and fails during the retrieval process, the system halts the operation and provides feedback to the user about the Plugin failure.  
  
Use Case Name: Register Customer   
Use Case ID: UC-05   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to register a new Customer. The system must have access to the Customer data entity and its associated fields.   
Postconditions: A new Customer is successfully created and stored in the system. A Notification is sent to the Administrator confirming the registration.   
  
Main Flow:   
1. The Administrator navigates to the Customer management section of the system.   
2. The Administrator selects the "Register Customer" option.   
3. The system displays a form for entering Customer details (e.g., name, email, phone number, address).   
4. The Administrator fills in the required information for the new Customer.   
5. The Administrator submits the form.   
6. The system validates the input data.   
7. The system creates a new Customer record in the database.   
8. The system generates a Notification indicating that the Customer was successfully registered.   
9. The Administrator is redirected to the Customer list, where the new Customer appears.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, duplicate email), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during Customer registration, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the registration process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: View Customer Profile   
Use Case ID: UC-06   
Actors: Administrator, Customer, Plugin   
Preconditions: The Administrator or Customer is logged in and has the necessary permissions to view a Customer profile. The system must have access to the Customer data entity and the specific Customer record to be viewed.   
Postconditions: The requested Customer profile is displayed to the user. No changes are made to the system data.   
  
Main Flow:   
1. The Administrator or Customer navigates to the Customer profile section of the system.   
2. The user selects a specific Customer from the list or views their own profile.   
3. The system retrieves the Customer profile details from the database.   
4. The system displays the Customer information (e.g., name, email, phone number, order history).   
5. The user reviews the information and may return to the previous screen or take further actions (e.g., update profile).   
  
Alternative Flow:   
1. If the requested Customer does not exist or cannot be found, the system displays an error message.   
2. If the system encounters a technical error while retrieving the Customer profile, it displays an error message and logs the issue.   
3. If the Plugin is involved and fails during the retrieval process, the system halts the operation and provides feedback to the user about the Plugin failure.  
  
Use Case Name: Update Customer Information   
Use Case ID: UC-07   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to update Customer information. The system must have access to the Customer data entity and the specific Customer record to be modified.   
Postconditions: The Customer information is successfully updated in the system. A Notification is sent to the Administrator confirming the update.   
  
Main Flow:   
1. The Administrator navigates to the Customer management section of the system.   
2. The Administrator selects an existing Customer from the list.   
3. The system displays the current Customer details in an editable form.   
4. The Administrator modifies the necessary fields (e.g., name, email, phone number, address).   
5. The Administrator submits the updated form.   
6. The system validates the input data.   
7. The system updates the Customer record in the database.   
8. The system generates a Notification indicating that the Customer was successfully updated.   
9. The Administrator is redirected to the Customer list, where the updated Customer details are visible.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, duplicate email), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the update process, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the update, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Delete Customer Account   
Use Case ID: UC-08   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to delete a Customer account. The system must have access to the Customer data entity and the specific Customer record to be deleted.   
Postconditions: The selected Customer account is successfully removed from the system. A Notification is sent to the Administrator confirming the deletion.   
  
Main Flow:   
1. The Administrator navigates to the Customer management section of the system.   
2. The Administrator selects an existing Customer from the list and chooses the "Delete Customer" option.   
3. The system prompts the Administrator to confirm the deletion.   
4. The Administrator confirms the deletion.   
5. The system checks for any associated Orders or Payments linked to the Customer.   
6. If no dependencies are found, the system deletes the Customer record from the database.   
7. The system generates a Notification indicating that the Customer account was successfully deleted.   
8. The Administrator is redirected to the Customer list, where the deleted Customer no longer appears.   
  
Alternative Flow:   
1. If there are associated Orders or Payments, the system displays an error message and prevents deletion.   
2. If the Administrator cancels the deletion confirmation, the system returns to the Customer list without making any changes.   
3. If the system encounters a technical error during deletion, it displays an error message and logs the issue for further investigation.   
4. If the Plugin is involved and fails during the deletion process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Place Order   
Use Case ID: UC-09   
Actors: Customer, Administrator, Payment, Notification, Plugin   
Preconditions: The Customer is logged in and has at least one Product selected in their cart. The system must have access to the Order, Product, and Payment data entities. The selected Products must be in stock.   
Postconditions: A new Order is successfully created in the system. A Payment is processed for the Order. A Notification is sent to the Customer confirming the order placement and to the Administrator for record-keeping.   
  
Main Flow:   
1. The Customer navigates to the shopping cart and selects the "Place Order" option.   
2. The system displays a summary of the selected Products, including total price and inventory status.   
3. The Customer confirms the order details and selects a payment method.   
4. The system creates a new Order record in the database, linking it to the Customer and the selected Products.   
5. The system processes the Payment for the Order using the selected method.   
6. If the payment is successful, the system updates the inventory of the ordered Products.   
7. The system generates a Notification to the Customer confirming the order and payment.   
8. The system logs the Order and Payment details for the Administrator.   
9. The Customer is redirected to a confirmation page.   
  
Alternative Flow:   
1. If the selected Products are out of stock, the system displays an error message and prevents order submission.   
2. If the payment fails, the system displays an error message and cancels the order creation.   
3. If the system encounters a technical error during order or payment processing, it displays an error message and logs the issue.   
4. If the Plugin is involved and fails during the order placement process, the system halts the operation and provides feedback to the Customer and Administrator.   
5. If the Customer cancels the order before payment is processed, the system returns to the cart without creating an Order.  
  
Use Case Name: View Order Details   
Use Case ID: UC-10   
Actors: Administrator, Customer, Plugin   
Preconditions: The Administrator or Customer is logged in and has the necessary permissions to view Order details. The system must have access to the Order data entity and the specific Order record to be viewed.   
Postconditions: The requested Order details are displayed to the user. No changes are made to the system data.   
  
Main Flow:   
1. The Administrator or Customer navigates to the Order section of the system.   
2. The user selects a specific Order from the list.   
3. The system retrieves the Order details, including associated Products, total amount, and Payment status.   
4. The system displays the Order information (e.g., order number, date, customer name, product list, shipping status).   
5. The user reviews the Order details and may return to the Order list or take further actions (e.g., update status).   
  
Alternative Flow:   
1. If the requested Order does not exist or cannot be found, the system displays an error message.   
2. If the system encounters a technical error while retrieving Order details, it displays an error message and logs the issue.   
3. If the Plugin is involved and fails during the retrieval process, the system halts the operation and provides feedback to the user about the Plugin failure.  
  
Use Case Name: Update Order Status   
Use Case ID: UC-11   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to update Order status. The system must have access to the Order data entity and the specific Order record to be modified.   
Postconditions: The Order status is successfully updated in the system. A Notification is sent to the Administrator and the associated Customer informing them of the status change.   
  
Main Flow:   
1. The Administrator navigates to the Order management section of the system.   
2. The Administrator selects an existing Order from the list.   
3. The system displays the current Order details, including the current status.   
4. The Administrator selects a new status (e.g., "Shipped," "Cancelled," "Processing").   
5. The Administrator confirms the status update.   
6. The system validates the selected status and checks for any constraints (e.g., inventory availability if status is "Shipped").   
7. The system updates the Order status in the database.   
8. The system generates a Notification to the Administrator and the Customer about the status change.   
9. The Administrator is redirected to the Order list, where the updated status is visible.   
  
Alternative Flow:   
1. If the selected status is invalid or not applicable (e.g., "Shipped" for an order with no payment), the system displays an error message and prompts the Administrator to choose a valid status.   
2. If the system encounters a technical error during the update, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the update process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Delete Order   
Use Case ID: UC-12   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to delete an Order. The system must have access to the Order data entity and the specific Order record to be deleted.   
Postconditions: The selected Order is successfully removed from the system. A Notification is sent to the Administrator confirming the deletion.   
  
Main Flow:   
1. The Administrator navigates to the Order management section of the system.   
2. The Administrator selects an existing Order from the list and chooses the "Delete Order" option.   
3. The system prompts the Administrator to confirm the deletion.   
4. The Administrator confirms the deletion.   
5. The system checks for any constraints (e.g., whether the Order has been processed or shipped).   
6. If no constraints are present, the system deletes the Order record from the database.   
7. The system generates a Notification indicating that the Order was successfully deleted.   
8. The Administrator is redirected to the Order list, where the deleted Order no longer appears.   
  
Alternative Flow:   
1. If the Order has already been processed (e.g., shipped or partially fulfilled), the system displays an error message and prevents deletion.   
2. If the Administrator cancels the deletion confirmation, the system returns to the Order list without making any changes.   
3. If the system encounters a technical error during deletion, it displays an error message and logs the issue for further investigation.   
4. If the Plugin is involved and fails during the deletion process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Process Payment   
Use Case ID: UC-13   
Actors: Customer, Administrator, Order, Payment, Notification, Plugin   
Preconditions: The Customer has placed an Order and selected a payment method. The system must have access to the Order and Payment data entities. The Order must have a valid total amount and be in a "Pending Payment" status.   
Postconditions: The Payment is successfully processed, and the Order status is updated to "Paid." A Notification is sent to the Customer and Administrator confirming the payment.   
  
Main Flow:   
1. The system receives a request to process the Payment for a confirmed Order.   
2. The system verifies the Order status to ensure it is eligible for payment processing.   
3. The system initiates the Payment transaction using the selected method (e.g., credit card, digital wallet).   
4. The Payment gateway confirms the transaction success.   
5. The system updates the Order status to "Paid" and records the Payment details.   
6. The system sends a Notification to the Customer confirming the successful payment.   
7. The system logs the Payment and Order update for the Administrator.   
8. The Customer is redirected to an order confirmation page.   
  
Alternative Flow:   
1. If the Payment transaction fails (e.g., insufficient funds, invalid card details), the system displays an error message and the Order remains in "Pending Payment" status.   
2. If the Order status is not eligible for payment (e.g., already paid or cancelled), the system prevents the operation and informs the user.   
3. If the system encounters a technical error during payment processing, it displays an error message and logs the issue for further investigation.   
4. If the Plugin is involved and fails during the payment process, the system halts the operation and provides feedback to the Customer and Administrator.  
  
Use Case Name: View Payment Details   
Use Case ID: UC-14   
Actors: Administrator, Customer, Plugin   
Preconditions: The Administrator or Customer is logged in and has the necessary permissions to view Payment details. The system must have access to the Payment data entity and the specific Payment record to be viewed.   
Postconditions: The requested Payment details are displayed to the user. No changes are made to the system data.   
  
Main Flow:   
1. The Administrator or Customer navigates to the Payment section of the system.   
2. The user selects a specific Payment from the list.   
3. The system retrieves the Payment details from the database, including the associated Order and Customer.   
4. The system displays the Payment information (e.g., transaction ID, amount, date, status, payment method).   
5. The user reviews the information and may return to the Payment list or take further actions (e.g., refund).   
  
Alternative Flow:   
1. If the requested Payment does not exist or cannot be found, the system displays an error message.   
2. If the system encounters a technical error while retrieving Payment details, it displays an error message and logs the issue.   
3. If the Plugin is involved and fails during the retrieval process, the system halts the operation and provides feedback to the user about the Plugin failure.  
  
Use Case Name: Manage Administrator Account   
Use Case ID: UC-15   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to manage other Administrator accounts or their own account. The system must have access to the Administrator data entity and its associated fields.   
Postconditions: The Administrator account is either created, updated, or deleted successfully. A Notification is sent to the Administrator confirming the action.   
  
Main Flow:   
1. The Administrator navigates to the Administrator management section of the system.   
2. The Administrator selects the "Create Administrator," "Update Administrator," or "Delete Administrator" option.   
3. For "Create Administrator," the system displays a form for entering details (e.g., username, password, role, email).   
4. The Administrator fills in the required information and submits the form.   
5. The system validates the input data and checks for existing accounts with the same username or email.   
6. If valid and no conflicts, the system creates, updates, or deletes the Administrator account in the database.   
7. The system generates a Notification indicating the success of the operation.   
8. The Administrator is redirected to the Administrator list, where the changes are visible.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, duplicate username or email), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the operation, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the process, the system halts the operation and provides feedback to the Administrator about the Plugin failure.  
  
Use Case Name: Send Notification   
Use Case ID: UC-16   
Actors: Administrator, Customer, Plugin   
Preconditions: The system must have access to the Notification data entity. A valid message content and recipient must be specified. The Plugin may be involved to enhance notification delivery (e.g., email, SMS, or push notifications).   
Postconditions: The Notification is successfully sent to the intended recipient. The system logs the notification attempt. If the Plugin is used, its execution is confirmed.   
  
Main Flow:   
1. The system determines that a Notification needs to be sent (e.g., after a Product is created, an Order is placed, or a Payment is processed).   
2. The system prepares the Notification message, including content and recipient details (e.g., email, phone number).   
3. The system checks if a Plugin is required for the delivery method (e.g., sending an email or SMS).   
4. If a Plugin is required, the system invokes the Plugin to send the Notification.   
5. The system confirms the successful delivery of the Notification.   
6. The system logs the Notification event for record-keeping.   
  
Alternative Flow:   
1. If the recipient details are invalid (e.g., invalid email format, inactive phone number), the system displays an error message and halts the notification process.   
2. If the Plugin is involved and fails to deliver the Notification (e.g., service unavailable, API error), the system logs the failure and displays an error message to the Administrator.   
3. If the system encounters a technical error while preparing or sending the Notification, it displays an error message and logs the issue for further investigation.  
  
Use Case Name: View Notification   
Use Case ID: UC-17   
Actors: Administrator, Customer, Plugin   
Preconditions: The user (Administrator or Customer) is logged in and has access to the Notification data entity. There must be at least one Notification associated with the user.   
Postconditions: The Notification is displayed to the user. The system marks the Notification as viewed if applicable.   
  
Main Flow:   
1. The Administrator or Customer navigates to the Notification section of the system.   
2. The system retrieves the list of Notifications associated with the user.   
3. The system displays the Notifications in a user-friendly format (e.g., list with timestamps and content).   
4. The user selects a specific Notification to view.   
5. The system displays the full details of the selected Notification.   
6. If the Notification has not been viewed before, the system marks it as viewed.   
  
Alternative Flow:   
1. If there are no Notifications for the user, the system displays a message indicating no notifications are available.   
2. If the system encounters a technical error while retrieving Notifications, it displays an error message and logs the issue.   
3. If the Plugin is involved and fails during the retrieval or display of Notifications, the system halts the operation and provides feedback to the user.  
  
Use Case Name: Manage Plugin   
Use Case ID: UC-18   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to manage Plugins. The system must have access to the Plugin data entity and its associated configuration settings.   
Postconditions: The Plugin is either activated, configured, updated, or deactivated successfully. A Notification is sent to the Administrator confirming the action.   
  
Main Flow:   
1. The Administrator navigates to the Plugin management section of the system.   
2. The Administrator selects the "Manage Plugin" option.   
3. The system displays a list of available Plugins with their current status and configuration options.   
4. The Administrator selects a specific Plugin and chooses an action (e.g., activate, update, configure, deactivate).   
5. The system validates the action and checks if the Plugin is compatible with the current system setup.   
6. The system executes the selected action (e.g., activates the Plugin or updates its configuration).   
7. The system generates a Notification confirming the Plugin action was successful.   
8. The Administrator is redirected to the Plugin list, where the updated Plugin status or configuration is visible.   
  
Alternative Flow:   
1. If the selected Plugin is not compatible with the system, the system displays an error message and prevents the action.   
2. If the Plugin configuration is incomplete or invalid, the system prompts the Administrator to correct the settings before proceeding.   
3. If the system encounters a technical error during Plugin management, it displays an error message and logs the issue for further investigation.   
4. If the Plugin fails to initialize or respond after activation, the system halts the operation and provides feedback to the Administrator.  
  
Use Case Name: Add Plugin   
Use Case ID: UC-19   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to add a new Plugin. The system must have access to the Plugin data entity and its associated fields. The Plugin must be compatible with the system and available for integration.   
Postconditions: A new Plugin is successfully added and activated in the system. A Notification is sent to the Administrator confirming the addition. The Plugin is ready for use in enhancing system functionality.   
  
Main Flow:   
1. The Administrator navigates to the Plugin management section of the system.   
2. The Administrator selects the "Add Plugin" option.   
3. The system displays a form for entering Plugin details (e.g., name, type, configuration settings).   
4. The Administrator fills in the required Plugin information and uploads the Plugin file if required.   
5. The Administrator submits the form.   
6. The system validates the Plugin information and checks for compatibility.   
7. The system installs and activates the Plugin.   
8. The system generates a Notification confirming the Plugin was successfully added.   
9. The Administrator is redirected to the Plugin list, where the new Plugin appears.   
  
Alternative Flow:   
1. If the Plugin information is invalid or incomplete, the system displays an error message and prompts the Administrator to correct it.   
2. If the Plugin is incompatible with the system, the system prevents the addition and displays an error message.   
3. If the Plugin file is missing or corrupted, the system displays an error message and halts the installation process.   
4. If the system encounters a technical error during Plugin installation, it displays an error message and logs the issue for further investigation.   
5. If the Plugin fails to initialize after installation, the system halts the operation and provides feedback to the Administrator.  
  
Use Case Name: Remove Plugin   
Use Case ID: UC-20   
Actors: Administrator   
Preconditions: The Administrator is logged in and has the necessary permissions to remove a Plugin. The system must have access to the Plugin data entity and the specific Plugin to be removed. The Plugin should not be in use for any critical system functions.   
Postconditions: The selected Plugin is successfully removed from the system. A Notification is sent to the Administrator confirming the removal. The system no longer uses the Plugin for any functionality.   
  
Main Flow:   
1. The Administrator navigates to the Plugin management section of the system.   
2. The Administrator selects an existing Plugin from the list and chooses the "Remove Plugin" option.   
3. The system prompts the Administrator to confirm the removal.   
4. The Administrator confirms the removal.   
5. The system checks if the Plugin is currently in use or affecting any active processes.   
6. If no dependencies are found, the system removes the Plugin from the database and deactivates it.   
7. The system generates a Notification confirming the Plugin was successfully removed.   
8. The Administrator is redirected to the Plugin list, where the removed Plugin no longer appears.   
  
Alternative Flow:   
1. If the Plugin is currently in use or affecting active processes, the system displays an error message and prevents removal.   
2. If the Administrator cancels the removal confirmation, the system returns to the Plugin list without making any changes.   
3. If the system encounters a technical error during removal, it displays an error message and logs the issue for further investigation.   
4. If the Plugin fails to deactivate or respond to the removal command, the system halts the operation and provides feedback to the Administrator.  
  
Use Case Name: Manage OrderItem   
Use Case ID: UC-21   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to manage OrderItems. The system must have access to the Order and OrderItem data entities and the specific OrderItem record to be modified.   
Postconditions: The OrderItem is either updated or removed successfully. A Notification is sent to the Administrator confirming the action. If the OrderItem is removed, the system updates the associated Order’s total amount and inventory accordingly.   
  
Main Flow:   
1. The Administrator navigates to the Order management section of the system.   
2. The Administrator selects an existing Order from the list and accesses its OrderItems.   
3. The system displays a list of OrderItems within the selected Order, including details such as Product name, quantity, and price.   
4. The Administrator selects a specific OrderItem and chooses an action (e.g., "Edit OrderItem" or "Remove OrderItem").   
5. If the action is "Edit OrderItem," the system displays an editable form for modifying the OrderItem (e.g., changing quantity or price).   
6. The Administrator updates the necessary fields and submits the form.   
7. The system validates the input data and recalculates the Order’s total amount if necessary.   
8. The system updates the OrderItem record in the database and logs the change.   
9. If the action is "Remove OrderItem," the system prompts the Administrator to confirm the removal.   
10. The Administrator confirms the removal.   
11. The system removes the OrderItem from the Order and updates the Order's total amount and inventory.   
12. The system generates a Notification to the Administrator confirming the successful action.   
13. The Administrator is redirected to the Order details page, where the changes to the OrderItem are visible.   
  
Alternative Flow:   
1. If the input data for editing an OrderItem is invalid (e.g., negative quantity, incorrect price format), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the management of the OrderItem, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the management process (e.g., during inventory update or notification delivery), the system halts the operation and provides feedback to the Administrator about the Plugin failure.   
4. If the Administrator cancels the removal confirmation, the system returns to the Order details page without making any changes.   
5. If the OrderItem is linked to a completed or shipped Order, the system prevents modification or removal and displays an error message.  
  
Use Case Name: Manage ProductCategory   
Use Case ID: UC-22   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to manage ProductCategories. The system must have access to the ProductCategory data entity and its associated fields.   
Postconditions: The ProductCategory is either created, updated, or deleted successfully. A Notification is sent to the Administrator confirming the action. If a ProductCategory is updated or deleted, the system ensures that any associated Products are updated or flagged appropriately.   
  
Main Flow:   
1. The Administrator navigates to the ProductCategory management section of the system.   
2. The Administrator selects the "Create ProductCategory," "Update ProductCategory," or "Delete ProductCategory" option.   
3. For "Create ProductCategory," the system displays a form for entering category details (e.g., name, description, parent category).   
4. The Administrator fills in the required information and submits the form.   
5. The system validates the input data and checks for existing categories with the same name.   
6. If valid and no conflicts, the system creates, updates, or deletes the ProductCategory in the database.   
7. The system generates a Notification confirming the success of the operation.   
8. The Administrator is redirected to the ProductCategory list, where the changes are visible.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, duplicate category name), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the operation, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the process (e.g., during logging or notification delivery), the system halts the operation and provides feedback to the Administrator.   
4. If the Administrator attempts to delete a ProductCategory that is associated with existing Products, the system displays a warning and prevents deletion until all dependencies are resolved.  
  
Use Case Name: Manage PaymentMethod   
Use Case ID: UC-23   
Actors: Administrator, Plugin   
Preconditions: The Administrator is logged in and has the necessary permissions to manage PaymentMethods. The system must have access to the PaymentMethod data entity and its associated fields. The PaymentMethod must be compatible with the system's existing payment processing workflows.   
Postconditions: The PaymentMethod is either created, updated, or deleted successfully. A Notification is sent to the Administrator confirming the action. If a PaymentMethod is deleted, the system ensures that no active Orders are using it and updates the associated Payment records accordingly.   
  
Main Flow:   
1. The Administrator navigates to the PaymentMethod management section of the system.   
2. The Administrator selects the "Create PaymentMethod," "Update PaymentMethod," or "Delete PaymentMethod" option.   
3. For "Create PaymentMethod," the system displays a form for entering payment method details (e.g., name, type, description, configuration settings).   
4. The Administrator fills in the required information and submits the form.   
5. The system validates the input data and checks for existing PaymentMethods with the same name or configuration.   
6. If valid and no conflicts, the system creates, updates, or deletes the PaymentMethod in the database.   
7. The system generates a Notification indicating the success of the operation.   
8. The Administrator is redirected to the PaymentMethod list, where the changes are visible.   
  
Alternative Flow:   
1. If the input data is invalid (e.g., missing required fields, duplicate PaymentMethod name), the system displays an error message and prompts the Administrator to correct the information.   
2. If the system encounters a technical error during the operation, it displays an error message and logs the issue for further investigation.   
3. If the Plugin is involved and fails during the process (e.g., during integration testing or notification delivery), the system halts the operation and provides feedback to the Administrator.   
4. If the Administrator attempts to delete a PaymentMethod that is currently in use by active Orders, the system displays a warning and prevents deletion until all dependencies are resolved.