项目文档

# Functional Requirement

### Chapter 1: Functional Requirements  
  
\*\*1.1 Customer Registration Function\*\*   
Function ID: FR-01   
Description: Creates new customer accounts with secure credential storage and auto-generated shopping carts.   
Input: Registration form data (Username, Password, Email, PhoneNumber, FirstName, LastName, Address, City, State, Country, PostalCode)   
Output: New Customer entity, linked Personal Information entity, and Shopping Cart entity   
  
\*\*1.2 Customer Login Function\*\*   
Function ID: FR-02   
Description: Authenticates customers and initiates secure sessions using stored credentials.   
Input: Login credentials (Email, Password)   
Output: Active session token and Shopping Cart linkage   
  
\*\*1.3 Customer Logout Function\*\*   
Function ID: FR-03   
Description: Terminates authenticated sessions while preserving cart contents.   
Input: Active session token   
Output: Session termination confirmation and cart state preservation   
  
\*\*1.4 Personal Information View Function\*\*   
Function ID: FR-04   
Description: Retrieves and displays stored personal details in read-only format.   
Input: CustomerID   
Output: Formatted Personal Information entity data   
  
\*\*1.5 Personal Information Update Function\*\*   
Function ID: FR-05   
Description: Modifies stored personal details with real-time validation.   
Input: Updated Personal Information fields (Email, Address, etc.)   
Output: Updated Personal Information entity   
  
\*\*1.6 Purchase History View Function\*\*   
Function ID: FR-06   
Description: Compiles and displays chronological order records.   
Input: CustomerID   
Output: List of Order entities with itemized details   
  
\*\*1.7 Payment Information View Function\*\*   
Function ID: FR-07   
Description: Retrieves and displays payment methods with sensitive data masking.   
Input: CustomerID   
Output: Masked Payment Information entities (e.g., ●●●●1234)   
  
\*\*1.8 Payment Information Update Function\*\*   
Function ID: FR-08   
Description: Modifies stored payment methods via gateway verification.   
Input: Updated Payment Information fields (CardNumber, ExpirationDate, CVV)   
Output: Verified and encrypted Payment Information entity   
  
\*\*1.9 Category Creation Function\*\*   
Function ID: FR-09   
Description: Adds new product categories with uniqueness validation.   
Input: CategoryName, CategoryDescription   
Output: New Category entity   
  
\*\*1.10 Category Modification Function\*\*   
Function ID: FR-10   
Description: Edits existing category attributes and propagates changes.   
Input: CategoryID, Updated fields (CategoryName/Description)   
Output: Updated Category entity   
  
\*\*1.11 Category Deletion Function\*\*   
Function ID: FR-11   
Description: Removes unused categories after integrity verification.   
Input: CategoryID   
Output: Deletion confirmation and inventory relationship updates   
  
\*\*1.12 Product Creation Function\*\*   
Function ID: FR-12   
Description: Adds new products with linked inventory initialization.   
Input: ProductName, Description, Price, CategoryID, InitialStock   
Output: New Product entity and Inventory entity   
  
\*\*1.13 Product Modification Function\*\*   
Function ID: FR-13   
Description: Updates product details and synchronizes inventory data.   
Input: ProductID, Updated fields (Price/Description/Stock)   
Output: Updated Product and Inventory entities   
  
\*\*1.14 Product Deletion Function\*\*   
Function ID: FR-14   
Description: Removes products after verifying no active transactions.   
Input: ProductID   
Output: Deletion confirmation and inventory association removal   
  
\*\*1.15 Product Categorization Function\*\*   
Function ID: FR-15   
Description: Assigns products to active categories with concurrency checks.   
Input: ProductID, CategoryID   
Output: Updated Product-Category relationship   
  
\*\*1.16 Stock Update Function\*\*   
Function ID: FR-16   
Description: Modifies inventory quantities with cart reservation checks.   
Input: ProductID, NewStockQuantity   
Output: Updated Inventory entity and cart synchronization   
  
\*\*1.17 Inventory Restock Function\*\*   
Function ID: FR-17   
Description: Increases stock levels with safety buffer handling.   
Input: ProductID, RestockQuantity   
Output: Updated Inventory entity with new\_stock calculation   
  
\*\*1.18 Cart Addition Function\*\*   
Function ID: FR-18   
Description: Adds products to cart with real-time inventory validation.   
Input: ProductID, Quantity   
Output: New or updated Cart Item entity   
  
\*\*1.19 Cart View Function\*\*   
Function ID: FR-19   
Description: Displays cart contents with pricing and availability status.   
Input: CartID   
Output: Formatted Shopping Cart entity with linked Cart Items and Products   
  
\*\*1.20 Cart Item Update Function\*\*   
Function ID: FR-20   
Description: Modifies item quantities within inventory constraints.   
Input: CartItemID, NewQuantity   
Output: Updated Cart Item entity   
  
\*\*1.21 Cart Item Removal Function\*\*   
Function ID: FR-21   
Description: Deletes items from cart and releases inventory reservations.   
Input: CartItemID   
Output: Removal confirmation and stock release   
  
\*\*1.22 Cart Clearance Function\*\*   
Function ID: FR-22   
Description: Removes all cart items and resets totals.   
Input: CartID   
Output: Empty Shopping Cart entity and inventory release   
  
\*\*1.23 Checkout Initiation Function\*\*   
Function ID: FR-23   
Description: Validates cart items and prepares payment processing.   
Input: CartID, PaymentMethod   
Output: Checkout entity with transaction summary   
  
\*\*1.24 Checkout Cancellation Function\*\*   
Function ID: FR-24   
Description: Terminates checkout while preserving cart state.   
Input: CheckoutID   
Output: Cancellation confirmation and cart preservation   
  
\*\*1.25 Order Finalization Function\*\*   
Function ID: FR-25   
Description: Creates confirmed orders after payment verification.   
Input: CheckoutID   
Output: Order entity, Order Confirmation entity, inventory deduction   
  
\*\*1.26 Order View Function\*\*   
Function ID: FR-26   
Description: Retrieves and displays order details with linked entities.   
Input: OrderID   
Output: Formatted Order Confirmation data with itemized products   
  
\*\*1.27 Order Cancellation Function\*\*   
Function ID: FR-27   
Description: Cancels eligible orders and initiates refunds.   
Input: OrderID   
Output: "Canceled" Order status, inventory restoration, refund record   
  
\*\*1.28 Confirmation Email Function\*\*   
Function ID: FR-28   
Description: Generates and dispatches order confirmation emails.   
Input: OrderID   
Output: Email dispatch record in Order Confirmation entity   
  
\*\*1.29 Plugin Installation Function\*\*   
Function ID: FR-29   
Description: Integrates new plugins with compatibility validation.   
Input: PluginFile   
Output: Activated Plugin entity and API Specifications linkage   
  
\*\*1.30 Plugin Removal Function\*\*   
Function ID: FR-30   
Description: Uninstalls plugins with dependency checks.   
Input: PluginID   
Output: Removal confirmation and configuration rollback   
  
\*\*1.31 API Specification View Function\*\*   
Function ID: FR-31   
Description: Retrieves and displays documented plugin interfaces.   
Input: PluginID   
Output: Structured API Specifications data   
  
\*\*1.32 Customer Management Function\*\*   
Function ID: FR-32   
Description: Creates/updates/deactivates customer accounts by administrators.   
Input: Customer data fields, Action type (Add/Edit/Deactivate)   
Output: Modified Customer entity and linked resource updates   
  
\*\*1.33 Administrator Management Function\*\*   
Function ID: FR-33   
Description: Manages admin accounts with privilege controls.   
Input: Administrator data fields, Action type (Add/Edit/Deactivate)   
Output: Modified Administrator entity and session synchronization

# External Description

### Chapter 2: External Interfaces  
  
This chapter describes the external interfaces that the system interacts with, categorized into user interfaces, hardware interfaces, software interfaces, and communication interfaces.  
  
#### 2.1 User Interfaces  
  
The system provides several user interfaces to facilitate interaction with customers and administrators.  
  
1. \*\*Customer Registration Interface\*\*  
 - \*\*Description\*\*: A web-based interface where customers can input their personal details, including username, password, email, phone number, and address.  
 - \*\*Interaction\*\*: The interface collects user input and validates it before storing the data in the system's database.  
  
2. \*\*Customer Login Interface\*\*  
 - \*\*Description\*\*: A web-based interface where customers can enter their email and password to access their account.  
 - \*\*Interaction\*\*: The interface authenticates the user and initiates a secure session upon successful login.  
  
3. \*\*Personal Information Management Interface\*\*  
 - \*\*Description\*\*: A web-based interface where customers can view and update their personal information, such as email and address.  
 - \*\*Interaction\*\*: The interface allows customers to input updated information, which is validated and stored in the database.  
  
4. \*\*Product Management Interface\*\*  
 - \*\*Description\*\*: A web-based interface used by administrators to add, modify, or delete products, including product name, description, price, and category.  
 - \*\*Interaction\*\*: The interface collects product details and updates the database, ensuring inventory data is synchronized.  
  
5. \*\*Order Management Interface\*\*  
 - \*\*Description\*\*: A web-based interface where customers can view their order history and details, including order status and itemized products.  
 - \*\*Interaction\*\*: The interface retrieves order data from the database and displays it in a formatted manner.  
  
6. \*\*Plugin Management Interface\*\*  
 - \*\*Description\*\*: A web-based interface used by administrators to install, configure, and remove plugins, including compatibility validation.  
 - \*\*Interaction\*\*: The interface handles plugin file uploads, validates compatibility, and updates plugin configurations.  
  
7. \*\*Administrator Management Interface\*\*  
 - \*\*Description\*\*: A web-based interface for managing administrator accounts, including adding, editing, and deactivating accounts.  
 - \*\*Interaction\*\*: The interface collects administrator data, validates it, and updates the database, ensuring session synchronization.  
  
#### 2.2 Hardware Interfaces  
  
The system interacts with the following hardware interfaces:  
  
1. \*\*Payment Terminal Interface\*\*  
 - \*\*Description\*\*: An interface that connects the system with payment terminals for processing payment transactions.  
 - \*\*Interaction\*\*: The interface communicates with payment terminals to receive payment information and send transaction confirmations.  
  
2. \*\*Inventory Management Interface\*\*  
 - \*\*Description\*\*: An interface that connects the system with inventory management hardware, such as barcode scanners.  
 - \*\*Interaction\*\*: The interface collects inventory data from hardware devices and updates the inventory database.  
  
#### 2.3 Software Interfaces  
  
The system interfaces with several software components and external systems:  
  
1. \*\*Payment Gateway API\*\*  
 - \*\*Description\*\*: An API that enables secure payment processing, including credit card transactions and payment verification.  
 - \*\*Interaction\*\*: The system sends payment details to the API, which processes the payment and returns a confirmation or error message.  
  
2. \*\*Email Service API\*\*  
 - \*\*Description\*\*: An API used to send confirmation emails, such as order confirmations and password reset emails.  
 - \*\*Interaction\*\*: The system sends email content and recipient details to the API, which handles the delivery of emails.  
  
3. \*\*Plugin API\*\*  
 - \*\*Description\*\*: An API that allows third-party plugins to integrate with the system, providing additional functionalities.  
 - \*\*Interaction\*\*: The system communicates with the plugin API to install, configure, and execute plugin functionalities.  
  
4. \*\*Inventory Management System\*\*  
 - \*\*Description\*\*: A software system that tracks and manages product inventory levels.  
 - \*\*Interaction\*\*: The system sends inventory updates and retrieves inventory data from the inventory management system.  
  
#### 2.4 Communication Interfaces  
  
The system uses the following communication interfaces to interact with external systems and users:  
  
1. \*\*Email Communication\*\*  
 - \*\*Description\*\*: The system sends confirmation emails to customers, such as order confirmations and password reset instructions.  
 - \*\*Interaction\*\*: The system uses an email service API to send emails, ensuring timely delivery to the recipient.  
  
2. \*\*Push Notifications\*\*  
 - \*\*Description\*\*: The system sends push notifications to users for order updates, payment confirmations, and other important alerts.  
 - \*\*Interaction\*\*: The system uses a push notification service to send real-time notifications to users' devices.  
  
3. \*\*API Communication\*\*  
 - \*\*Description\*\*: The system communicates with external APIs, such as payment gateways and email services, to perform specific tasks.  
 - \*\*Interaction\*\*: The system sends HTTP requests to the APIs, receives responses, and processes the data accordingly.  
  
### Summary  
  
This chapter has outlined the external interfaces that the system interacts with, ensuring that all external data sources mentioned in the functional requirements are clearly defined. The interfaces are categorized into user interfaces, hardware interfaces, software interfaces, and communication interfaces, each with a clear description of their role and interaction method. This provides a comprehensive understanding of how the system interfaces with external entities, facilitating accurate implementation and integration.

# Use Case

### Use Case Description   
\*\*Use Case Name:\*\* Register Customer   
\*\*Use Case ID:\*\* UC-01   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation and data storage)   
  
\*\*Preconditions:\*\*   
1. Customer accesses the system's registration interface.   
2. No prior authentication is required.   
  
\*\*Postconditions:\*\*   
1. A new Customer entity is created in the system.   
2. Personal Information entity is stored and linked to the Customer.   
3. Customer gains authenticated access to the system.   
  
\*\*Main Flow:\*\*   
1. Customer selects the "Register" option.   
2. System displays a registration form requesting Personal Information (e.g., name, email, password).   
3. Customer enters required details and submits the form.   
4. System validates data format (e.g., email syntax, password strength).   
5. System verifies email uniqueness against existing Customer records.   
6. System creates a new Customer entity and associated Personal Information.   
7. System auto-generates an empty Shopping Cart for the new Customer.   
8. System confirms successful registration and logs in the Customer.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid data at Step 4:\*\*   
 - 4a. System rejects submission, highlights errors, and prompts re-entry.   
 - 4b. Customer corrects data and resubmits.   
2. \*\*Duplicate email at Step 5:\*\*   
 - 5a. System notifies Customer that the email is already registered.   
 - 5b. Customer enters a new email or selects "Forgot Password."   
3. \*\*Submission timeout:\*\*   
 - 3a. System discards partial data after 5 minutes of inactivity.   
 - 3b. Customer restarts the registration process.   
  
---   
\*(Consistent with standard use case structure; length optimized for core actions.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Login Customer   
\*\*Use Case ID:\*\* UC-02   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for authentication and session management)   
  
\*\*Preconditions:\*\*   
1. Customer is registered in the system.   
2. Customer has valid credentials (email and password).   
3. System login interface is accessible.   
  
\*\*Postconditions:\*\*   
1. Customer is authenticated and granted access to the system.   
2. Customer session is initiated and linked to their Shopping Cart and Order history.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to the login page.   
2. System displays login form with email and password fields.   
3. Customer enters credentials and submits the form.   
4. System validates credentials against stored Customer and Personal Information entities.   
5. System authenticates Customer, creates a session, and loads associated Shopping Cart.   
6. System redirects Customer to the dashboard/homepage.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid Credentials at Step 4:\*\*   
 - 4a. System rejects login, displays error message: "Invalid email or password."   
 - 4b. System clears password field and allows re-entry.   
2. \*\*Forgot Password Triggered:\*\*   
 - 3a. Customer selects "Forgot Password" before submission.   
 - 3b. System interrupts login flow and initiates password reset process (UC-03).   
3. \*\*Session Conflict:\*\*   
 - 5a. If Customer has an existing active session, system terminates prior session and notifies: "Previous session ended."   
 - 5b. Proceeds with new session creation.   
  
---   
\*(Consistent with UC-01 structure; core flows emphasize authentication and session handling.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Logout Customer   
\*\*Use Case ID:\*\* UC-03   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for session termination and cleanup)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated and has an active session.   
2. System dashboard/interface is accessible to the Customer.   
  
\*\*Postconditions:\*\*   
1. Customer session is terminated and authentication tokens revoked.   
2. Shopping Cart state is preserved for future sessions.   
3. Customer is redirected to a public page (e.g., login screen or homepage).   
  
\*\*Main Flow:\*\*   
1. Customer selects "Logout" from the system interface.   
2. System verifies active session validity.   
3. System terminates session, clears authentication tokens, and disconnects Shopping Cart linkage.   
4. System redirects Customer to the public login page.   
5. System displays confirmation: "You have been logged out successfully."   
  
\*\*Alternative Flow:\*\*   
1. \*\*Session expiration during logout:\*\*   
 - 2a. If session expired pre-emptively, system skips termination and notifies: "Session already expired."   
 - 2b. System directly redirects to login page.   
2. \*\*Concurrent session conflict:\*\*   
 - 3a. If multiple active sessions exist (e.g., other devices), system terminates all associated sessions.   
 - 3b. System logs termination audit trail in Order Confirmation history.   
3. \*\*Network disruption:\*\*   
 - 4a. If redirect fails due to connectivity loss, system displays local confirmation: "Logout complete. Reconnect to continue."   
 - 4b. Customer manually reloads the interface.   
  
---   
\*(Consistent with UC-01/UC-02 structure; emphasizes session invalidation and state preservation.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Personal Information   
\*\*Use Case ID:\*\* UC-04   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Customer is registered and authenticated in the system (per UC-01/UC-02).   
2. Customer has valid Personal Information stored in the system.   
3. Customer accesses the profile management interface.   
  
\*\*Postconditions:\*\*   
1. Customer's Personal Information is displayed in read-only format.   
2. Audit trail of access is recorded in system logs.   
  
\*\*Main Flow:\*\*   
1. Customer selects "View Profile" from account dashboard.   
2. System fetches Personal Information entity linked to authenticated Customer.   
3. System displays Personal Information (name, contact details, address) in structured UI.   
4. Customer reviews information and selects "Back" to exit.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Missing Personal Information at Step 2:\*\*   
 - 2a. System detects no stored Personal Information for Customer.   
 - 2b. System displays placeholder message: "No profile data available. Update your profile?" with link to edit function.   
2. \*\*Session timeout during retrieval:\*\*   
 - 3a. If authentication expires before Step 3 completion, system terminates flow.   
 - 3b. System redirects to login page (UC-02) with message: "Session expired. Re-authenticate to view profile."   
3. \*\*Data corruption error:\*\*   
 - 2a. System fails to decrypt/parse stored Personal Information.   
 - 2b. System displays sanitized error: "Information temporarily unavailable" and logs exception for Administrator review.   
  
---   
\*(Consistent with UC-01/UC-02 structure; focuses on data retrieval integrity and session-aware access control.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Personal Information   
\*\*Use Case ID:\*\* UC-05   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data validation and storage)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated and has an active session (per UC-02).   
2. Customer has accessed the profile management interface (per UC-04).   
  
\*\*Postconditions:\*\*   
1. Personal Information entity is updated in the system.   
2. Updated data propagates to dependent entities (e.g., Order Confirmation).   
3. System displays success confirmation and updated information.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Edit Profile" on profile management interface.   
2. System loads current Personal Information into editable form.   
3. Customer modifies fields (e.g., email, address) and submits changes.   
4. System validates format and completeness of updated data.   
5. If email changed, system verifies uniqueness against existing Customer records.   
6. System updates Personal Information entity and linked Customer record.   
7. System displays updated profile with success notification.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Validation failure at Step 4:\*\*   
 - 4a. System rejects submission, highlights invalid fields with error messages.   
 - 4b. Customer corrects data and resubmits.   
2. \*\*Duplicate email at Step 5:\*\*   
 - 5a. System notifies: "Email already registered" and retains form data.   
 - 5b. Customer modifies email or cancels update.   
3. \*\*Session timeout during Steps 2-6:\*\*   
 - 6a. System discards changes, redirects to login (UC-02) with message: "Session expired."   
 - 6b. Customer re-authenticates and restarts update process.   
  
---   
\*(Consistent with UC-01-UC-04 structure; core focus on data mutation and session-aware validation.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Purchase History   
\*\*Use Case ID:\*\* UC-06   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Customer has accessed the account dashboard interface.   
3. At least one completed Order exists in the system (optional, handled in Alternative Flow).   
  
\*\*Postconditions:\*\*   
1. Purchase History linked to the Customer is displayed.   
2. Order Confirmation records remain unchanged (read-only operation).   
  
\*\*Main Flow:\*\*   
1. Customer selects "Purchase History" from account dashboard.   
2. System retrieves all Order entities associated with the Customer's ID from Order Confirmation records.   
3. System compiles Purchase History data (date, products, amounts) from linked Order and Product entities.   
4. System displays chronological list of orders with summary details (order ID, date, total).   
5. Customer reviews history and selects "Back" to exit.   
  
\*\*Alternative Flow:\*\*   
1. \*\*No purchase history at Step 2:\*\*   
 - 2a. System detects zero Order entities for Customer.   
 - 2b. System displays "No orders found. Start shopping?" with catalog link.   
2. \*\*Session timeout during retrieval:\*\*   
 - 3a. Authentication expires before Step 3 completion.   
 - 3b. System aborts flow, redirects to login (UC-02) with "Session expired" message.   
3. \*\*Data inconsistency error:\*\*   
 - 3a. System detects corrupted Order/Product linkage (e.g., missing Inventory reference).   
 - 3b. System displays sanitized error: "Partial data unavailable" and logs exception for Administrator.   
  
---   
\*(Consistent with UC-01-UC-05 structure; emphasizes historical data retrieval with session and integrity safeguards.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Payment Information   
\*\*Use Case ID:\*\* UC-07   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Customer has accessed account dashboard or payment settings interface.   
3. At least one valid Payment Information record exists for the Customer.   
  
\*\*Postconditions:\*\*   
1. Payment Information is displayed in read-only format.   
2. Sensitive data (e.g., full card numbers) remains masked per security protocols.   
3. Access audit recorded in Order Confirmation history.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Payment Methods" from account dashboard.   
2. System retrieves Payment Information entities linked to Customer ID.   
3. System masks sensitive fields (e.g., displays only last 4 digits of card numbers).   
4. System displays formatted Payment Information (card type, expiration, billing address).   
5. Customer reviews information and exits interface.   
  
\*\*Alternative Flow:\*\*   
1. \*\*No Payment Information at Step 2:\*\*   
 - 2a. System detects no stored Payment Information.   
 - 2b. Displays "No payment methods found. Add new payment method?" with setup link.   
2. \*\*Session timeout during Steps 2-4:\*\*   
 - 3a. Authentication expires mid-retrieval.   
 - 3b. System aborts process, redirects to login (UC-02) with "Session expired" notification.   
3. \*\*Data decryption failure:\*\*   
 - 2a. System cannot decrypt secured Payment Information fields.   
 - 2b. Displays sanitized error: "Secure data unavailable" and logs incident for Administrator review.   
  
---   
\*(Consistent with UC-01-UC-06 structure; focuses on secure retrieval and session-bound data masking.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Category   
\*\*Use Case ID:\*\* UC-08   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation, storage, and propagation)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated and has administrative privileges.   
2. Category management interface is accessible.   
  
\*\*Postconditions:\*\*   
1. New category is created OR existing category is updated/deleted in the system.   
2. Changes propagate to linked Product and Inventory entities if applicable.   
3. Category list is refreshed with current state.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Categories" from admin dashboard.   
2. System displays category list with action options (Add/Edit/Delete).   
3. Administrator chooses "Add Category".   
4. System displays a form for category details (name, description).   
5. Administrator enters required fields and submits.   
6. System validates data (e.g., non-empty name, unique name).   
7. System creates new Category entity and links to global inventory.   
8. System confirms success and updates category list.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Edit existing category at Step 3:\*\*   
 - 3a. Administrator selects a category and clicks "Edit".   
 - 3b. System loads category details into editable form.   
 - 3c. Administrator modifies data and submits.   
 - 3d. System validates and updates Category entity.   
 - 3e. Changes propagate to linked Product entities.   
2. \*\*Delete category at Step 3:\*\*   
 - 3a. Administrator selects category and clicks "Delete".   
 - 3b. System verifies no linked Product entities exist.   
 - 3c. System removes Category and updates inventory relationships.   
 - 3d. Deletion confirmed in category list.   
3. \*\*Validation failure at Step 6 (Add/Edit):\*\*   
 - 6a. System rejects submission, highlights errors (e.g., duplicate name).   
 - 6b. Administrator corrects data and resubmits.   
4. \*\*Linked products during deletion (Step 3b):\*\*   
 - 3b.1. System detects products associated with category.   
 - 3b.2. Aborts deletion, displays: "Category cannot be deleted while linked to products".   
  
---   
\*(Consistent with prior UC structures; core flows emphasize CRUD operations with integrity checks.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Product   
\*\*Use Case ID:\*\* UC-09   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation, storage, and inventory updates)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Product management interface is accessible.   
3. At least one Category exists in the system (for product categorization).   
  
\*\*Postconditions:\*\*   
1. New product is created OR existing product is updated/deleted.   
2. Inventory entity is created/updated for stock tracking.   
3. Product list reflects current state across all interfaces.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Products" from admin dashboard.   
2. System displays product list with actions (Add/Edit/Delete).   
3. Administrator chooses "Add Product".   
4. System displays form with fields: name, description, price, category, initial stock.   
5. Administrator enters data and submits.   
6. System validates data (e.g., price ≥0, stock ≥0, category exists).   
7. System creates Product entity, links to Category, and generates associated Inventory record.   
8. System confirms creation and refreshes product list.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Edit existing product at Step 3:\*\*   
 - 3a. Administrator selects product and clicks "Edit".   
 - 3b. System loads product details and linked Inventory data into form.   
 - 3c. Administrator modifies fields (e.g., price, description) and submits.   
 - 3d. System validates and updates Product/Inventory entities.   
2. \*\*Delete product at Step 3:\*\*   
 - 3a. Administrator selects product and clicks "Delete".   
 - 3b. System verifies no active Cart Items or Order links exist.   
 - 3c. System removes Product and linked Inventory/Category associations.   
 - 3d. Deletion confirmed in product list.   
3. \*\*Validation failure at Step 6 (Add/Edit):\*\*   
 - 6a. System rejects submission, highlights errors (e.g., negative price).   
 - 6b. Administrator corrects data and resubmits.   
4. \*\*Active references during deletion (Step 3b):\*\*   
 - 3b.1. System detects linked Cart Items or Orders.   
 - 3b.2. Aborts deletion, displays: "Product cannot be deleted while referenced in active transactions".   
  
---   
\*(Consistent with UC-08 structure; core flows emphasize product lifecycle management with inventory/category integrity checks.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Categorize Product   
\*\*Use Case ID:\*\* UC-10   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation and relationship management)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Product and Category management interfaces are accessible.   
3. At least one uncategorized Product exists in the system.   
  
\*\*Postconditions:\*\*   
1. Product entity is linked to specified Category.   
2. Category-product relationships are updated across all dependent entities (Inventory, Shopping Cart).   
3. Product visibility is filtered according to new category in user interfaces.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Products" from admin dashboard.   
2. System displays uncategorized products list with "Assign Category" action.   
3. Administrator selects target product and chooses "Assign Category".   
4. System retrieves active Category entities and displays selection interface.   
5. Administrator selects target Category and confirms.   
6. System validates Category status (active/inactive).   
7. System links Product entity to Category and updates Inventory relationships.   
8. System confirms successful categorization and refreshes product list.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Inactive category at Step 6:\*\*   
 - 6a. System rejects assignment, displays: "Cannot assign to inactive category".   
 - 6b. Administrator selects different category or aborts.   
2. \*\*Concurrent modification conflict:\*\*   
 - 7a. System detects conflicting category update by another administrator.   
 - 7b. System displays: "Category recently modified. Reloading data..." and repeats Step 4.   
3. \*\*Missing prerequisite data:\*\*   
 - 4a. System finds no active categories.   
 - 4b. Aborts flow, displays: "No categories available. Create categories first (UC-08)".   
  
---   
\*(Consistent with UC-08/UC-09 structure; emphasizes category-product relationship management with concurrency handling.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Inventory   
\*\*Use Case ID:\*\* UC-11   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation, storage, and propagation)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Inventory management interface is accessible.   
3. At least one Product exists in the system (per UC-09).   
  
\*\*Postconditions:\*\*   
1. Inventory records are updated for target products.   
2. Stock changes propagate to Shopping Cart and Order Confirmation entities.   
3. System displays real-time inventory status across interfaces.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Inventory" from admin dashboard.   
2. System displays product inventory list (product name, current stock).   
3. Administrator chooses target product and selects "Update Stock".   
4. System displays current stock value in editable field.   
5. Administrator enters new stock quantity and submits.   
6. System validates input (non-negative integer).   
7. System updates Inventory entity linked to Product.   
8. System synchronizes stock status with Shopping Cart items.   
9. System confirms update and refreshes inventory list.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid stock value at Step 6:\*\*   
 - 6a. System rejects submission, displays "Stock must be ≥0".   
 - 6b. Administrator corrects value and resubmits.   
2. \*\*Concurrent stock depletion during update:\*\*   
 - 7a. If current stock drops below Cart Item reservations mid-update.   
 - 7b. System aborts update, displays "Stock conflict: Active cart reservations exist".   
 - 7c. Administrator adjusts stock after reviewing pending orders.   
3. \*\*Product deletion during process:\*\*   
 - 3a. If target product is deleted by another administrator.   
 - 3b. System halts flow, displays "Product no longer exists" and refreshes list.   
  
---   
\*(Consistent with UC-08-UC-10 structure; focuses on stock synchronization and conflict resolution.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Restock Inventory   
\*\*Use Case ID:\*\* UC-12   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation, storage, and propagation)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Inventory management interface is accessible (per UC-11).   
3. Target Product exists in the system and has an associated Inventory record.   
  
\*\*Postconditions:\*\*   
1. Inventory stock level for the target product is increased.   
2. Updated stock propagates to Shopping Cart availability checks.   
3. Restock audit trail is recorded in system logs.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Restock" action on target Inventory record.   
2. System displays current stock level and restock input field.   
3. Administrator enters positive restock quantity and confirms.   
4. System validates input (positive integer ≤ maximum threshold).   
5. System updates Inventory entity: new\_stock = current\_stock + restock\_quantity.   
6. System syncs availability to linked Shopping Cart items.   
7. System confirms "Inventory restocked successfully" and refreshes data.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid quantity at Step 4:\*\*   
 - 4a. System rejects input, displays "Quantity must be 1-9999".   
 - 4b. Administrator corrects value and resubmits.   
2. \*\*Product discontinuation during restock:\*\*   
 - 5a. System detects Product marked as "discontinued".   
 - 5b. Aborts update, displays "Cannot restock discontinued products".   
3. \*\*Concurrent stock depletion:\*\*   
 - 5a. If current stock drops below safety threshold mid-process.   
 - 5b. System overrides with warning: "Low stock during restock - added safety buffer".   
 - 5c. Final stock = (current\_stock + restock\_quantity) + safety\_buffer.   
  
---   
\*(Consistent with UC-11 structure; emphasizes stock replenishment with conflict resolution.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add to Shopping Cart   
\*\*Use Case ID:\*\* UC-13   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation and cart management)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Customer is viewing a valid Product in the catalog.   
3. Target Product exists, is active, and has available Inventory stock.   
  
\*\*Postconditions:\*\*   
1. New Cart Item is created or existing Cart Item quantity is incremented.   
2. Shopping Cart's total items and price are recalculated.   
3. Product-Inventory linkage is preserved for stock validation.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Add to Cart" for a Product.   
2. System verifies Product status (active) and available Inventory stock.   
3. System retrieves Customer's Shopping Cart.   
4. If Product exists in Cart:   
 - 4a. Increment existing Cart Item quantity by 1.   
5. Else:   
 - 5a. Create new Cart Item linked to Product (quantity=1).   
6. Update Shopping Cart totals (item count and price).   
7. System displays: "Product added to cart".   
  
\*\*Alternative Flow:\*\*   
1. \*\*Product validation failure at Step 2:\*\*   
 - 2a. System displays "Product unavailable" and terminates process.   
2. \*\*Insufficient stock at Step 2:\*\*   
 - 2a. If stock=0: System displays "Out of stock".   
 - 2b. If stock>0 but less than requested:   
 - Cap Cart Item quantity at current stock.   
 - Display: "Only [X] units available. Cart updated".   
3. \*\*Session expiration during process:\*\*   
 - 3a. System redirects to login (UC-02) with "Session expired" notification.   
  
---   
\*(Consistent with UC-01-UC-12 structure; emphasizes cart-inventory synchronization with real-time validation.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Shopping Cart   
\*\*Use Case ID:\*\* UC-14   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Shopping Cart exists for the Customer (created during registration, UC-01).   
3. Customer accesses the cart interface via navigation element.   
  
\*\*Postconditions:\*\*   
1. Shopping Cart contents are displayed with real-time item details.   
2. Cart Item linkages to Product/Inventory remain unchanged (read-only operation).   
3. System records cart view event in session logs.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Shopping Cart" icon or menu option.   
2. System retrieves Shopping Cart entity and linked Cart Items.   
3. For each Cart Item, System fetches current Product details (name, price) and Inventory status.   
4. System calculates subtotal, taxes, and grand total.   
5. System displays formatted cart contents with itemized list and summary.   
6. Customer reviews cart and selects next action (e.g., continue shopping or checkout).   
  
\*\*Alternative Flow:\*\*   
1. \*\*Empty cart at Step 2:\*\*   
 - 2a. System detects zero Cart Items.   
 - 2b. Displays "Your cart is empty. Browse products?" with catalog link.   
2. \*\*Product unavailable at Step 3:\*\*   
 - 3a. System identifies deactivated Product or zero Inventory stock.   
 - 3b. Flags affected Cart Item as "Unavailable" with removal option.   
3. \*\*Session expiration during retrieval:\*\*   
 - 3a. Authentication fails mid-process.   
 - 3b. System aborts flow, redirects to login (UC-02) with "Session expired" prompt.   
  
---   
\*(Consistent with UC-01-UC-13 structure; emphasizes real-time cart-inventory synchronization and graceful error handling.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Modify Shopping Cart   
\*\*Use Case ID:\*\* UC-15   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation and cart updates)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (UC-02).   
2. Shopping Cart contains at least one Cart Item (UC-13/UC-14).   
3. Customer accesses the Shopping Cart interface (UC-14).   
  
\*\*Postconditions:\*\*   
1. Cart Item quantity is updated or item is removed.   
2. Shopping Cart totals are recalculated.   
3. Inventory availability is revalidated for modified items.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Edit Cart" on Shopping Cart interface.   
2. System displays Cart Items with editable quantity fields and "Remove" options.   
3. Customer modifies item quantity or selects "Remove" for target Cart Item.   
4. For quantity changes:   
 - 4a. System validates new quantity (≥1 and ≤ available Inventory stock).   
 - 4b. Updates Cart Item quantity.   
5. For removals:   
 - 5a. System deletes target Cart Item.   
6. System recalculates Shopping Cart totals (item count and price).   
7. System confirms: "Cart updated successfully" and displays revised cart.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid quantity at Step 4a:\*\*   
 - 4a.1. System rejects input, displays "Quantity must be 1-[current stock]".   
 - 4a.2. Customer re-enters valid quantity or cancels.   
2. \*\*Item stock depletion during edit:\*\*   
 - 4a.1. Available stock drops below requested quantity mid-process.   
 - 4a.2. System auto-adjusts to max available stock and notifies: "Only [X] units left. Quantity adjusted."   
3. \*\*Product deactivation during edit:\*\*   
 - 3a. System detects target Product is now inactive.   
 - 3b. Removes item automatically and displays: "[Product] discontinued. Item removed from cart."   
4. \*\*Session expiration:\*\*   
 - 3a. Redirects to login (UC-02) with "Session expired" notification.   
  
---   
\*(Consistent with UC-13/UC-14 structure; emphasizes real-time inventory-cart synchronization and atomic modifications.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Clear Shopping Cart   
\*\*Use Case ID:\*\* UC-16   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for cart clearance and state reset)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Shopping Cart contains at least one Cart Item (verified via UC-14).   
3. Customer accesses Shopping Cart interface (UC-14) or checkout page.   
  
\*\*Postconditions:\*\*   
1. All Cart Item entities linked to the Shopping Cart are deleted.   
2. Shopping Cart totals reset to zero (item count = 0, price = 0.00).   
3. Inventory stock reservations for deleted Cart Items are released.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Clear Cart" action in Shopping Cart interface.   
2. System verifies Cart Item existence and Inventory linkage.   
3. System deletes all Cart Item entities associated with the Shopping Cart.   
4. System resets Shopping Cart metrics (item count, total price).   
5. System releases reserved Inventory stock for deleted items.   
6. System confirms: "Shopping cart cleared successfully."   
  
\*\*Alternative Flow:\*\*   
1. \*\*Empty cart at Step 2:\*\*   
 - 2a. System detects zero Cart Items.   
 - 2b. Displays: "Cart is already empty" and skips clearance.   
2. \*\*Session expiration during clearance:\*\*   
 - 3a. Authentication fails mid-process.   
 - 3b. System aborts operation, redirects to login (UC-02) with "Session expired".   
3. \*\*Inventory sync failure at Step 5:\*\*   
 - 5a. System cannot release reserved stock (e.g., database conflict).   
 - 5b. Flags error: "Cart cleared, but inventory update delayed" and logs issue for Administrator.   
  
---   
\*(Consistent with UC-13-UC-15 structure; emphasizes atomic cart/inventory state reset with conflict handling.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Confirm Checkout   
\*\*Use Case ID:\*\* UC-17   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation, payment processing, and order creation)   
- Payment Gateway (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (UC-02).   
2. Shopping Cart contains ≥1 valid Cart Item (UC-13, UC-14).   
3. All Cart Items have real-time verified Inventory availability.   
4. Payment Information is stored or provided during checkout.   
  
\*\*Postconditions:\*\*   
1. Order entity is created with status "Confirmed".   
2. Order Confirmation record is generated and stored.   
3. Inventory stock is decremented for purchased items.   
4. Shopping Cart is cleared (all Cart Items removed).   
5. Payment transaction is recorded in Payment Information.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Checkout" from Shopping Cart interface (UC-14).   
2. System verifies Cart Item validity (active products, sufficient stock).   
3. System displays order summary (items, totals) and requests payment confirmation.   
4. Customer confirms payment using stored Payment Information or enters new details.   
5. System sends payment authorization to Payment Gateway via API Specifications.   
6. Payment Gateway returns transaction success.   
7. System creates Order entity linked to Customer and Cart Items.   
8. System decrements Inventory stock for each purchased item.   
9. System clears Shopping Cart (deletes all Cart Items).   
10. System generates Order Confirmation with details.   
11. System displays Order Confirmation to Customer.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Cart validation failure at Step 2:\*\*   
 - 2a. System detects invalid item (e.g., out-of-stock).   
 - 2b. Removes item automatically and notifies: "[Product] unavailable. Cart updated."   
 - 2c. Restarts Main Flow at Step 2.   
2. \*\*Payment failure at Step 6:\*\*   
 - 6a. Payment Gateway declines transaction.   
 - 6b. System notifies: "Payment declined. Update method or retry."   
 - 6c. Customer updates Payment Information and returns to Step 4.   
3. \*\*Session expiration during Steps 1-10:\*\*   
 - 3a. System aborts checkout, preserves Cart contents.   
 - 3b. Redirects to login (UC-02) with "Session expired. Resume checkout after login."   
4. \*\*Inventory conflict at Step 8:\*\*   
 - 8a. Stock depleted mid-checkout (e.g., concurrent purchase).   
 - 8b. System rolls back transaction, notifies: "Stock changed. Review cart."   
 - 8c. Returns Customer to Shopping Cart (UC-14).   
  
---   
\*(Consistent with UC-13-UC-16 structure; focuses on atomic transaction integrity and payment-inventory-cart synchronization.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Cancel Checkout   
\*\*Use Case ID:\*\* UC-18   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for process termination and state preservation)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (UC-02).   
2. Checkout process has been initiated (per UC-17).   
3. No Order Confirmation record exists yet.   
  
\*\*Postconditions:\*\*   
1. Checkout process is terminated without order creation.   
2. Shopping Cart remains fully preserved with all Cart Items.   
3. Inventory stock reservations remain unchanged.   
4. Customer is returned to Shopping Cart interface (UC-14).   
  
\*\*Main Flow:\*\*   
1. Customer selects "Cancel Checkout" during checkout process.   
2. System verifies no payment authorization was sent to Payment Gateway.   
3. System discards all temporary checkout data (e.g., shipping address selections).   
4. System preserves Shopping Cart state and linked Cart Items.   
5. System transitions customer back to Shopping Cart view (UC-14).   
6. System displays: "Checkout canceled. Your cart has been preserved."   
  
\*\*Alternative Flow:\*\*   
1. \*\*Partial payment authorization occurred:\*\*   
 - 2a. System detects pending payment authorization.   
 - 2b. Automatically voids transaction via Payment Gateway API.   
 - 2c. Proceeds to Step 3 with notification: "Pending payment canceled."   
2. \*\*Concurrent system modification:\*\*   
 - 4a. If Cart Items modified by another process (e.g., stock depletion by other customer).   
 - 4b. System preserves current valid state and flags conflicts: "Cart updated during checkout."   
3. \*\*Session expiration during cancellation:\*\*   
 - 1a. Authentication fails before completion.   
 - 1b. System preserves cart state, redirects to login (UC-02) with "Session expired. Cart saved."   
  
---   
\*(Consistent with UC-17 structure; emphasizes atomic termination and state preservation.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Confirm Order   
\*\*Use Case ID:\*\* UC-19   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for order validation and confirmation)   
- Payment Gateway (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (UC-02).   
2. Checkout process completed successfully (per UC-17).   
3. Order entity exists with status "Payment Approved".   
4. Order Confirmation record is initialized but not finalized.   
  
\*\*Postconditions:\*\*   
1. Order Confirmation record is permanently stored with all transaction details.   
2. Order status transitions to "Confirmed".   
3. Inventory stock is committed (irreversible deduction).   
4. Customer receives digital/written confirmation.   
  
\*\*Main Flow:\*\*   
1. System automatically validates Payment Gateway transaction success (from UC-17).   
2. System verifies Inventory stock consistency for all Order items.   
3. System finalizes Order Confirmation record (timestamp, items, totals, payment ID).   
4. System binds Order Confirmation to Customer and Payment Information entities.   
5. System updates Order status to "Confirmed" and locks modifications.   
6. System notifies Customer via UI/email: "Order #[ID] confirmed".   
  
\*\*Alternative Flow:\*\*   
1. \*\*Inventory discrepancy at Step 2:\*\*   
 - 2a. System detects stock depletion below purchased quantity.   
 - 2b. Auto-upgrades shipping to expedited service at no cost.   
 - 2c. Notifies: "Item availability adjusted. Free expedited shipping applied."   
2. \*\*Payment verification failure at Step 1:\*\*   
 - 1a. Payment Gateway reports transaction anomaly.   
 - 1b. System freezes Order, alerts Administrator for manual review.   
 - 1c. Notifies Customer: "Confirmation delayed. Contact support if unresolved in 24h."   
3. \*\*Data corruption during Step 3:\*\*   
 - 3a. System fails to persist Order Confirmation.   
 - 3b. Reverts to last valid checkpoint and retries (max 3 attempts).   
 - 3c. If persistent failure, triggers Plugin-based backup logging.   
  
---   
\*(Consistent with UC-17/UC-18 structure; emphasizes atomic confirmation with inventory-payment-cart synchronization.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Order   
\*\*Use Case ID:\*\* UC-20   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. Target Order exists in the system and is linked to the Customer.   
3. Order Confirmation record is available and accessible.   
  
\*\*Postconditions:\*\*   
1. Order details are displayed in read-only format.   
2. Order status and linked entities remain unchanged.   
3. Access event is recorded in system logs.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Order History" from account dashboard or receives order confirmation link.   
2. Customer chooses target Order ID from the list.   
3. System retrieves Order entity and linked Order Confirmation, Payment Information, and Cart Item details.   
4. System compiles data: order date, items, prices, payment method (masked), shipping address, and status.   
5. System displays formatted order summary with itemized breakdown.   
6. Customer reviews details and exits the interface.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid Order ID at Step 3:\*\*   
 - 3a. System detects no matching Order for the Customer.   
 - 3b. Displays "Order not found" and returns to Order History list.   
2. \*\*Session expiration during retrieval:\*\*   
 - 4a. Authentication fails mid-process.   
 - 4b. System aborts flow, redirects to login (UC-02) with "Session expired. Re-login to view order."   
3. \*\*Data corruption at Step 4:\*\*   
 - 4a. System cannot resolve linked entities (e.g., missing Product details).   
 - 4b. Displays sanitized error: "Partial data unavailable" with support contact option.   
 - 4c. Logs exception for Administrator review.   
  
---   
\*(Consistent with UC-06/UC-17 structure; emphasizes atomic data retrieval with access control and error resilience.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Cancel Order   
\*\*Use Case ID:\*\* UC-21   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation, status updates, and refund processing)   
- Payment Gateway (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (UC-02).   
2. Target Order exists with status "Confirmed" or "Processing" (per UC-19).   
3. Order cancellation period is active (system-defined timeframe after confirmation).   
4. No shipment initiation recorded in Order Confirmation.   
  
\*\*Postconditions:\*\*   
1. Order status transitions to "Canceled".   
2. Full refund initiated via Payment Gateway (if payment captured).   
3. Inventory stock restored for canceled items.   
4. Order Confirmation record updated with cancellation timestamp.   
  
\*\*Main Flow:\*\*   
1. Customer selects target Order in Order History interface (UC-20).   
2. System verifies Order eligibility for cancellation (status and timeframe).   
3. Customer selects "Cancel Order" action.   
4. System displays cancellation confirmation prompt.   
5. Customer confirms cancellation request.   
6. System updates Order status to "Canceled".   
7. If payment was captured:   
 - 7a. System initiates refund via Payment Gateway API Specifications.   
8. System restores Inventory stock quantities for all order items.   
9. System notifies Customer: "Order #[ID] canceled successfully. Refund initiated."   
  
\*\*Alternative Flow:\*\*   
1. \*\*Ineligible Order at Step 2:\*\*   
 - 2a. System blocks cancellation (e.g., order shipped/cancellation expired).   
 - 2b. Displays: "Cancellation unavailable. Contact support."   
2. \*\*Refund failure at Step 7a:\*\*   
 - 7a.1. Payment Gateway declines refund request.   
 - 7a.2. System retries 3x; if persistent failure, flags Administrator intervention.   
 - 7a.3. Notifies Customer: "Cancellation complete. Refund delayed."   
3. \*\*Inventory restoration conflict at Step 8:\*\*   
 - 8a. System detects concurrent stock modification.   
 - 8b. Overrides with max available capacity and logs discrepancy.   
4. \*\*Customer aborts at Step 5:\*\*   
 - 5a. Customer selects "Cancel Action".   
 - 5b. System returns to Order details view without changes.   
  
---   
\*(Consistent with UC-17-UC-20 structure; emphasizes atomic cancellation with refund-inventory synchronization.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Send Order Confirmation Email   
\*\*Use Case ID:\*\* UC-22   
\*\*Actors:\*\*   
- System (Primary, for email generation and dispatch)   
- Customer (Secondary, as recipient)   
- Email Service Plugin (External, via Plugin/API Specifications)   
  
\*\*Preconditions:\*\*   
1. Order Confirmation record is finalized (per UC-19).   
2. Customer's email address is valid and stored in Personal Information.   
3. Email Service Plugin is operational and authenticated.   
  
\*\*Postconditions:\*\*   
1. Order Confirmation email is delivered to Customer.   
2. Email dispatch status is logged in Order Confirmation entity.   
3. Audit trail recorded in system logs.   
  
\*\*Main Flow:\*\*   
1. System triggers email process upon Order Confirmation finalization (UC-19 Step 6).   
2. System retrieves Customer's email from Personal Information entity.   
3. System compiles email content using Order Confirmation data (order ID, items, totals).   
4. System invokes Email Service Plugin via API Specifications with payload.   
5. Email Service Plugin delivers email to Customer.   
6. System receives delivery success confirmation.   
7. System updates Order Confirmation record: "Email dispatched at [timestamp]".   
  
\*\*Alternative Flow:\*\*   
1. \*\*Invalid email at Step 2:\*\*   
 - 2a. System detects malformed/empty email address.   
 - 2b. Logs "Email invalid: Order #[ID]" and halts process.   
 - 2c. Flags Administrator alert via system dashboard.   
2. \*\*Plugin failure at Step 4:\*\*   
 - 4a. Email Service Plugin returns API error (e.g., authentication failure).   
 - 4b. System retries 3x at 5-minute intervals; if persistent failure, logs "Email service unavailable".   
3. \*\*Delivery failure at Step 5:\*\*   
 - 5a. Email Service Plugin reports undeliverable (e.g., bounced).   
 - 5b. System stores fallback PDF in Order Confirmation record.   
 - 5c. Displays in Customer UI: "View confirmation PDF" with download option.   
  
---   
\*(Consistent with prior UC structures; emphasizes atomic email generation with fail-safe mechanisms.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Install Plugin   
\*\*Use Case ID:\*\* UC-23   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation and integration)   
- Plugin Repository (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Plugin management interface is accessible.   
3. Target plugin is available in the Plugin Repository.   
  
\*\*Postconditions:\*\*   
1. Plugin is successfully integrated into the system.   
2. Plugin functionality is enabled per API Specifications.   
3. System configuration updated to reflect new capabilities.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Plugin Management" from admin dashboard.   
2. System retrieves available plugins from Plugin Repository via API Specifications.   
3. Administrator selects target plugin and initiates "Install" action.   
4. System verifies plugin compatibility with current system version.   
5. System downloads and integrates plugin components.   
6. System validates functionality against API Specifications.   
7. System enables plugin and updates configuration registry.   
8. System confirms: "Plugin installed successfully" with activation status.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Compatibility failure at Step 4:\*\*   
 - 4a. System detects version conflict.   
 - 4b. Aborts installation, displays: "Incompatible with system version [X]. Requires [Y]."   
2. \*\*Dependency missing at Step 5:\*\*   
 - 5a. Plugin requires additional components not present.   
 - 5b. System auto-installs dependencies via Plugin Repository.   
 - 5c. Resumes Main Flow at Step 6.   
3. \*\*Validation failure at Step 6:\*\*   
 - 6a. Plugin violates API Specifications.   
 - 6b. System quarantines plugin, notifies: "Security violation. Installation blocked."   
 - 6c. Logs incident for Administrator review.   
  
---   
\*(Consistent with UC-08/UC-22 structure; emphasizes atomic integration with security and dependency handling.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Uninstall Plugin   
\*\*Use Case ID:\*\* UC-24   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation and removal)   
- Plugin Repository (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Target plugin is installed and active in the system (verified via Plugin Management interface).   
3. Plugin management interface is accessible.   
  
\*\*Postconditions:\*\*   
1. Plugin is completely removed from the system.   
2. Plugin functionality is disabled per API Specifications.   
3. System configuration reverts to pre-installation state.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Plugin Management" from admin dashboard.   
2. System displays installed plugins with "Uninstall" options.   
3. Administrator selects target plugin and initiates "Uninstall" action.   
4. System verifies no active dependencies (e.g., pending API calls or processes).   
5. System disables plugin functionality and removes all components.   
6. System updates configuration registry and clears plugin-related caches.   
7. System notifies Plugin Repository via API Specifications.   
8. System confirms: "Plugin uninstalled successfully".   
  
\*\*Alternative Flow:\*\*   
1. \*\*Active dependencies at Step 4:\*\*   
 - 4a. System detects running processes linked to the plugin.   
 - 4b. Aborts uninstallation, displays: "Active dependencies exist. Retry after [X] minutes".   
2. \*\*Partial removal failure at Step 5:\*\*   
 - 5a. System fails to delete critical components (e.g., locked files).   
 - 5b. Auto-quarantines residual files, notifies: "Partial removal. Manual cleanup required".   
 - 5c. Logs error for Administrator intervention.   
3. \*\*Configuration rollback failure at Step 6:\*\*   
 - 6a. System cannot restore pre-installation settings.   
 - 6b. Preserves safe fallback state and alerts: "Config reset incomplete. Verify system stability".   
  
---   
\*(Consistent with UC-23 structure; emphasizes atomic removal with dependency and integrity checks.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Plugin Specifications   
\*\*Use Case ID:\*\* UC-25   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for data retrieval and display)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Plugin Management interface is accessible.   
3. At least one plugin is installed in the system (per UC-23).   
  
\*\*Postconditions:\*\*   
1. API Specifications for the target plugin are displayed in structured format.   
2. No system modifications occur (read-only operation).   
3. Access event is logged in system audit trails.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Plugin Management" from admin dashboard.   
2. System displays list of installed plugins with "View Specifications" actions.   
3. Administrator selects target plugin and chooses "View Specifications".   
4. System retrieves the plugin's API Specifications from secured storage.   
5. System parses specifications into structured format (endpoints, methods, parameters).   
6. System displays API documentation with syntax highlighting and expandable sections.   
7. Administrator reviews specifications and closes the view.   
  
\*\*Alternative Flow:\*\*   
1. \*\*No installed plugins at Step 2:\*\*   
 - 2a. System detects zero installed plugins.   
 - 2b. Displays: "No plugins available. Install plugins first (UC-23)."   
2. \*\*Specifications retrieval failure at Step 4:\*\*   
 - 4a. System cannot locate valid API Specifications.   
 - 4b. Displays sanitized error: "Specifications unavailable" with technical support contact.   
 - 4c. Logs exception for plugin maintainer review.   
3. \*\*Session expiration during retrieval:\*\*   
 - 5a. Administrator authentication expires mid-process.   
 - 5b. System preserves partial data and redirects to login (UC-02) with "Session expired" prompt.   
  
---   
\*(Consistent with UC-23/UC-24 structure; emphasizes secure API documentation retrieval with integrity checks.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Customer   
\*\*Use Case ID:\*\* UC-26   
\*\*Actors:\*\*   
- Administrator (Primary)   
- System (Secondary, for validation, storage, and propagation)   
  
\*\*Preconditions:\*\*   
1. Administrator is authenticated with administrative privileges (per UC-02 variant).   
2. Customer management interface is accessible.   
3. Personal Information entity exists for target customers (for modification scenarios).   
  
\*\*Postconditions:\*\*   
1. New Customer entity is created OR existing Customer status/attributes are updated.   
2. Changes propagate to linked entities (Personal Information, Shopping Cart, Orders).   
3. Audit trail is recorded in system logs.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Customers" from admin dashboard.   
2. System displays customer list with action options (Add/Edit/Deactivate).   
3. Administrator chooses "Add Customer".   
4. System displays form with fields: name, email, password, status (active/inactive).   
5. Administrator enters data and submits.   
6. System validates format and uniqueness of email.   
7. System creates Customer entity with default Personal Information placeholder.   
8. System auto-generates Shopping Cart and confirms: "Customer added successfully".   
  
\*\*Alternative Flow:\*\*   
1. \*\*Edit customer at Step 3:\*\*   
 - 3a. Administrator selects customer and clicks "Edit".   
 - 3b. System loads editable details (email, status) excluding password.   
 - 3c. Administrator modifies fields and submits.   
 - 3d. System validates email uniqueness and updates Customer entity.   
 - 3e. Propagates changes to linked Order Confirmation records.   
2. \*\*Deactivate customer at Step 3:\*\*   
 - 3a. Administrator selects "Deactivate" on active customer.   
 - 3b. System verifies no pending orders or checkout processes.   
 - 3c. Updates Customer status to "inactive" and revokes authentication tokens.   
 - 3d. Preserves Shopping Cart for future reactivation.   
3. \*\*Validation failure at Step 6:\*\*   
 - 6a. Duplicate email detected: Aborts creation, displays "Email already registered".   
 - 6b. Invalid format: Highlights field-specific errors (e.g., invalid email syntax).   
4. \*\*Active transactions during deactivation (Step 3b):\*\*   
 - 3b.1. System blocks deactivation for pending checkouts/orders.   
 - 3b.2. Displays: "Cannot deactivate with active transactions. Resolve orders first."   
  
---   
\*(Consistent with UC-08/UC-09 structure; emphasizes customer lifecycle control with transaction integrity checks.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Payment Information   
\*\*Use Case ID:\*\* UC-27   
\*\*Actors:\*\*   
- Customer (Primary)   
- System (Secondary, for validation, security, and data storage)   
- Payment Gateway (External, via API Specifications)   
  
\*\*Preconditions:\*\*   
1. Customer is authenticated with an active session (per UC-02).   
2. At least one valid Payment Information record exists for the Customer (verified via UC-07).   
3. Customer accesses the Payment Methods interface (UC-07).   
  
\*\*Postconditions:\*\*   
1. Target Payment Information entity is updated with new details.   
2. Sensitive data (e.g., card numbers) remains encrypted per security protocols.   
3. Updated payment details propagate to future Order Confirmation records.   
4. System displays success confirmation with masked verification.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Edit" on target Payment Method in Payment Methods interface.   
2. System displays editable form with current masked details (e.g., ●●●●1234) and security fields.   
3. Customer updates fields (e.g., expiration date, billing address, card verification code).   
4. Customer submits changes.   
5. System validates data format (e.g., future expiration date, valid CVV length).   
6. System sends new details to Payment Gateway via API Specifications for live verification.   
7. Payment Gateway returns validation success.   
8. System encrypts and updates Payment Information entity.   
9. System confirms: "Payment method updated" with last 4 digits and expiration.   
  
\*\*Alternative Flow:\*\*   
1. \*\*Validation failure at Step 5:\*\*   
 - 5a. System rejects submission, highlights errors (e.g., "Invalid expiration date").   
 - 5b. Customer corrects data and resubmits.   
2. \*\*Payment Gateway rejection at Step 7:\*\*   
 - 7a. Payment Gateway reports invalid card (e.g., expired/blocked).   
 - 7b. System displays: "Verification failed. Check details or use another method".   
 - 7c. Preserves original payment details unchanged.   
3. \*\*Duplicate card detection:\*\*   
 - 6a. System identifies identical card details across Customer's records.   
 - 6b. Aborts update, displays: "Duplicate payment method detected".   
4. \*\*Session expiration during Steps 1-8:\*\*   
 - 4a. Authentication fails mid-process.   
 - 4b. System discards changes, redirects to login (UC-02) with "Session expired. Resume update after login".   
5. \*\*Concurrent modification conflict:\*\*   
 - 8a. System detects simultaneous update by another session (e.g., mobile/desktop).   
 - 8b. Preserves latest valid version and notifies: "Payment method updated elsewhere".   
  
---   
\*(Consistent with UC-07/UC-17 structure; emphasizes atomic updates with real-time payment gateway validation and security compliance.)\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Administrator   
\*\*Use Case ID:\*\* UC-28   
\*\*Actors:\*\*   
- Super Administrator (Primary)   
- System (Secondary, for validation, storage, and access control)   
  
\*\*Preconditions:\*\*   
1. Super Administrator is authenticated with elevated privileges (distinct from standard Administrator privileges).   
2. Administrator management interface is accessible via admin dashboard.   
3. At least one Administrator entity exists in the system (for modification/deletion scenarios).   
  
\*\*Postconditions:\*\*   
1. New Administrator entity is created OR existing Administrator attributes/privileges are updated OR Administrator is deactivated.   
2. Access control lists and authentication tokens are synchronized with changes.   
3. Audit trail is recorded in system logs with Super Administrator action stamp.   
  
\*\*Main Flow:\*\*   
1. Super Administrator selects "Manage Administrators" from admin dashboard.   
2. System displays Administrator list with action options (Add/Edit/Deactivate).   
3. Super Administrator chooses "Add Administrator".   
4. System displays form with fields: username, email, privilege level (e.g., Inventory-only, Full-access), and temporary password.   
5. Super Administrator enters data and submits.   
6. System validates uniqueness of username/email and password strength.   
7. System creates Administrator entity, encrypts credentials, and assigns privileges.   
8. System sends activation email to new Administrator with setup instructions.   
9. System confirms: "Administrator added. Activation email sent."   
  
\*\*Alternative Flow:\*\*   
1. \*\*Edit Administrator at Step 3:\*\*   
 - 3a. Super Administrator selects target Administrator and clicks "Edit".   
 - 3b. System loads current attributes (excluding password) into editable form.   
 - 3c. Super Administrator modifies privilege level or contact details and submits.   
 - 3d. System validates changes and updates Administrator entity.   
 - 3e. Propagates privilege updates to all active sessions.   
2. \*\*Deactivate Administrator at Step 3:\*\*   
 - 3a. Super Administrator selects "Deactivate" on active Administrator.   
 - 3b. System verifies no pending critical operations (e.g., inventory restocks in progress).   
 - 3c. Revokes authentication tokens and updates status to "inactive".   
 - 3d. Preserves audit records linked to deactivated Administrator.   
3. \*\*Validation Failure at Step 6 (Add/Edit):\*\*   
 - 6a. Duplicate username/email detected: System rejects submission with "Identifier already in use".   
 - 6b. Weak password: Enforces complexity rules and prompts regeneration.   
4. \*\*Self-Deactivation Attempt:\*\*   
 - 3a. Super Administrator tries to deactivate own account.   
 - 3b. System blocks action and displays: "Cannot deactivate your own account".   
5. \*\*Concurrent Session Conflict:\*\*   
 - 7a. System detects privilege modification during target Administrator's active session.   
 - 7b. Forces session termination and notifies: "Permissions updated. Re-login required".   
  
---