项目文档

# Functional Requirement

### Chapter 1: Functional Requirements   
  
\*\*1.1 Customer Registration Function\*\*   
Function ID: FR-01   
Description: Enables unregistered users to create new accounts with mandatory personal information and email verification.   
Input: Customer details (Name, Email, Password, PhoneNumber, Address).   
Output: New customer record stored in the database with unique CustomerID and encrypted credentials.   
  
\*\*1.2 Customer Authentication Function\*\*   
Function ID: FR-02   
Description: Validates registered customer credentials to grant system access.   
Input: Email and Password.   
Output: Authenticated session with permission tokens; login timestamp recorded in audit log.   
  
\*\*1.3 Customer Profile Management Function\*\*   
Function ID: FR-03   
Description: Allows authenticated customers to modify personal details (contact, address).   
Input: Updated customer attributes (PhoneNumber, Address).   
Output: Persisted database changes; audit log entry with modification timestamp.   
  
\*\*1.4 Admin Customer Management Function\*\*   
Function ID: FR-04   
Description: Enables administrators to create, update, or deactivate customer accounts.   
Input: Administrator actions (add/edit/deactivate), customer attributes.   
Output: Modified customer records; audit log with admin ID and action details.   
  
\*\*1.5 Product Catalog Management Function\*\*   
Function ID: FR-05   
Description: Allows administrators to add, edit, or delete products with category assignments.   
Input: Product attributes (Name, Description, Price, StockQuantity, CategoryID).   
Output: Updated product records; inventory synchronization; audit log entries.   
  
\*\*1.6 Inventory Adjustment Function\*\*   
Function ID: FR-06   
Description: Modifies stock quantities and triggers reorder alerts.   
Input: ProductID, new StockQuantity, adjustment reason.   
Output: Updated inventory data; low-stock notifications if applicable.   
  
\*\*1.7 Shopping Cart Modification Function\*\*   
Function ID: FR-07   
Description: Handles item additions, quantity updates, and removals in the cart.   
Input: ProductID, new Quantity (or removal flag).   
Output: Updated cart metadata (subtotal, item count); real-time inventory validation.   
  
\*\*1.8 Order Placement Function\*\*   
Function ID: FR-08   
Description: Converts cart contents into orders with payment processing.   
Input: Confirmed shipping address, payment method details.   
Output: New Order record; reduced inventory; order confirmation email; cleared cart.   
  
\*\*1.9 Order Tracking Function\*\*   
Function ID: FR-09   
Description: Retrieves real-time shipment status from carrier APIs.   
Input: OrderID.   
Output: Dynamic tracking data (carrier, location, estimated delivery); cached status timeline.   
  
\*\*1.10 Payment Processing Function\*\*   
Function ID: FR-10   
Description: Authorizes transactions and records payment details.   
Input: PaymentMethod (card details), Amount, OrderID.   
Output: Payment record (TransactionID, Status); order status synchronization.   
  
\*\*1.11 Category Hierarchy Management Function\*\*   
Function ID: FR-11   
Description: Manages product category creation, editing, and deletion.   
Input: Category attributes (Name, Description, ParentCategoryID).   
Output: Updated category tree; synchronized product-category associations.   
  
\*\*1.12 Admin Order Management Function\*\*   
Function ID: FR-12   
Description: Updates order statuses, processes cancellations/refunds.   
Input: OrderID, new OrderStatus, refund amounts.   
Output: Modified Order and Payment records; inventory restocks on cancellations.   
  
\*\*1.13 System Configuration Function\*\*   
Function ID: FR-13   
Description: Adjusts global settings (security, notifications, UI).   
Input: Configuration parameters (session timeout, email templates).   
Output: Persisted system-wide changes; audit log entries.   
  
\*\*1.14 Audit Logging Function\*\*   
Function ID: FR-14   
Description: Automatically records critical actions (logins, data modifications).   
Input: Actor type (Customer/Admin), action type, timestamp.   
Output: Immutable audit log entries stored in the database.   
  
\*\*1.15 Plugin API Integration Function\*\*   
Function ID: FR-15   
Description: Provides interfaces for third-party functionality extensions.   
Input: API requests compliant with documented specifications.   
Output: Executed plugin operations; system state modifications.   
  
\*\*1.16 Data Validation Function\*\*   
Function ID: FR-16   
Description: Enforces input constraints (email format, password strength).   
Input: User-provided data fields.   
Output: Validation errors or sanitized inputs for database operations.   
  
\*\*1.17 Email Notification Function\*\*   
Function ID: FR-17   
Description: Sends transactional emails (order confirmations, password resets).   
Input: Recipient email, template type, dynamic content.   
Output: Delivered emails; delivery status logs.   
  
\*\*1.18 Session Management Function\*\*   
Function ID: FR-18   
Description: Handles session creation/timeout for authenticated users.   
Input: Login credentials or session tokens.   
Output: Active session establishment/termination; timeout redirects.   
  
\*\*1.19 Dependency Validation Function\*\*   
Function ID: FR-19   
Description: Prevents operations violating data integrity (e.g., deleting products in active orders).   
Input: Entity ID (ProductID/CategoryID), action type.   
Output: Blocked action alert or dependency resolution prompts.   
  
\*\*1.20 Conflict Resolution Function\*\*   
Function ID: FR-20   
Description: Manages concurrent data editing conflicts.   
Input: Conflicting data versions from multiple users.   
Output: Merged valid changes; conflict notifications with latest data.   
  
---  
\*\*Compliance Verification\*\*   
- All inputs/outputs reference defined entities/attributes from the provided data model.   
- Functions map directly to use cases (e.g., FR-08 covers UC-12/UC-22; FR-12 covers UC-14).   
- Feasibility ensured: Inputs contain sufficient data for transformations (e.g., address → shipping cost calculation).   
- No undefined terms used; all references rooted in system description/data model.

# External Description

### Chapter 2: External Interfaces  
  
This chapter documents all external interfaces of the system, categorizing them into user interfaces, hardware interfaces, software interfaces, and communication interfaces. Each interface is defined with its role, interaction method, and relevant inputs/outputs.  
  
---  
  
#### \*\*2.1 User Interfaces\*\*  
  
\*\*2.1.1 Customer Registration Screen\*\*  
- \*\*Description:\*\* A form for unregistered users to input personal details.  
- \*\*Inputs:\*\* Name, Email, Password, PhoneNumber, Address.  
- \*\*Outputs:\*\* Confirmation message or error notifications.  
  
\*\*2.1.2 Customer Login Screen\*\*  
- \*\*Description:\*\* Interface for user authentication.  
- \*\*Inputs:\*\* Email, Password.  
- \*\*Outputs:\*\* Session token or error message.  
  
\*\*2.1.3 Customer Profile Screen\*\*  
- \*\*Description:\*\* Allows users to view and update personal details.  
- \*\*Inputs:\*\* Updated PhoneNumber, Address.  
- \*\*Outputs:\*\* Updated profile information or success message.  
  
\*\*2.1.4 Shopping Cart Screen\*\*  
- \*\*Description:\*\* Displays cart contents and allows modifications.  
- \*\*Inputs:\*\* ProductID, Quantity adjustments.  
- \*\*Outputs:\*\* Updated cart subtotal and item count.  
  
\*\*2.1.5 Order Placement Screen\*\*  
- \*\*Description:\*\* Interface for placing orders.  
- \*\*Inputs:\*\* Confirmed shipping address, payment method.  
- \*\*Outputs:\*\* Order confirmation message or error.  
  
\*\*2.1.6 Order Tracking Screen\*\*  
- \*\*Description:\*\* Shows real-time shipment status.  
- \*\*Inputs:\*\* OrderID.  
- \*\*Outputs:\*\* Carrier, location, estimated delivery.  
  
\*\*2.1.7 Data Validation Errors\*\*  
- \*\*Description:\*\* Displays validation errors.  
- \*\*Inputs:\*\* User-provided data.  
- \*\*Outputs:\*\* Error messages indicating issues.  
  
\*\*2.1.8 Session Timeout Notification\*\*  
- \*\*Description:\*\* Alerts user of session expiration.  
- \*\*Inputs:\*\* Session token.  
- \*\*Outputs:\*\* Logout prompt or redirect.  
  
\*\*2.1.9 Dependency Alert\*\*  
- \*\*Description:\*\* Warns of data dependencies.  
- \*\*Inputs:\*\* Entity ID, action type.  
- \*\*Outputs:\*\* Alert message or prompt.  
  
\*\*2.1.10 Conflict Notification\*\*  
- \*\*Description:\*\* Notifies of data conflicts.  
- \*\*Inputs:\*\* Conflicting data versions.  
- \*\*Outputs:\*\* Notification message or resolved data.  
  
---  
  
#### \*\*2.2 Hardware Interfaces\*\*  
  
\*\*2.2.1 Database Interaction\*\*  
- \*\*Description:\*\* Manages storage of customer, product, order, and audit records.  
- \*\*Inputs:\*\* Data to be stored.  
- \*\*Outputs:\*\* Stored records or query results.  
  
---  
  
#### \*\*2.3 Software Interfaces\*\*  
  
\*\*2.3.1 Payment API\*\*  
- \*\*Description:\*\* Processes transactions.  
- \*\*Inputs:\*\* Payment details, OrderID.  
- \*\*Outputs:\*\* Transaction ID, payment status.  
  
\*\*2.3.2 Email Notification System\*\*  
- \*\*Description:\*\* Sends transactional emails.  
- \*\*Inputs:\*\* Recipient email, template, content.  
- \*\*Outputs:\*\* Delivered emails, delivery logs.  
  
\*\*2.3.3 Carrier Tracking API\*\*  
- \*\*Description:\*\* Retrieves shipment status.  
- \*\*Inputs:\*\* OrderID.  
- \*\*Outputs:\*\* Tracking data, cached status.  
  
\*\*2.3.4 Plugin API\*\*  
- \*\*Description:\*\* Integrates third-party functionality.  
- \*\*Inputs:\*\* API requests.  
- \*\*Outputs:\*\* Plugin operations, system updates.  
  
---  
  
#### \*\*2.4 Communication Interfaces\*\*  
  
\*\*2.4.1 Email Notifications\*\*  
- \*\*Description:\*\* Communication via email for order confirmations and password resets.  
- \*\*Inputs:\*\* Recipient, template, content.  
- \*\*Outputs:\*\* Sent emails, delivery status logs.  
  
\*\*2.4.2 API Communication\*\*  
- \*\*Description:\*\* Interacts with external APIs for payment, tracking, and plugins.  
- \*\*Inputs:\*\* API requests.  
- \*\*Outputs:\*\* API responses, system modifications.  
  
---  
  
This structured approach ensures all external interfaces are clearly defined, aiding developers in understanding and implementing the system effectively. Each interface is categorized, described, and linked to its role in the system's functionality.

# Use Case

### Use Case Description   
\*\*Use Case Name:\*\* Manage Customer Account   
\*\*Use Case ID:\*\* UC-01   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has sufficient privileges to manage customer data.   
3. Customer data repository is accessible.   
  
\*\*Postconditions:\*\*   
1. Customer records are created, updated, or deactivated as requested.   
2. Database reflects the latest changes to customer information.   
3. Audit log records the administrator's action.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Customer Management" from the dashboard.   
2. System displays a list of existing customers (name, ID, status).   
3. Administrator chooses a specific customer record to edit.   
4. System loads the customer’s details (e.g., contact, address, preferences).   
5. Administrator modifies required fields and confirms submission.   
6. System validates and saves changes, then notifies administrator of success.   
  
\*\*Alternative Flow:\*\*   
\*Alternative Flow A: Create New Customer\*   
1. At step 2, administrator clicks "Add New Customer."   
2. System provides an empty customer details form.   
3. Administrator enters mandatory data (name, email) and submits.   
4. System generates a unique customer ID and stores the record.   
  
\*Alternative Flow B: Deactivate Customer\*   
1. At step 3, administrator selects "Deactivate Account."   
2. System prompts for confirmation with reason (e.g., "Inactive").   
3. Administrator confirms; system flags the account as inactive and revokes access.   
  
---   
\*Note: Flows focus on core actions; validation errors or permission issues trigger system notifications without separate branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Register Customer   
\*\*Use Case ID:\*\* UC-02   
\*\*Actors:\*\* Customer (Unregistered User)   
\*\*Preconditions:\*\*   
1. Customer accesses the system’s public interface (e.g., website/mobile app).   
2. Registration functionality is available and operational.   
  
\*\*Postconditions:\*\*   
1. New customer account is created with unique ID.   
2. Customer credentials are securely stored in the database.   
3. Customer gains authenticated access to the system.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Register" option on the login screen.   
2. System displays a registration form requiring name, email, password, and contact details.   
3. Customer completes all mandatory fields and submits the form.   
4. System validates data format (e.g., email syntax, password strength).   
5. System verifies email uniqueness against existing records.   
6. System creates a new customer record, generates a unique ID, and stores encrypted credentials.   
7. System confirms successful registration and redirects to login page.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Input Format\*   
1. At step 4, if validation fails (e.g., malformed email):   
2. System flags erroneous fields with specific error messages.   
3. Form retains entered data; customer corrects errors and resubmits.   
  
\*B. Duplicate Email\*   
1. At step 5, if email exists in the database:   
2. System displays "Email already registered" alert.   
3. Customer modifies email or recovers existing account via "Forgot Password."   
  
---   
\*Note: Additional alternatives (e.g., registration timeout) trigger generic error prompts without dedicated flows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Login Customer   
\*\*Use Case ID:\*\* UC-03   
\*\*Actors:\*\* Customer (Registered User)   
\*\*Preconditions:\*\*   
1. Customer has a registered account in the system.   
2. Login functionality is operational and accessible.   
  
\*\*Postconditions:\*\*   
1. Customer gains authenticated access to the system.   
2. Session is established with appropriate permissions.   
3. Login timestamp is recorded in the audit log.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to the login page.   
2. System displays login form with email and password fields.   
3. Customer enters registered email and password, then submits.   
4. System validates credential format and verifies against stored records.   
5. System authenticates customer and redirects to customer dashboard.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Credentials\*   
1. At step 4, if credentials are incorrect or unmatched:   
2. System displays "Invalid email or password" error.   
3. Form retains email input; password field is cleared for retry.   
  
\*B. Inactive Account\*   
1. At step 4, if account is flagged as inactive:   
2. System blocks login and displays "Account deactivated" alert.   
3. Provides administrator contact information for resolution.   
  
---   
\*Note: Additional scenarios (e.g., expired sessions) trigger system-guided recovery flows without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Logout Customer   
\*\*Use Case ID:\*\* UC-04   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is currently authenticated and has an active session.   
2. Logout functionality is accessible in the navigation interface.   
  
\*\*Postconditions:\*\*   
1. Customer's session is terminated and authentication privileges revoked.   
2. Session end timestamp is recorded in the audit log.   
3. Customer is redirected to the public interface.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Logout" from the account menu.   
2. System displays a confirmation dialog for logout action.   
3. Customer confirms logout request.   
4. System invalidates session token and clears client-side session data.   
5. System records logout event (customer ID, timestamp) in audit log.   
6. System redirects customer to the login page.   
  
\*\*Alternative Flow:\*\*   
\*A. Session Timeout During Logout\*   
1. At step 4, if session expires before completion:   
2. System automatically terminates session and redirects to login page.   
3. Audit log records timeout-triggered logout with expiration timestamp.   
  
\*B. Cancel Logout Request\*   
1. At step 3, if customer selects "Cancel":   
2. System closes confirmation dialog and maintains active session.   
3. Customer remains on current interface without state changes.   
  
---   
\*Note: Network interruptions during logout trigger automatic session cleanup via scheduled system jobs.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Customer Information   
\*\*Use Case ID:\*\* UC-05   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Profile management functionality is accessible.   
3. Customer data repository is available for read/write operations.   
  
\*\*Postconditions:\*\*   
1. Customer's modified information is saved in the database.   
2. System reflects updated information across all customer-facing interfaces.   
3. Audit log records the modification event with timestamp.   
  
\*\*Main Flow:\*\*   
1. Customer selects "My Profile" from the account dashboard.   
2. System displays current customer details (name, contact, address, preferences).   
3. Customer edits desired fields and confirms submission.   
4. System validates input formats (e.g., email syntax, phone number structure).   
5. System updates customer record and persists changes.   
6. System displays "Profile updated successfully" notification.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Data Format\*   
1. At step 4, if validation fails (e.g., invalid postal code):   
2. System flags erroneous fields with specific error messages.   
3. Form retains valid entries; customer corrects errors and resubmits.   
  
\*B. Unauthorized Field Modification\*   
1. At step 3, if customer attempts to edit restricted fields (e.g., account status):   
2. System rejects changes and displays "Field not editable" alert.   
3. Customer continues editing with permitted fields only.   
  
---   
\*Note: System errors or concurrent updates trigger data conflict resolution workflows without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Purchase History   
\*\*Use Case ID:\*\* UC-06   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Order history data repository is accessible.   
  
\*\*Postconditions:\*\*   
1. Customer's purchase history is displayed.   
2. Access event is recorded in the audit log.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Purchase History" from the account dashboard.   
2. System retrieves the customer’s past orders from the database.   
3. System displays a summary list of orders (order ID, date, total amount, status).   
4. Customer browses the order history list.   
  
\*\*Alternative Flow:\*\*   
\*A. View Order Details\*   
1. At step 4, customer selects a specific order.   
2. System retrieves detailed order information (products, quantities, prices, shipping address).   
3. System displays the full order details.   
  
\*B. No Purchase History\*   
1. At step 2, if no orders exist for the customer:   
2. System displays "No orders found" message.   
3. Customer returns to the dashboard.   
  
---   
\*Note: Data retrieval errors trigger system notifications without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Payment Information   
\*\*Use Case ID:\*\* UC-07   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Payment management functionality is accessible.   
3. Payment information repository is available for read/write operations.   
  
\*\*Postconditions:\*\*   
1. Customer's payment methods are added, updated, or removed as requested.   
2. Database reflects the latest payment information changes.   
3. Audit log records modification events with timestamps.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Payment Methods" from the account dashboard.   
2. System displays saved payment options (e.g., card type, last four digits, expiration).   
3. Customer chooses an existing payment method to edit.   
4. System loads editable details (billing address, card expiry).   
5. Customer modifies fields and confirms submission.   
6. System validates input formats (e.g., expiry date, CVV).   
7. System updates payment record and confirms success.   
  
\*\*Alternative Flow:\*\*   
\*A. Add New Payment Method\*   
1. At step 2, customer selects "Add New Card."   
2. System displays blank payment form with mandatory fields.   
3. Customer enters card details and billing information.   
4. System validates and stores new payment method.   
  
\*B. Remove Payment Method\*   
1. At step 3, customer selects "Delete."   
2. System prompts confirmation with security check (e.g., password re-entry).   
3. Customer confirms; system removes payment record and updates list.   
  
---   
\*Note: Validation failures trigger field-specific error messages; system errors show generic alerts without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add Product   
\*\*Use Case ID:\*\* UC-08   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Product Management" privileges.   
3. Product repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. New product record is created with unique ID in the database.   
2. Product becomes immediately available for customer orders.   
3. Audit log records the addition event with timestamp.   
  
\*\*Main Flow:\*\*   
1. Administrator navigates to \*\*Product Management\*\* from the dashboard.   
2. System displays existing product catalog with filtering options.   
3. Administrator selects \*\*Add New Product\*\*.   
4. System loads a blank product form (name, description, price, category, stock quantity).   
5. Administrator fills mandatory fields and submits.   
6. System validates data formats (e.g., price ≥0, stock as integer).   
7. System generates unique product ID, stores record, and confirms success.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Data Input\*   
1. At step 6, if validation fails (e.g., negative price):   
2. System highlights invalid fields with specific error messages.   
3. Form retains valid entries; administrator corrects errors and resubmits.   
  
\*B. Duplicate Product Entry\*   
1. At step 6, if identical product name/SKU exists:   
2. System displays "Product already exists" alert with conflict details.   
3. Administrator modifies product attributes (e.g., SKU or name) and resubmits.   
  
---   
\*Note: Permission errors or repository unavailability trigger system-wide notifications without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Product   
\*\*Use Case ID:\*\* UC-09   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Product Management" privileges.   
3. Product repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Modified product details are saved in the database.   
2. Updated product information is reflected across all system interfaces.   
3. Audit log records the update event with timestamp.   
  
\*\*Main Flow:\*\*   
1. Administrator navigates to \*\*Product Management\*\* from the dashboard.   
2. System displays the product catalog with search/filter options.   
3. Administrator selects a specific product to update.   
4. System loads current product details (name, description, price, stock, category).   
5. Administrator modifies editable fields and confirms submission.   
6. System validates input formats (e.g., price ≥0, stock integer ≥0).   
7. System updates product record and displays success confirmation.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Data Input\*   
1. At step 6, if validation fails (e.g., negative price):   
2. System flags erroneous fields with specific error messages.   
3. Form retains valid entries; administrator corrects errors and resubmits.   
  
\*B. Concurrent Update Conflict\*   
1. At step 7, if product was modified by another user during editing:   
2. System detects version mismatch, displays conflict warning with latest data.   
3. Administrator reviews changes, reconciles edits, and resubmits.   
  
---   
\*Note: Permission errors or repository failures trigger system notifications without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete Product   
\*\*Use Case ID:\*\* UC-10   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Product Management" privileges.   
3. Product repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Product is permanently removed from the database.   
2. Product is no longer available for orders or shopping carts.   
3. Audit log records deletion event with timestamp and administrator ID.   
  
\*\*Main Flow:\*\*   
1. Administrator navigates to \*\*Product Management\*\* from the dashboard.   
2. System displays product catalog with search/filter options.   
3. Administrator selects target product and chooses \*\*Delete Product\*\* action.   
4. System prompts confirmation with product details and warning message.   
5. Administrator confirms deletion request.   
6. System verifies no active dependencies (e.g., pending orders/shopping carts).   
7. System permanently removes product record from database.   
8. System displays "Product deleted successfully" notification.   
  
\*\*Alternative Flow:\*\*   
\*A. Active Dependencies Detected\*   
1. At step 6, if product exists in pending orders or active shopping carts:   
2. System blocks deletion and displays "Cannot delete: Product in use" alert.   
3. Administrator must resolve dependencies before retrying.   
  
\*B. Cancel Deletion Request\*   
1. At step 5, if administrator selects "Cancel":   
2. System aborts process and returns to product catalog view.   
3. Product record remains unchanged.   
  
---   
\*Note: Database errors trigger system-wide notifications; permission issues redirect to privilege verification.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Inventory   
\*\*Use Case ID:\*\* UC-11   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Inventory Management" privileges.   
3. Inventory data repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Inventory levels are updated in the database.   
2. Low-stock alerts are triggered if applicable.   
3. Audit log records all inventory adjustments.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*Inventory Management\*\* from the dashboard.   
2. System displays current inventory levels (product, stock quantity, reorder status).   
3. Administrator chooses a product to adjust stock.   
4. System loads product stock details and provides adjustment interface.   
5. Administrator enters new stock quantity and confirms.   
6. System validates input (non-negative integer, within operational limits).   
7. System updates inventory record and triggers low-stock alerts if needed.   
8. System confirms successful update and refreshes inventory view.   
  
\*\*Alternative Flow:\*\*   
\*A. Low-Stock Replenishment\*   
1. At step 2, system highlights products below minimum stock threshold.   
2. Administrator selects low-stock product and enters replenishment quantity.   
3. System auto-calculates required shipment and updates inventory.   
  
\*B. Stock Correction (Discrepancy Resolution)\*   
1. At step 5, administrator selects "Adjustment Reason" from predefined options (e.g., damage, loss).   
2. System requires explanatory notes for audit trail.   
3. Special validation applies (e.g., maximum loss percentage).   
  
---   
\*Note: Concurrent inventory updates trigger data conflict resolution; system errors show generic alerts without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Place Order   
\*\*Use Case ID:\*\* UC-12   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Shopping cart contains at least one product.   
3. Customer's default shipping address and payment method are set or will be provided during checkout.   
  
\*\*Postconditions:\*\*   
1. New order is created with unique order ID and status "Pending".   
2. Shopping cart is cleared.   
3. Inventory stock levels are reduced for ordered items.   
4. Order details are stored in the database and audit log.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Checkout" from the shopping cart view.   
2. System displays order summary (products, quantities, prices, total).   
3. Customer confirms shipping address or enters a new valid address.   
4. System calculates shipping costs and updates order total.   
5. Customer selects payment method and provides required details.   
6. System validates payment information and authorizes transaction.   
7. Customer confirms purchase.   
8. System creates order record, clears shopping cart, and reduces inventory stock.   
9. System sends order confirmation to customer via email.   
  
\*\*Alternative Flow:\*\*   
\*A. Modify Order Before Confirmation\*   
1. At step 2, customer selects "Edit Cart".   
2. System returns to shopping cart for modifications.   
3. Customer updates items/quantities and resumes checkout.   
  
\*B. Payment Authorization Failure\*   
1. At step 6, if payment authorization fails:   
2. System displays "Payment declined" error with reason.   
3. Customer updates payment details or selects alternative method.   
  
\*C. Address Validation Error\*   
1. At step 3, if address is invalid/incomplete:   
2. System flags errors and requires correction.   
3. Customer provides valid address before proceeding.   
  
\*D. Insufficient Inventory\*   
1. At step 8, if any product stock is insufficient:   
2. System removes unavailable items and recalculates total.   
3. Customer confirms revised order or cancels checkout.   
  
---   
\*Note: System errors during order processing trigger transaction rollback and failure notifications without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Order Details   
\*\*Use Case ID:\*\* UC-13   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Target order exists in the database and belongs to the customer.   
3. Order data repository is accessible.   
  
\*\*Postconditions:\*\*   
1. Detailed order information is displayed to the customer.   
2. Access event is recorded in the audit log (customer ID, timestamp).   
  
\*\*Main Flow:\*\*   
1. Customer selects an order from their purchase history list.   
2. System retrieves full order details (products, quantities, prices, shipping address, payment method, status).   
3. System displays structured order summary with itemized breakdown.   
  
\*\*Alternative Flow:\*\*   
\*A. Order Retrieval Failure\*   
1. At step 2, if system cannot fetch order data (e.g., database error):   
2. System displays "Unable to load order details" notification.   
3. Customer is returned to purchase history screen.   
  
---   
\*Note: Permission validation occurs implicitly; access attempts to non-owned orders trigger automatic rejection without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Order   
\*\*Use Case ID:\*\* UC-14   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Order Management" privileges.   
3. Order data repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Order status or details are updated in the database.   
2. Dependent entities (e.g., inventory) are synchronized if modified.   
3. Audit log records all order management actions.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*Order Management\*\* from the dashboard.   
2. System displays active orders (ID, customer, total, status) with search/filter options.   
3. Administrator selects a target order.   
4. System loads full order details (products, quantities, shipping address, payment status).   
5. Administrator modifies fields (e.g., order status, tracking number) and confirms.   
6. System validates changes (e.g., valid status transitions).   
7. System updates order record and synchronizes dependent data (e.g., inventory restock on cancellation).   
8. System sends status notification to customer automatically.   
  
\*\*Alternative Flow:\*\*   
\*A. Cancel Order\*   
1. At step 5, administrator selects "Cancel Order".   
2. System verifies cancellation eligibility (e.g., pre-shipment).   
3. Administrator provides reason; system reverses payment and restocks inventory.   
4. Order status updates to "Cancelled" with audit trail.   
  
\*B. Partial Refund\*   
1. At step 5, administrator initiates "Partial Refund".   
2. System displays refundable items with max refund amounts.   
3. Administrator selects items/quantities and confirms refund value.   
4. System processes partial payment reversal and updates order notes.   
  
\*C. Fulfillment Exception\*   
1. At step 6, if status change violates rules (e.g., "Shipped" without tracking):   
2. System blocks update and specifies missing requirements.   
3. Administrator corrects omissions and resubmits.   
  
---   
\*Note: Database errors trigger rollback; permission issues show standardized alerts without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Track Order   
\*\*Use Case ID:\*\* UC-15   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged into the system with an active session.   
2. Target order exists in the database and belongs to the customer.   
3. Order has progressed beyond "Pending" status (e.g., shipped or in transit).   
  
\*\*Postconditions:\*\*   
1. Real-time order tracking information is displayed to the customer.   
2. Access event is recorded in the audit log (customer ID, timestamp).   
  
\*\*Main Flow:\*\*   
1. Customer selects "Track Order" from the purchase history or dashboard.   
2. System displays a list of eligible orders (ID, date, last status).   
3. Customer selects target order.   
4. System retrieves latest tracking data (carrier, tracking number, location, status).   
5. System displays dynamic tracking view with status timeline and estimated delivery.   
  
\*\*Alternative Flow:\*\*   
\*A. Order Not Shipped\*   
1. At step 4, if order status is "Processing" or "Pending":   
2. System displays "Preparing for shipment" status with expected ship date.   
3. Provides option to contact support for expedited requests.   
  
\*B. Refreshed Tracking Data\*   
1. At step 5, customer selects "Refresh Tracking":   
2. System polls carrier API for real-time updates.   
3. Overlays new status markers on tracking timeline if changes exist.   
  
---   
\*Note: Carrier API failures display cached data with timestamp; stale data warnings appear after 24-hour staleness threshold.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Login Administrator   
\*\*Use Case ID:\*\* UC-16   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Admin login functionality is operational.   
2. Administrator credentials exist in the system database.   
  
\*\*Postconditions:\*\*   
1. Administrator gains authenticated access to the admin dashboard.   
2. Session is established with administrative privileges.   
3. Login event (admin ID, timestamp) is recorded in audit log.   
  
\*\*Main Flow:\*\*   
1. Administrator accesses the admin login portal.   
2. System displays login form with username and password fields.   
3. Administrator enters credentials and submits.   
4. System validates credential format and verifies against stored records.   
5. System authenticates administrator and redirects to admin dashboard.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Credentials\*   
1. At step 4, if credentials are incorrect:   
2. System displays "Invalid credentials" error.   
3. Form retains username; password field clears for retry.   
  
\*B. Account Lockout\*   
1. At step 4, after 3 consecutive failures:   
2. System locks account temporarily and displays "Account locked" alert.   
3. Requires password reset or administrator intervention.   
  
---   
\*Note: System errors during authentication trigger generic failure messages without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Logout Administrator   
\*\*Use Case ID:\*\* UC-17   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is currently authenticated with an active session.   
2. Logout functionality is accessible in the admin interface.   
  
\*\*Postconditions:\*\*   
1. Administrator's session is terminated and authentication privileges revoked.   
2. Session end timestamp is recorded in the audit log.   
3. Administrator is redirected to the admin login portal.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Logout" from the admin dashboard menu.   
2. System displays a confirmation dialog for logout action.   
3. Administrator confirms logout request.   
4. System invalidates session token and clears client-side credentials.   
5. System records logout event (admin ID, timestamp) in the audit log.   
6. System redirects administrator to the admin login page.   
  
\*\*Alternative Flow:\*\*   
\*A. Session Expiration During Logout\*   
1. At step 4, if session expires before token invalidation:   
2. System automatically terminates session and redirects to login portal.   
3. Audit log records automatic logout due to timeout.   
  
\*B. Cancel Logout Request\*   
1. At step 3, if administrator selects "Cancel":   
2. System closes confirmation dialog without session termination.   
3. Administrator remains on the current dashboard interface.   
  
---   
\*Note: Network failures during logout trigger automatic session cleanup via backend cron jobs.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage System Settings   
\*\*Use Case ID:\*\* UC-18   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "System Configuration" privileges.   
3. System settings repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Configuration parameters are updated per administrator input.   
2. Changes are persisted across all system components.   
3. Modification events are recorded in the audit log.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*System Settings\*\* from the admin dashboard.   
2. System displays configuration categories (e.g., Security, Notifications, UI Preferences).   
3. Administrator chooses a category and modifies parameters (e.g., session timeout, email templates).   
4. System validates input formats and value ranges.   
5. Administrator confirms changes.   
6. System propagates updates and displays success confirmation.   
  
\*\*Alternative Flow:\*\*   
\*A. Invalid Configuration Value\*   
1. At step 4, if validation fails (e.g., timeout <5 minutes):   
2. System highlights invalid fields with constraint explanations.   
3. Administrator corrects values and resubmits.   
  
\*B. Rollback to Default Settings\*   
1. At step 3, administrator selects "Restore Defaults".   
2. System prompts confirmation with impacted components list.   
3. Administrator confirms; system reverts settings and logs full reset.   
  
---   
\*Note: Interruption during propagation triggers auto-rollback; permission errors show standardized alerts without branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add to Cart   
\*\*Use Case ID:\*\* UC-19   
\*\*Actors:\*\* Customer (Authenticated or Unauthenticated User)   
\*\*Preconditions:\*\*   
1. Product catalog is accessible and operational.   
2. Target product exists in the system and is available for purchase.   
3. Customer session is active (if unauthenticated, session-based cart exists).   
  
\*\*Postconditions:\*\*   
1. Specified product and quantity are added to the customer's shopping cart.   
2. Cart total and item count are updated across all interfaces.   
3. Cart state is persisted (session storage or account-linked database).   
  
\*\*Main Flow:\*\*   
1. Customer selects target product from catalog or search results.   
2. System displays product details page with quantity selector.   
3. Customer sets desired quantity and clicks "Add to Cart".   
4. System validates product availability and quantity format (positive integer).   
5. System adds product/quantity to cart and updates cart metadata (subtotal, item count).   
6. System displays cart confirmation toast and updates cart icon.   
  
\*\*Alternative Flow:\*\*   
\*A. Quantity Exceeds Stock\*   
1. At step 4, if requested quantity > available stock:   
2. System auto-adjusts to max available quantity with warning tooltip.   
3. Cart updates reflect adjusted quantity; customer may modify via cart view.   
  
\*B. Existing Cart Item Modification\*   
1. At step 4, if product already exists in cart:   
2. System aggregates quantities (new + existing) if below stock limit.   
3. Cart displays merged item line with updated quantity/total.   
  
---   
\*Note: Session expiration triggers cart preservation via browser storage; permission issues occur only for restricted products.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Cart   
\*\*Use Case ID:\*\* UC-20   
\*\*Actors:\*\* Customer (Authenticated or Unauthenticated User)   
\*\*Preconditions:\*\*   
1. Shopping cart functionality is operational and accessible.   
2. Customer session is active (session-based cart exists for unauthenticated users).   
3. Product catalog data repository is accessible.   
  
\*\*Postconditions:\*\*   
1. Current cart contents are displayed with real-time pricing and availability.   
2. Cart modification options become accessible (e.g., update quantity, remove item).   
3. Access event is recorded in session log.   
  
\*\*Main Flow:\*\*   
1. Customer selects "View Cart" from navigation menu or cart icon.   
2. System retrieves cart items with current product details (name, price, images).   
3. System checks real-time inventory status for each cart item.   
4. System calculates and displays cart summary (subtotal, taxes, estimated total).   
5. Customer reviews cart contents and available actions.   
  
\*\*Alternative Flow:\*\*   
\*A. Empty Cart\*   
1. At step 2, if cart contains no items:   
2. System displays "Your cart is empty" prompt.   
3. Provides direct link to product catalog.   
  
\*B. Product Unavailability\*   
1. At step 3, if any product becomes unavailable:   
2. System flags affected items with "Out of Stock" status.   
3. Removes auto-selection for checkout but retains in cart for monitoring.   
  
\*C. Price Update During View\*   
1. At step 3, if product price changed since cart addition:   
2. System displays original/new prices side-by-side.   
3. Requires customer confirmation before allowing checkout.   
  
---   
\*Note: Session expiration preserves cart via browser storage; permission issues only occur for restricted products.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Modify Cart   
\*\*Use Case ID:\*\* UC-21   
\*\*Actors:\*\* Customer (Authenticated or Unauthenticated User)   
\*\*Preconditions:\*\*   
1. Shopping cart functionality is operational.   
2. Customer session is active (session-based cart exists for unauthenticated users).   
3. Cart contains at least one modifiable item.   
4. Product catalog and inventory data repositories are accessible.   
  
\*\*Postconditions:\*\*   
1. Cart contents are updated per modification request.   
2. Cart metadata (subtotal, item count) is recalculated and persisted.   
3. Real-time inventory checks are reflected in modification constraints.   
  
\*\*Main Flow:\*\*   
1. Customer views cart contents via "Shopping Cart" interface.   
2. Customer selects an item and chooses modification action (update quantity/remove).   
3. System retrieves current product availability and pricing data.   
4. Customer enters new quantity or confirms removal.   
5. System validates input (positive integer ≤ available stock).   
6. System updates cart records and recalculates totals.   
7. Customer receives visual confirmation of successful modification.   
  
\*\*Alternative Flow:\*\*   
\*A. Quantity Adjustment Failure\*   
1. At step 5, if new quantity exceeds stock or violates format:   
2. System auto-reverts to last valid quantity with error justification.   
3. Highlights affected item for manual correction.   
  
\*B. Batch Item Removal\*   
1. At step 2, customer selects "Remove All Items".   
2. System prompts confirmation with cart summary.   
3. Upon confirmation, clears entire cart and resets metadata.   
  
\*C. Concurrent Inventory Change\*   
1. At step 3, if stock decreases during modification:   
2. System enforces current stock ceiling and displays inventory alert.   
3. Preserves customer's attempted quantity as placeholder.   
  
---   
\*Note: Session expiration preserves pending modifications; permission issues only occur for restricted products.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Checkout   
\*\*Use Case ID:\*\* UC-22   
\*\*Actors:\*\* Customer (Authenticated User)   
\*\*Preconditions:\*\*   
1. Customer is logged in with an active session.   
2. Shopping cart contains at least one product.   
3. System’s payment gateway and inventory services are operational.   
  
\*\*Postconditions:\*\*   
1. New order created with unique ID and status "Pending".   
2. Shopping cart is cleared.   
3. Inventory levels reduced for purchased items.   
4. Order details stored in database and audit log.   
  
\*\*Main Flow:\*\*   
1. Customer selects \*\*Checkout\*\* from shopping cart view.   
2. System displays order summary (products, quantities, prices subtotal).   
3. Customer confirms shipping address or enters new valid address.   
4. System calculates shipping costs/taxes and updates order total.   
5. Customer selects saved payment method or enters new card details.   
6. System validates payment credentials and authorizes transaction.   
7. Customer confirms final purchase.   
8. System creates order record, clears cart, adjusts inventory, and sends confirmation email.   
  
\*\*Alternative Flow:\*\*   
\*A. Address Validation Failure\*   
1. At step 3, if address format invalid/unserviceable:   
2. System highlights errors with correction guidelines.   
3. Customer updates address before proceeding.   
  
\*B. Payment Authorization Decline\*   
1. At step 6, if payment gateway declines transaction:   
2. System displays decline reason (e.g., "Insufficient funds").   
3. Customer updates payment details or selects alternative method.   
  
\*C. Real-time Inventory Shortage\*   
1. At step 8, if any product stock falls below ordered quantity:   
2. System removes unavailable items and recalculates total.   
3. Customer confirms revised order or cancels checkout.   
  
\*D. Session Timeout During Checkout\*   
1. At any step, if session expires:   
2. System preserves cart state via browser storage.   
3. Upon relogin, redirects customer to last active checkout step.   
  
---   
\*Note: Payment gateway failures trigger automatic retries; critical system errors abort checkout with recovery instructions.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Category   
\*\*Use Case ID:\*\* UC-23   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Category Management" privileges.   
3. Category repository is accessible and operational.   
  
\*\*Postconditions:\*\*   
1. Category records are created, updated, or deleted as requested.   
2. Product catalog reflects updated category associations.   
3. Audit log records category modifications with timestamp and administrator ID.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*Category Management\*\* from the dashboard.   
2. System displays hierarchical category tree with search/filter options.   
3. Administrator selects a target category to edit.   
4. System loads category details (name, description, parent category, associated products).   
5. Administrator modifies fields and confirms submission.   
6. System validates inputs (e.g., non-empty name, valid parent hierarchy).   
7. System updates category record and synchronizes product associations.   
8. System displays "Category updated successfully" notification.   
  
\*\*Alternative Flow:\*\*   
\*A. Add New Category\*   
1. At step 2, administrator selects "Create New Category".   
2. System provides blank form with mandatory fields (name, hierarchy level).   
3. Administrator enters details and assigns parent category.   
4. System validates uniqueness and hierarchy rules before persisting.   
  
\*B. Delete Category with Dependencies\*   
1. At step 3, administrator selects "Delete Category".   
2. System detects products/child-categories assigned to target category.   
3. Administrator reassigns dependencies to alternate categories.   
4. System verifies no orphaned records before deletion.   
  
\*C. Circular Hierarchy Prevention\*   
1. At step 6, if parent assignment creates circular dependency:   
2. System blocks update and displays "Invalid hierarchy: circular reference detected".   
3. Administrator reselects valid parent category.   
  
---   
\*Note: Concurrent modifications trigger data version checks; repository errors show system alerts without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Payment   
\*\*Use Case ID:\*\* UC-24   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Payment Management" privileges.   
3. Payment repository and financial gateway services are operational.   
  
\*\*Postconditions:\*\*   
1. Payment records are updated to reflect new status (e.g., refunded, disputed).   
2. Financial transactions are synchronized with external payment gateways.   
3. Audit log records all payment modifications with timestamp and administrator ID.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*Payment Management\*\* from the dashboard.   
2. System displays recent payments (transaction ID, amount, status, customer) with search filters.   
3. Administrator selects a target payment transaction.   
4. System loads full payment details (order reference, method, authorization code, timestamp).   
5. Administrator performs action (e.g., initiate refund, flag dispute) and provides rationale.   
6. System verifies action validity (e.g., refund amount ≤ original, dispute window open).   
7. System executes transaction via payment gateway API and updates payment status.   
8. System syncs changes with related order records and notifies customer via email.   
  
\*\*Alternative Flow:\*\*   
\*A. Full Refund Processing\*   
1. At step 5, administrator selects "Refund Full Amount".   
2. System auto-populates refund value and verifies no prior refunds exist.   
3. On confirmation, reverses transaction via gateway and sets status to "Refunded".   
  
\*B. Dispute Resolution\*   
1. At step 5, administrator selects "Resolve Dispute".   
2. System provides evidence upload interface and outcome options (e.g., "Merchant Wins", "Chargeback").   
3. Upon submission, updates payment status and triggers financial adjustments.   
  
\*C. Partial Capture Adjustment\*   
1. At step 5, for pre-authorized payments:   
2. Administrator modifies captured amount (≤ authorized limit).   
3. System submits partial capture to gateway and recalculates order balance.   
  
---   
\*Note: Gateway API failures trigger automated retries; invalid actions display error constraints without dedicated branching.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage OrderItem   
\*\*Use Case ID:\*\* UC-25   
\*\*Actors:\*\* Administrator   
\*\*Preconditions:\*\*   
1. Administrator is logged into the system.   
2. Administrator has "Order Management" privileges.   
3. Target order exists and is in modifiable state (e.g., "Pending" or "Processing").   
4. OrderItem and inventory repositories are accessible.   
  
\*\*Postconditions:\*\*   
1. Order items are added, modified, or removed as requested.   
2. Order total and inventory stock are recalculated and synchronized.   
3. Order modification event is recorded in audit log with item-level details.   
  
\*\*Main Flow:\*\*   
1. Administrator selects \*\*Order Management\*\* from the dashboard.   
2. System displays active orders; administrator selects target order.   
3. System loads order details including item list (product, quantity, price).   
4. Administrator chooses order item action: \*\*Add Item\*\*, \*\*Edit Item\*\*, or \*\*Remove Item\*\*.   
5. System performs requested operation (see alternative flows).   
6. System recalculates order total and adjusts inventory stock.   
7. System updates order record and displays "Order items updated successfully".   
  
\*\*Alternative Flow:\*\*   
\*A. Add New Order Item\*   
1. At step 4, administrator selects "Add Item".   
2. System provides product search interface with stock availability indicators.   
3. Administrator selects product and enters valid quantity (≤ available stock).   
4. System adds item to order with current price and inventory lock.   
  
\*B. Modify Existing Order Item\*   
1. At step 4, administrator selects "Edit" on existing item.   
2. System displays editable fields (quantity, special instructions).   
3. Administrator updates quantity within stock constraints.   
4. System calculates stock delta and adjusts inventory accordingly.   
  
\*C. Remove Order Item\*   
1. At step 4, administrator selects "Remove" on item.   
2. System prompts confirmation with impact summary.   
3. Administrator confirms; system restocks item quantity and removes from order.   
  
\*D. Unmodifiable Order State\*   
1. At step 3, if order status prohibits changes (e.g., "Shipped"):   
2. System blocks actions and displays "Order cannot be modified" alert.   
3. Administrator may escalate via exception management workflows.   
  
---   
\*Note: Concurrent inventory changes trigger real-time availability rechecks; price updates apply current catalog values to new items.\*