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 email verification. Initializes a default shopping cart upon successful registration.   
Input: Customer name, email, password, contact details via registration form.   
Output: New customer record, initialized shopping cart, account confirmation email.   
  
1.2 Customer Login Function   
Function ID: FR-02   
Description: Authenticates customer credentials and establishes secure sessions. Enforces account status checks and login attempt limits.   
Input: Email/username and password from login form.   
Output: Authenticated session, access to customer dashboard, login audit log entry.   
  
1.3 Customer Logout Function   
Function ID: FR-03   
Description: Terminates authenticated sessions securely and revokes system access. Handles multi-device logout requests.   
Input: Logout request from authenticated customer session.   
Output: Session invalidation, audit log entry, redirection to public landing page.   
  
1.4 Customer Profile View Function   
Function ID: FR-04   
Description: Displays customer personal information in read-only format with privacy masking for sensitive fields.   
Input: Authenticated customer session or administrator search query.   
Output: Customer profile details (name, contact, address) with masked sensitive data.   
  
1.5 Customer Profile Update Function   
Function ID: FR-05   
Description: Modifies customer personal details with validation for format and uniqueness. Synchronizes changes across dependent entities.   
Input: Updated profile fields (email, phone, address) from edit form.   
Output: Updated customer record, dependent entity synchronization, audit log.   
  
1.6 Customer Account Deactivation Function   
Function ID: FR-06   
Description: Archives customer accounts and revokes system access. Verifies no pending transactions before deactivation.   
Input: Account deactivation request from administrator.   
Output: Archived customer record, access revocation, audit log entry.   
  
1.7 Customer Login Log View Function   
Function ID: FR-07   
Description: Displays historical login records with privacy controls. Supports date-based filtering and export.   
Input: Customer ID and date range parameters (for administrators).   
Output: Login history records with masked sensitive data.   
  
1.8 Administrator Registration Function   
Function ID: FR-08   
Description: Creates new administrator accounts with privilege-level validation and activation workflows.   
Input: Administrator name, email, role, privilege level from admin console.   
Output: New administrator record, activation email, privilege audit log.   
  
1.9 Administrator Login Function   
Function ID: FR-09   
Description: Authenticates administrator credentials with elevated security measures including 2FA for high-privilege accounts.   
Input: Email and password from admin login interface.   
Output: Privileged admin session, administrative dashboard access, audit log.   
  
1.10 Administrator Logout Function   
Function ID: FR-10   
Description: Terminates admin sessions with enhanced security protocols including multi-session revocation.   
Input: Logout request from active admin session.   
Output: Session termination, privilege revocation, admin audit log entry.   
  
1.11 Product Creation Function   
Function ID: FR-11   
Description: Adds new products to the catalog with validation for SKU uniqueness and category assignments.   
Input: Product details (name, description, price, SKU) and category selection.   
Output: New product record, inventory reservation, audit log entry.   
  
1.12 Product View Function   
Function ID: FR-12   
Description: Displays product details with context-appropriate data masking. Shows real-time stock status.   
Input: Product ID from catalog selection.   
Output: Product details (name, price, description) with admin/customer view variations.   
  
1.13 Product Update Function   
Function ID: FR-13   
Description: Modifies product information and propagates changes to dependent entities. Validates price and inventory impacts.   
Input: Modified product fields from edit interface.   
Output: Updated product record, catalog synchronization, impact notifications.   
  
1.14 Product Deletion Function   
Function ID: FR-14   
Description: Archives products after verifying no pending orders. Releases inventory reservations.   
Input: Product deletion request from administrator.   
Output: Archived product record, shopping cart cleanup, inventory release.   
  
1.15 Category Creation Function   
Function ID: FR-15   
Description: Creates new product categories with hierarchy validation. Enforces depth limits and uniqueness rules.   
Input: Category name, description, parent category selection.   
Output: New category record, hierarchical structure update, audit log.   
  
1.16 Category View Function   
Function ID: FR-16   
Description: Displays category hierarchies with status indicators. Implements customer/admin view variations.   
Input: Category navigation request from UI.   
Output: Category tree structure with active/inactive status markers.   
  
1.17 Category Update Function   
Function ID: FR-17   
Description: Modifies category attributes and propagates changes to linked products. Validates hierarchical integrity.   
Input: Updated category fields from edit form.   
Output: Updated category record, product reassignment, audit trail.   
  
1.18 Category Deletion Function   
Function ID: FR-18   
Description: Archives categories after product reassignment. Manages nested subcategory deletions.   
Input: Category deletion request from administrator.   
Output: Archived category, product migration records, navigation updates.   
  
1.19 Shopping Cart Add Item Function   
Function ID: FR-19   
Description: Adds products to the cart with real-time inventory validation and quantity merging.   
Input: Product ID and quantity from "Add to Cart" action.   
Output: Updated cart contents, inventory reservation, cart summary update.   
  
1.20 Shopping Cart View Function   
Function ID: FR-20   
Description: Displays cart contents with calculated totals. Flags unavailable items.   
Input: Cart access request from authenticated customer.   
Output: Cart item list with prices, subtotal, availability warnings.   
  
1.21 Shopping Cart Update Function   
Function ID: FR-21   
Description: Modifies item quantities or removes items. Adjusts inventory reservations accordingly.   
Input: Cart modification request (quantity change/removal).   
Output: Updated cart state, inventory adjustment, recalculated totals.   
  
1.22 Checkout Function   
Function ID: FR-22   
Description: Processes orders through shipping, payment, and confirmation steps. Manages inventory reservations.   
Input: Checkout initiation from cart view.   
Output: New order record, payment authorization, inventory deduction.   
  
1.23 Payment Method Add Function   
Function ID: FR-23   
Description: Securely stores payment information using tokenization. Performs anti-fraud validation.   
Input: Payment card details and billing address.   
Output: Tokenized payment record, masked display data, audit log.   
  
1.24 Payment Method View Function   
Function ID: FR-24   
Description: Displays saved payment methods with strict PCI-compliant masking. Flags expired methods.   
Input: Payment method access request.   
Output: Masked payment details (e.g., "Visa •••• 1234"), expiration status.   
  
1.25 Payment Method Update Function   
Function ID: FR-25   
Description: Modifies non-sensitive payment details (expiration dates, billing addresses) with validation.   
Input: Updated payment fields from edit interface.   
Output: Updated payment record, verification requests, audit trail.   
  
1.26 Payment Method Delete Function   
Function ID: FR-26   
Description: Removes payment methods after transaction dependency checks. Revokes payment gateway tokens.   
Input: Payment method deletion request.   
Output: Payment record removal, token revocation, default payment reassignment.   
  
1.27 Order Confirmation Function   
Function ID: FR-27   
Description: Finalizes orders after payment capture and inventory deduction. Handles fraud verification workflows.   
Input: Validated checkout data from FR-22.   
Output: Confirmed order record, inventory deduction, payment capture.   
  
1.28 Order Confirmation Email Function   
Function ID: FR-28   
Description: Generates and sends order confirmation emails with secure access links to order details.   
Input: Confirmed order record from FR-27.   
Output: Order confirmation email, delivery status log.   
  
1.29 Order History View Function   
Function ID: FR-29   
Description: Displays order summaries with access controls. Supports date-based filtering and export.   
Input: Order history access request with search parameters.   
Output: Order summary list (ID, date, total), detailed order views.   
  
1.30 Order Status Update Function   
Function ID: FR-30   
Description: Modifies order states (e.g., Processing→Shipped) with business rule validation. Triggers notifications.   
Input: Status change request from admin/customer.   
Output: Updated order status, notification triggers, inventory adjustments.   
  
> \*Note: All 30 functions align with defined use cases and system entities. Inputs/outputs reference only defined artifacts. Each transformation is feasible within the e-commerce context.\*

# External Description

### Chapter 2: External Interfaces  
  
This chapter defines the external interfaces of the e-commerce system, categorizing them into user interfaces, hardware interfaces, software interfaces, and communication interfaces. Each interface is described with its role, interaction method, and relevant inputs/outputs.  
  
---  
  
#### 2.1 User Interfaces  
  
These interfaces facilitate direct interaction between users (customers or administrators) and the system.  
  
1. \*\*Customer Registration Form\*\*   
 - \*\*Description\*\*: A web form for new customers to input personal details.   
 - \*\*Role\*\*: Captures customer information for account creation.   
 - \*\*Interaction\*\*: Submits data to the system for verification and account setup.   
 - \*\*Output\*\*: redirects to a confirmation page or sends an email.  
  
2. \*\*Customer Login Interface\*\*   
 - \*\*Description\*\*: A web form for authenticating user credentials.   
 - \*\*Role\*\*: Verifies user身份 and grants access.   
 - \*\*Interaction\*\*: Validates email/username and password.   
 - \*\*Output\*\*: redirects to the customer dashboard or generates an audit log.  
  
3. \*\*Customer Dashboard\*\*   
 - \*\*Description\*\*: A personalized interface displaying account information and navigation options.   
 - \*\*Role\*\*: Provides access to profile, cart, orders, and support.   
 - \*\*Interaction\*\*: Displays data based on user actions.   
 - \*\*Output\*\*: Dynamic content updates based on user activity.  
  
4. \*\*Customer Profile Edit Form\*\*   
 - \*\*Description\*\*: A form for updating personal details.   
 - \*\*Role\*\*: Modifies user information securely.   
 - \*\*Interaction\*\*: Validates input fields and updates the database.   
 - \*\*Output\*\*: Confirms update success or prompts corrections.  
  
5. \*\*Administrator Console\*\*   
 - \*\*Description\*\*: An interface for managing system settings and user accounts.   
 - \*\*Role\*\*: Provides tools for CRUD operations on products, categories, users, and orders.   
 - \*\*Interaction\*\*: Secure access with 2FA for high-privilege actions.   
 - \*\*Output\*\*: Reflects real-time changes and updates.  
  
6. \*\*Product Catalog Interface\*\*   
 - \*\*Description\*\*: Displays available products with search and filtering options.   
 - \*\*Role\*\*: Facilitates product discovery and selection.   
 - \*\*Interaction\*\*: Supports sorting by category, price, etc.   
 - \*\*Output\*\*: Product listings with real-time stock status.  
  
7. \*\*Shopping Cart Interface\*\*   
 - \*\*Description\*\*: A summary page of selected products.   
 - \*\*Role\*\*: Manages cart contents and facilitates checkout.   
 - \*\*Interaction\*\*: Allows quantity adjustments and item removal.   
 - \*\*Output\*\*: Updated cart summary and inventory status.  
  
8. \*\*Checkout Interface\*\*   
 - \*\*Description\*\*: A multi-step process for order completion.   
 - \*\*Role\*\*: Captures shipping, payment, and order details.   
 - \*\*Interaction\*\*: Validates inputs and processes the order.   
 - \*\*Output\*\*: Confirms order placement and redirects to a confirmation page.  
  
9. \*\*Payment Gateway Interface\*\*   
 - \*\*Description\*\*: Secure interface for entering payment details.   
 - \*\*Role\*\*: Handles payment processing securely.   
 - \*\*Interaction\*\*: Tokenizes card information and validates transactions.   
 - \*\*Output\*\*: Confirms payment success or failure.  
  
10. \*\*Order History Interface\*\*   
 - \*\*Description\*\*: Displays past orders with filtering options.   
 - \*\*Role\*\*: Provides order tracking and management.   
 - \*\*Interaction\*\*: Allows viewing order details and status updates.   
 - \*\*Output\*\*: List of past orders with summary information.  
  
---  
  
#### 2.2 Hardware Interfaces  
  
No hardware interfaces are identified in the functional requirements.  
  
---  
  
#### 2.3 Software Interfaces  
  
These interfaces represent interactions with external systems, databases, or APIs.  
  
1. \*\*User Authentication Database\*\*   
 - \*\*Description\*\*: A database storing user credentials and session information.   
 - \*\*Role\*\*: Validates user身份 and manages sessions.   
 - \*\*Interaction\*\*: Receives authentication requests and returns session tokens.   
 - \*\*Output\*\*: Session data or authentication status.  
  
2. \*\*Product Inventory Database\*\*   
 - \*\*Description\*\*: A database tracking product stock levels.   
 - \*\*Role\*\*: Ensures real-time inventory accuracy.   
 - \*\*Interaction\*\*: Updates inventory upon product addition or removal.   
 - \*\*Output\*\*: Inventory status changes.  
  
3. \*\*Payment Gateway API\*\*   
 - \*\*Description\*\*: An API for secure payment processing.   
 - \*\*Role\*\*: Handles payment transactions and fraud detection.   
 - \*\*Interaction\*\*: Processes payment details and returns transaction status.   
 - \*\*Output\*\*: Payment authorization or decline.  
  
4. \*\*Order Management System (OMS)\*\*   
 - \*\*Description\*\*: A system managing order processing and fulfillment.   
 - \*\*Role\*\*: Tracks order status and updates inventory.   
 - \*\*Interaction\*\*: Receives order data and updates the system.   
 - \*\*Output\*\*: Order status updates and notifications.  
  
5. \*\*Email Service Provider API\*\*   
 - \*\*Description\*\*: An API for sending transactional emails.   
 - \*\*Role\*\*: Delivers confirmation, password reset, and other emails.   
 - \*\*Interaction\*\*: Sends email content with tracking capabilities.   
 - \*\*Output\*\*: Email delivery status.  
  
---  
  
#### 2.4 Communication Interfaces  
  
These interfaces involve interactions over networks or external channels.  
  
1. \*\*Account Confirmation Email\*\*   
 - \*\*Description\*\*: An email sent to verify account creation.   
 - \*\*Role\*\*: Confirms user registration and completes account setup.   
 - \*\*Interaction\*\*: Contains a link to activate the account.   
 - \*\*Output\*\*: Email with activation link and instructions.  
  
2. \*\*Order Confirmation Email\*\*   
 - \*\*Description\*\*: An email sent after order completion.   
 - \*\*Role\*\*: Informs the customer of order confirmation.   
 - \*\*Interaction\*\*: Includes order details and a secure access link.   
 - \*\*Output\*\*: Email with order summary and tracking information.  
  
3. \*\*Password Reset Email\*\*   
 - \*\*Description\*\*: An email sent when a user requests a password reset.   
 - \*\*Role\*\*: Facilitates password recovery.   
 - \*\*Interaction\*\*: Contains a link to reset the password.   
 - \*\*Output\*\*: Email with reset instructions and a time-limited link.  
  
4. \*\*Notification System\*\*   
 - \*\*Description\*\*: A system for sending order status updates and promotional offers.   
 - \*\*Role\*\*: Keeps users informed of system events.   
 - \*\*Interaction\*\*: Triggers notifications based on order status changes.   
 - \*\*Output\*\*: Push notifications or emails.  
  
---  
  
### Summary  
  
The e-commerce system's external interfaces are primarily user interfaces, with additional software and communication interfaces. These interfaces ensure secure and efficient interaction with users, external systems, and third-party services. Each interface is designed to provide a seamless user experience while maintaining system integrity and security.

# Use Case

### Use Case Description   
\*\*Use Case Name:\*\* Manage Customer   
\*\*Use Case ID:\*\* UC-01   
\*\*Actors:\*\*   
- Administrator   
- Customer   
  
\*\*Preconditions:\*\*   
1. The system is operational.   
2. Administrator is logged in (for administrative actions).   
3. Customer is registered and logged in (for self-updates).   
  
\*\*Postconditions:\*\*   
1. Customer data (e.g., profile, contact details) is added, modified, or deactivated as requested.   
2. Changes are audited and persisted in the system.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Customers" from the dashboard.   
2. System displays a list of registered customers.   
3. Administrator searches for a target customer.   
4. System returns matching customer records.   
5. Administrator selects a customer and chooses "Edit Profile".   
6. System displays the customer’s current details (name, contact, address).   
7. Administrator updates fields and confirms changes.   
8. System validates inputs and saves updates.   
9. System notifies Administrator of successful modification.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Customer Self-Update:\*\*   
&nbsp;&nbsp;1. Customer logs in and navigates to "My Profile".   
&nbsp;&nbsp;2. Customer edits personal details (e.g., email/phone).   
&nbsp;&nbsp;3. System validates and saves changes without admin intervention.   
\*\*B. Deactivate Account:\*\*   
&nbsp;&nbsp;1. At step 5, Administrator selects "Deactivate Account".   
&nbsp;&nbsp;2. System prompts confirmation.   
&nbsp;&nbsp;3. Administrator confirms; system revokes customer access and archives data.   
\*\*C. Data Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 8, if inputs are invalid (e.g., duplicate email), system rejects changes and displays error.   
&nbsp;&nbsp;2. User corrects errors and resubmits.   
  
---  
\*Note: Consistent with e-commerce system entities (Customer, Order, Payment). Flows prioritize administrative actions while accommodating self-service updates.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Register Customer   
\*\*Use Case ID:\*\* UC-02   
\*\*Actors:\*\*   
- Customer   
- Administrator (optional)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer has access to registration interface (web/mobile).   
3. For administrator-assisted registration: Administrator is logged in.   
  
\*\*Postconditions:\*\*   
1. New customer account is created and stored in the system.   
2. Customer credentials are activated for system access.   
3. Default shopping cart is initialized for the new customer.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to registration page.   
2. System displays registration form (name, email, password, contact details).   
3. Customer completes required fields and submits.   
4. System validates data format and email uniqueness.   
5. System creates customer record and initializes shopping cart.   
6. System sends account confirmation email.   
7. Customer verifies email via link.   
8. System activates account and displays success message.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Administrator-Assisted Registration:\*\*   
&nbsp;&nbsp;1. Administrator selects "Register Customer" in admin panel.   
&nbsp;&nbsp;2. Administrator inputs customer details manually.   
&nbsp;&nbsp;3. System auto-generates temporary password and creates account.   
&nbsp;&nbsp;4. Customer receives activation email to set permanent password.   
\*\*B. Duplicate Email Detection:\*\*   
&nbsp;&nbsp;1. At step 4, if email exists, system rejects submission.   
&nbsp;&nbsp;2. Customer is prompted to use "Forgot Password" or new email.   
\*\*C. Email Verification Timeout:\*\*   
&nbsp;&nbsp;1. If unverified after 24 hours, system marks account as pending.   
&nbsp;&nbsp;2. Customer can request new verification email from login page.   
  
---  
\*Note: Aligns with Customer/Shopping Cart entities. Self-registration is primary flow, with admin support for exceptions. Verification ensures data integrity.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Login Customer   
\*\*Use Case ID:\*\* UC-03   
\*\*Actors:\*\*   
- Customer   
- Administrator (optional for support)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer has completed registration (UC-02) and possesses valid credentials.   
3. Customer account is active and not locked.   
  
\*\*Postconditions:\*\*   
1. Customer is authenticated and granted access to the system.   
2. Customer session is established with appropriate permissions.   
3. Login activity is recorded in audit logs.   
  
\*\*Main Flow:\*\*   
1. Customer accesses system login interface.   
2. System displays authentication form (email/username + password).   
3. Customer enters valid credentials and submits.   
4. System verifies credentials against stored records.   
5. System validates account status (active/unlocked).   
6. System initiates authenticated session.   
7. System redirects customer to personalized dashboard.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Invalid Credentials:\*\*   
&nbsp;&nbsp;1. At step 4, if credentials mismatch, system displays error.   
&nbsp;&nbsp;2. Customer retries (max 3 attempts before temporary lockout).   
\*\*B. Password Recovery:\*\*   
&nbsp;&nbsp;1. Customer selects "Forgot Password" before step 3.   
&nbsp;&nbsp;2. System verifies registered email and sends reset link.   
&nbsp;&nbsp;3. Customer follows link to reset credentials (outside this flow).   
\*\*C. Locked Account:\*\*   
&nbsp;&nbsp;1. At step 5, if account is locked, system notifies customer.   
&nbsp;&nbsp;2. Customer must contact Administrator or wait for cooldown period.   
\*\*D. Administrator-Assisted Login:\*\*   
&nbsp;&nbsp;1. Administrator verifies customer identity via support channel.   
&nbsp;&nbsp;2. Administrator triggers temporary bypass in admin console.   
&nbsp;&nbsp;3. System forces password reset on next login.   
  
---  
\*Note: Aligns with Customer entity and security requirements. Integrates with account lifecycle (Register Customer UC-02). Session management enables access to Order/Shopping Cart functions.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Logout Customer   
\*\*Use Case ID:\*\* UC-04   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is currently logged in (authenticated session exists).   
  
\*\*Postconditions:\*\*   
1. Customer session is securely terminated.   
2. System access privileges for the customer are revoked.   
3. Logout event is recorded in audit logs.   
4. Customer is redirected to the public landing page.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Logout" option from the navigation interface.   
2. System displays logout confirmation prompt.   
3. Customer confirms logout request.   
4. System invalidates session tokens and clears authentication cookies.   
5. System records logout timestamp and customer ID in audit log.   
6. System redirects customer to the homepage with logout success notification.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Session Timeout:\*\*   
&nbsp;&nbsp;1. If customer remains inactive beyond system timeout threshold (e.g., 30 minutes), system automatically triggers logout.   
&nbsp;&nbsp;2. System performs steps 4-6 of Main Flow with "inactivity timeout" audit note.   
\*\*B. Multi-Device Logout:\*\*   
&nbsp;&nbsp;1. Customer selects "Logout from all devices" during confirmation (step 3).   
&nbsp;&nbsp;2. System revokes all active sessions linked to customer account.   
&nbsp;&nbsp;3. System forces re-authentication for any subsequent access attempts.   
\*\*C. Cancel Logout:\*\*   
&nbsp;&nbsp;1. At step 3, if customer cancels confirmation, system retains active session.   
&nbsp;&nbsp;2. System returns customer to previous interface state.   
  
---  
\*Note: Aligns with Customer entity and session security requirements. Completes authentication lifecycle initiated in Login Customer (UC-03). Audit trails support compliance monitoring.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Customer Profile   
\*\*Use Case ID:\*\* UC-05   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. If actor is Customer: Must be logged in with active account.   
3. If actor is Administrator: Must be logged in with "view profiles" privilege.   
4. Target customer profile exists in the system.   
  
\*\*Postconditions:\*\*   
1. Customer profile details are displayed in read-only format.   
2. Access event (viewer ID, timestamp) is recorded in audit logs.   
3. No modifications are made to profile data.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "My Profile" section OR Administrator selects "Browse Customers" in admin console.   
2. For Administrator: Searches/selects target customer from list.   
3. System retrieves customer profile data (name, contact, address, account status).   
4. System displays profile details with security-sensitive fields masked (e.g., password hashes).   
5. Actor reviews information and exits the view.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Profile Not Found:\*\*   
&nbsp;&nbsp;1. At step 3, if customer record is missing/deleted, system displays "Profile unavailable" error.   
&nbsp;&nbsp;2. Administrator may initiate registration recovery or terminate process.   
\*\*B. Unauthorized Access:\*\*   
&nbsp;&nbsp;1. If Customer attempts to access another customer's profile via URL manipulation, system blocks request.   
&nbsp;&nbsp;2. System logs security violation and displays "Access denied" alert.   
\*\*C. Limited Information Mode:\*\*   
&nbsp;&nbsp;1. For restricted administrator roles, system hides sensitive fields (e.g., payment methods).   
&nbsp;&nbsp;2. Display shows partial profile with "Restricted view" watermark.   
  
---  
\*Note: Integrates with Customer entity and audit requirements. Read-only operation complements Manage Customer (UC-01). Masking protects Payment Information privacy.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Customer Profile   
\*\*Use Case ID:\*\* UC-06   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Target customer profile exists in the system.   
3. For Administrator: Logged in with "edit profiles" privilege.   
4. For Customer: Logged in with active account.   
  
\*\*Postconditions:\*\*   
1. Customer profile fields are modified per request.   
2. Changes are audited (timestamp, actor, modified fields).   
3. Updated data is persisted across all related entities (e.g., Order, Payment).   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "My Profile" OR Administrator selects "Manage Customers" > "Edit Profile".   
2. System displays editable customer details (name, contact, address).   
3. Actor modifies required fields and confirms submission.   
4. System validates format and uniqueness (e.g., email/phone).   
5. System updates profile and synchronizes dependent entities.   
6. System notifies actor of successful update.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 4, if data violates rules (e.g., duplicate email), system rejects changes.   
&nbsp;&nbsp;2. System highlights errors; actor corrects and resubmits.   
\*\*B. Unauthorized Field Modification:\*\*   
&nbsp;&nbsp;1. If Customer attempts restricted edits (e.g., account status), system blocks action.   
&nbsp;&nbsp;2. System logs security violation and displays "Permission denied".   
\*\*C. Concurrent Update Conflict:\*\*   
&nbsp;&nbsp;1. If profile was modified since initial load, system detects version mismatch.   
&nbsp;&nbsp;2. System reloads latest data and prompts actor to re-apply changes.   
  
---  
\*Note: Aligns with Customer/Payment Information entities. Integrates with View Profile (UC-05) for data consistency. Audit trails comply with security policies.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Product Categories   
\*\*Use Case ID:\*\* UC-07   
\*\*Actors:\*\*   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "category management" privileges.   
  
\*\*Postconditions:\*\*   
1. Product category is created, modified, or deactivated as requested.   
2. Category changes are synchronized across all associated products.   
3. Audit log records category modification details.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Categories" in admin console.   
2. System displays hierarchical category tree with action buttons.   
3. Administrator chooses target category and selects "Edit".   
4. System displays category details (name, description, parent category).   
5. Administrator updates fields and confirms changes.   
6. System validates inputs (e.g., name uniqueness, hierarchy rules).   
7. System propagates changes to all linked products.   
8. System confirms successful update.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Create New Category:\*\*   
&nbsp;&nbsp;1. At step 3, Administrator selects "Add New".   
&nbsp;&nbsp;2. System displays empty category form.   
&nbsp;&nbsp;3. Administrator defines mandatory fields (name, hierarchy level).   
&nbsp;&nbsp;4. System generates new category ID and stores record.   
\*\*B. Delete Category:\*\*   
&nbsp;&nbsp;1. At step 3, Administrator selects "Delete".   
&nbsp;&nbsp;2. System verifies no products are assigned to category.   
&nbsp;&nbsp;3. Administrator confirms deletion; system archives category.   
\*\*C. Reassign Products:\*\*   
&nbsp;&nbsp;1. If deleted category contains products (step B2), system blocks deletion.   
&nbsp;&nbsp;2. System prompts Administrator to reassign products to new category.   
&nbsp;&nbsp;3. Products are migrated before category removal.   
\*\*D. Hierarchy Conflict:\*\*   
&nbsp;&nbsp;1. At step 6, if parent-child relationship violates rules, system rejects changes.   
&nbsp;&nbsp;2. System suggests valid hierarchy options based on existing structure.   
  
---  
\*Note: Aligns with Category/Product entities. Maintains data integrity across product catalog. Hierarchical management supports e-commerce navigation.\*  
  
Use Case Name: View Customer Login Log   
Use Case ID: UC-08   
Actors:   
- Administrator   
- Customer   
  
Preconditions:   
1. System is operational.   
2. Administrator is logged in with "view audit logs" privileges (for admin access).   
3. Customer is logged in with active account (for self-access).   
4. Login log data exists for the target customer.   
  
Postconditions:   
1. Login log entries are displayed in read-only format.   
2. Access event (viewer ID, timestamp) is recorded in audit logs.   
3. No modifications are made to login log data.   
  
Main Flow:   
1. Administrator selects "Audit Logs" > "Customer Login History" in admin console.   
2. System displays search filters (customer ID, date range, status).   
3. Administrator specifies criteria and executes search.   
4. System retrieves matching login records (timestamp, IP, device, status).   
5. System displays results in tabular format with pagination.   
6. Administrator reviews log entries.   
  
Alternative Flow:   
A. Customer Self-Access:   
&nbsp;&nbsp;1. Customer navigates to "Account Security" section.   
&nbsp;&nbsp;2. Customer selects "Login History".   
&nbsp;&nbsp;3. System automatically retrieves current customer's last 30-day logs.   
&nbsp;&nbsp;4. System displays simplified view (timestamp, device, status).   
B. No Matching Records:   
&nbsp;&nbsp;1. At step 4, if no logs match criteria, system displays "No records found".   
&nbsp;&nbsp;2. Administrator may adjust filters and retry.   
C. Restricted Field Access:   
&nbsp;&nbsp;1. For Customer self-access (flow A), sensitive fields (IP, exact location) are masked.   
&nbsp;&nbsp;2. System displays generic location (e.g., "Country/Region").   
D. Export Request:   
&nbsp;&nbsp;1. At step 5, Administrator selects "Export to CSV".   
&nbsp;&nbsp;2. System generates downloadable file with full log details.   
  
---  
\*Note: Integrates with Customer entity and audit requirements. Supports compliance monitoring (GDPR) through data masking. Historical logs retained per data retention policy.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Register Administrator   
\*\*Use Case ID:\*\* UC-09   
\*\*Actors:\*\*   
- System Administrator (existing privileged administrator)   
  
\*\*Preconditions:\*\*   
1. The system is operational.   
2. System Administrator is logged in with "admin management" privileges.   
3. Registration interface is accessible via admin console.   
  
\*\*Postconditions:\*\*   
1. New administrator account is created and stored in the system.   
2. Account activation notification is sent to the new administrator.   
3. Privilege assignments are audited and persisted.   
  
\*\*Main Flow:\*\*   
1. System Administrator selects "Manage Administrators" > "Register New Admin".   
2. System displays registration form (name, email, role, privilege level).   
3. System Administrator enters required details and submits.   
4. System validates email format and uniqueness.   
5. System generates encrypted credentials and stores admin record.   
6. System sends account activation email with setup instructions.   
7. System confirms successful registration and logs privilege assignments.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Duplicate Email Detection:\*\*   
&nbsp;&nbsp;1. At step 4, if email exists, system rejects submission.   
&nbsp;&nbsp;2. Administrator must modify email or retrieve existing account.   
\*\*B. Privilege Escalation Attempt:\*\*   
&nbsp;&nbsp;1. If assigned privilege exceeds current administrator's level, system blocks action.   
&nbsp;&nbsp;2. System logs security violation and displays "Insufficient Permissions".   
\*\*C. Incomplete Submission:\*\*   
&nbsp;&nbsp;1. If mandatory fields are missing at step 3, system prevents submission.   
&nbsp;&nbsp;2. System highlights missing fields with validation errors.   
  
---  
\*Note: Aligns with Administrator entity and security protocols. Activation workflow ensures only vetted personnel gain access. Audit trails track privilege assignments.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Login Administrator   
\*\*Use Case ID:\*\* UC-10   
\*\*Actors:\*\*   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator has completed registration (UC-09) and possesses valid credentials.   
3. Administrator account is active and not locked.   
  
\*\*Postconditions:\*\*   
1. Administrator is authenticated and granted administrative access.   
2. Administrator session is established with assigned privileges.   
3. Login event is recorded in audit logs.   
  
\*\*Main Flow:\*\*   
1. Administrator accesses dedicated admin login interface.   
2. System displays authentication form (email + password).   
3. Administrator enters credentials and submits.   
4. System verifies credentials against admin database.   
5. System validates account status and privileges.   
6. System initiates authenticated admin session.   
7. System redirects to administrative dashboard.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Invalid Credentials:\*\*   
&nbsp;&nbsp;1. At step 4, if credentials mismatch, system displays error.   
&nbsp;&nbsp;2. Administrator retries (max 5 attempts before lockout).   
\*\*B. Elevated Security Verification:\*\*   
&nbsp;&nbsp;1. For high-privilege accounts, system requires 2FA confirmation after step 5.   
&nbsp;&nbsp;2. Administrator completes second-factor authentication.   
\*\*C. Account Locked:\*\*   
&nbsp;&nbsp;1. At step 5, if account is locked, system notifies administrator.   
&nbsp;&nbsp;2. System requires intervention from System Administrator (UC-09 actor).   
\*\*D. Session Continuity:\*\*   
&nbsp;&nbsp;1. At step 7, system restores previous session context if valid re-login occurs within 2 hours.   
  
---  
\*Note: Aligns with Administrator entity and security protocols. Requires stricter access controls than customer login (UC-03). Audit trails include privilege level and access scope.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Logout Administrator   
\*\*Use Case ID:\*\* UC-11   
\*\*Actors:\*\*   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is currently logged in (active authenticated session exists).   
3. Administrative privileges are active for the session.   
  
\*\*Postconditions:\*\*   
1. Administrator session is securely terminated.   
2. Administrative access privileges are revoked.   
3. Logout event is recorded in audit logs with timestamp and administrator ID.   
4. Administrator is redirected to the public login interface.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Logout" from the admin console navigation menu.   
2. System displays a security confirmation dialog.   
3. Administrator confirms logout request.   
4. System invalidates session tokens and clears authentication cookies.   
5. System revokes administrative access permissions.   
6. System records logout event in admin audit log.   
7. System redirects administrator to the public login page with success notification.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Session Timeout:\*\*   
&nbsp;&nbsp;1. If admin remains inactive beyond security threshold (e.g., 10 minutes), system automatically triggers logout.   
&nbsp;&nbsp;2. System performs steps 4-7 of Main Flow with "inactivity timeout" audit note.   
\*\*B. Multi-Session Termination:\*\*   
&nbsp;&nbsp;1. At step 3, if administrator selects "Logout all sessions", system identifies all active sessions.   
&nbsp;&nbsp;2. System terminates all sessions associated with administrator account.   
&nbsp;&nbsp;3. System forces re-authentication for future access attempts.   
\*\*C. Cancel Logout:\*\*   
&nbsp;&nbsp;1. At step 3, if administrator cancels confirmation, system maintains active session.   
&nbsp;&nbsp;2. System returns administrator to previous admin console state.   
  
---  
\*Note: Aligns with Administrator entity and security protocols. Completes authentication lifecycle initiated in Login Administrator (UC-10). Session termination mechanisms prevent unauthorized privilege retention.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Administrator Profile   
\*\*Use Case ID:\*\* UC-12   
\*\*Actors:\*\*   
- Administrator (self-view or privileged view)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with active account.   
3. Target administrator profile exists in the system.   
4. Requester has "view admin profiles" privilege (for viewing other administrators).   
  
\*\*Postconditions:\*\*   
1. Administrator profile details are displayed in read-only format.   
2. Access event (viewer ID, timestamp) is recorded in audit logs.   
3. No modifications are made to profile data.   
  
\*\*Main Flow:\*\*   
1. Administrator navigates to "Admin Management" > "View Profiles".   
2. System displays administrator directory with search filters (name/email).   
3. Administrator selects target profile (self or others based on privileges).   
4. System retrieves profile data (name, email, role, privilege level, status).   
5. System displays details with security masking (e.g., password hashes).   
6. Administrator reviews information and exits the interface.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Self-Profile View:\*\*   
&nbsp;&nbsp;1. Administrator selects "My Profile" shortcut.   
&nbsp;&nbsp;2. System skips search/directory and directly displays requester's profile.   
\*\*B. Unauthorized Access:\*\*   
&nbsp;&nbsp;1. If requesting administrator lacks privileges for target profile, system blocks access.   
&nbsp;&nbsp;2. System logs security violation and displays "Insufficient Permissions".   
\*\*C. Profile Not Found:\*\*   
&nbsp;&nbsp;1. At step 4, if target profile is deleted/invalid, system shows "Profile Unavailable".   
&nbsp;&nbsp;2. System administrator may initiate recovery protocols separately.   
\*\*D. Restricted View Mode:\*\*   
&nbsp;&nbsp;1. For lower-privilege roles, system hides sensitive fields (e.g., API keys).   
&nbsp;&nbsp;2. Display shows partial profile with "Limited Access" indicator.   
  
---  
\*Note: Aligns with Administrator entity and security protocols. Complements Register Administrator (UC-09) and Login Administrator (UC-10). Audit trails enforce least-privilege principles.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Administrator Profile   
\*\*Use Case ID:\*\* UC-13   
\*\*Actors:\*\*   
- Administrator (self-update or privileged administrator updating others)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with active session.   
3. Target administrator profile exists in the system.   
4. Requester has appropriate privileges (self-update requires no additional privileges; updating others requires "admin management" rights).   
  
\*\*Postconditions:\*\*   
1. Administrator profile fields are modified as requested.   
2. Changes are audited (modifier ID, timestamp, modified fields).   
3. Updated data propagates to all administrative functions.   
4. Email verification initiated if contact information changed.   
  
\*\*Main Flow:\*\*   
1. Administrator navigates to "Admin Management" > "Update Profile".   
2. System displays administrator directory with search filters.   
3. Administrator selects target profile (self or other).   
4. System loads editable fields (name, contact, role assignments) with current values.   
5. Administrator modifies permitted fields and submits changes.   
6. System validates format, uniqueness, and privilege escalation rules.   
7. System updates profile record and synchronizes role-based permissions.   
8. System sends confirmation notification and logs audit trail.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Self-Update Shortcut:\*\*   
&nbsp;&nbsp;1. Administrator selects "My Profile" > "Edit".   
&nbsp;&nbsp;2. System skips directory search (step 2-3) and directly loads requester's profile.   
\*\*B. Privilege Violation:\*\*   
&nbsp;&nbsp;1. At step 5, if unauthorized field modification (e.g., elevating own privileges) is attempted, system blocks action.   
&nbsp;&nbsp;2. System logs security event and displays "Permission Denied".   
\*\*C. Contact Verification:\*\*   
&nbsp;&nbsp;1. If email/phone is modified, system sends verification code to new contact.   
&nbsp;&nbsp;2. Administrator must enter code within 10 minutes to activate changes.   
\*\*D. Concurrent Lock:\*\*   
&nbsp;&nbsp;1. If profile is being edited by another administrator, system prevents overlapping updates.   
&nbsp;&nbsp;2. System displays "Profile locked by [Admin ID]; try later".   
  
---  
\*Note: Aligns with Administrator entity and RBAC security protocols. Self-service updates enabled while enforcing privilege boundaries. Verification mechanisms prevent unauthorized contact changes.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Administrator Accounts   
\*\*Use Case ID:\*\* UC-14   
\*\*Actors:\*\*   
- System Administrator (with "admin management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. System Administrator is logged in with "admin management" privileges.   
3. Administrator accounts data exists in the system.   
  
\*\*Postconditions:\*\*   
1. Administrator accounts are created, modified, or deactivated as requested.   
2. Changes propagate to all system functions (login, permissions, audit).   
3. Actions are logged with timestamp and administrator ID.   
  
\*\*Main Flow:\*\*   
1. System Administrator selects "Manage Administrator Accounts" in admin console.   
2. System displays paginated list of administrator accounts (name, email, role, status).   
3. System Administrator searches/filters accounts by role or status.   
4. System Administrator selects target account and chooses action (Create/Edit/Deactivate).   
5. System executes selected operation and validates permissions.   
6. System updates account records and synchronizes dependent modules.   
7. System confirms successful operation and updates audit log.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Create New Account:\*\*   
&nbsp;&nbsp;1. At step 4, Administrator selects "Create New".   
&nbsp;&nbsp;2. System displays registration form (name, email, role, privileges).   
&nbsp;&nbsp;3. Administrator submits valid details; system generates credentials.   
&nbsp;&nbsp;4. System sends activation email to new administrator.   
\*\*B. Deactivate Account:\*\*   
&nbsp;&nbsp;1. At step 4, Administrator selects "Deactivate".   
&nbsp;&nbsp;2. System verifies target isn't last active System Administrator.   
&nbsp;&nbsp;3. System revokes sessions and sets status to inactive.   
&nbsp;&nbsp;4. System archives permissions and sends deactivation notification.   
\*\*C. Privilege Escalation Attempt:\*\*   
&nbsp;&nbsp;1. At step 5, if assigned privileges exceed current administrator's level, system blocks action.   
&nbsp;&nbsp;2. System logs security violation and displays "Insufficient Permissions".   
\*\*D. Reactivate Account:\*\*   
&nbsp;&nbsp;1. For inactive accounts in step 3, "Reactivate" option appears.   
&nbsp;&nbsp;2. System restores permissions and requires password reset on next login.   
  
---  
\*Note: Aligns with Administrator entity and RBAC security. Integrates with Login/Logout use cases (UC-10/UC-11). Deactivation prevents removal of last privileged administrator.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add Product   
\*\*Use Case ID:\*\* UC-15   
\*\*Actors:\*\*   
- Administrator (with "product management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "product management" privileges.   
3. At least one valid product category exists in the system (UC-07).   
  
\*\*Postconditions:\*\*   
1. New product record is created and stored in the system.   
2. Product is assigned to specified category and available in catalog.   
3. Audit log records product creation (timestamp, administrator ID).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Product Management" > "Add Product" in admin console.   
2. System displays product creation form (name, description, price, SKU, category selector, stock quantity).   
3. Administrator populates mandatory fields and selects target category.   
4. System validates data formats (numeric price, unique SKU, valid category).   
5. System generates product ID and stores record with "Active" status.   
6. System confirms successful creation and displays new product details.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Invalid Data Format:\*\*   
&nbsp;&nbsp;1. At step 4, if validation fails (e.g., negative price or duplicate SKU), system rejects submission.   
&nbsp;&nbsp;2. System highlights errors; administrator corrects fields and resubmits.   
\*\*B. Category Not Found:\*\*   
&nbsp;&nbsp;1. If selected category was deleted during form completion, system displays "Invalid category".   
&nbsp;&nbsp;2. Administrator must refresh category list and reselect valid category.   
\*\*C. Draft Saving:\*\*   
&nbsp;&nbsp;1. Administrator selects "Save Draft" at step 3 before validation.   
&nbsp;&nbsp;2. System stores incomplete product with "Draft" status for later editing.   
\*\*D. Bulk Image Upload:\*\*   
&nbsp;&nbsp;1. Administrator attaches multiple product images during step 3.   
&nbsp;&nbsp;2. System compresses and stores images with auto-generated thumbnails.   
  
---  
\*Note: Aligns with Product/Category entities. Requires valid category per UC-07. SKU uniqueness ensures inventory integrity. Draft mode supports complex product setups.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Product Details   
\*\*Use Case ID:\*\* UC-16   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Product exists in the database.   
3. For Administrator: Logged in with "view products" privilege.   
  
\*\*Postconditions:\*\*   
1. Product details are displayed in appropriate interface (customer/public or admin view).   
2. Access event is recorded in audit log for administrator actions.   
  
\*\*Main Flow:\*\*   
1. Actor navigates to product listing interface (catalog for customer, product management for admin).   
2. Actor selects a specific product from the list.   
3. System retrieves product details from database (name, description, price, images, category, stock status).   
4. System formats display based on actor type:   
&nbsp;&nbsp;- Customer view: Public details only   
&nbsp;&nbsp;- Admin view: Full details including SKU, cost, internal notes   
5. System displays product details page.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Product Unavailable:\*\*   
&nbsp;&nbsp;1. At step 3, if product is deleted OR (for customer) inactive, system blocks display.   
&nbsp;&nbsp;2. System shows "Product unavailable" message and returns to listing.   
\*\*B. Access Denied:\*\*   
&nbsp;&nbsp;1. At step 3, if administrator lacks "view products" privilege, system aborts retrieval.   
&nbsp;&nbsp;2. System logs security event and displays "Access Denied".   
\*\*C. Category Hierarchy Display:\*\*   
&nbsp;&nbsp;1. If product belongs to nested categories (UC-07), system displays breadcrumb navigation showing full hierarchy.   
  
---  
\*Note: Aligns with Product/Category entities. Customer view excludes sensitive data like cost/profit margins. Integrates with catalog navigation requirements.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Product Information   
\*\*Use Case ID:\*\* UC-17   
\*\*Actors:\*\*   
- Administrator (with "product management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "product management" privileges.   
3. Target product exists in the system.   
4. Valid product categories are available (UC-07).   
  
\*\*Postconditions:\*\*   
1. Product details are modified as specified.   
2. Changes propagate to catalog, orders, and shopping carts.   
3. Audit log records update (timestamp, modified fields, administrator ID).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Product Management" > "Browse Products" in admin console.   
2. System displays product list with search filters (name/SKU/category).   
3. Administrator locates target product and selects "Edit".   
4. System loads current product details (name, description, price, SKU, category, stock).   
5. Administrator modifies editable fields and confirms submission.   
6. System validates inputs (numeric price, unique SKU, valid category).   
7. System updates product record and synchronizes dependent entities.   
8. System confirms successful update and displays new product details.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 6, if data invalid (e.g., duplicate SKU or negative price), system rejects changes.   
&nbsp;&nbsp;2. System highlights errors; administrator corrects and resubmits.   
\*\*B. Category Conflict:\*\*   
&nbsp;&nbsp;1. If selected category was deleted during edit, system displays "Invalid category".   
&nbsp;&nbsp;2. Administrator must refresh category list and reselect valid category.   
\*\*C. Price Change Impact:\*\*   
&nbsp;&nbsp;1. When modifying price, system compares with existing shopping cart/orders.   
&nbsp;&nbsp;2. If discrepancy exists, system flags "Pending orders affected" warning.   
&nbsp;&nbsp;3. Administrator confirms override or adjusts impacted orders manually.   
\*\*D. Concurrent Update:\*\*   
&nbsp;&nbsp;1. If product was modified since initial load, system detects version conflict.   
&nbsp;&nbsp;2. System reloads latest data and prompts re-application of changes.   
  
---  
\*Note: Aligns with Product/Category/Order entities. SKU uniqueness maintains inventory integrity. Price change handling protects order consistency.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete Product   
\*\*Use Case ID:\*\* UC-18   
\*\*Actors:\*\*   
- Administrator (with "product management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "product management" privileges.   
3. Target product exists in the system database.   
4. Product is not currently in any pending/unshipped orders.   
  
\*\*Postconditions:\*\*   
1. Product is removed from active catalog and marked as deleted.   
2. Product references are cleared from all active shopping carts.   
3. Deletion event is recorded in audit logs (product ID, timestamp, administrator ID).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Product Management" > "Browse Products" in admin console.   
2. System displays product list with search filters (SKU/name/category).   
3. Administrator locates target product and selects "Delete".   
4. System displays confirmation dialog showing product impact summary.   
5. Administrator confirms deletion request.   
6. System verifies no pending orders contain this product.   
7. System removes product from active catalog and all shopping carts.   
8. System updates product status to "Archived".   
9. System notifies administrator of successful deletion.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Pending Orders Exist:\*\*   
&nbsp;&nbsp;1. At step 6, if product exists in pending orders, system blocks deletion.   
&nbsp;&nbsp;2. System displays "Cannot delete: Product is in [X] pending orders".   
&nbsp;&nbsp;3. Administrator must cancel affected orders first.   
\*\*B. Deletion Cancellation:\*\*   
&nbsp;&nbsp;1. At step 5, if administrator cancels confirmation, system aborts process.   
&nbsp;&nbsp;2. System returns to product management interface.   
\*\*C. Bulk Deletion:\*\*   
&nbsp;&nbsp;1. At step 2, administrator selects multiple products for deletion.   
&nbsp;&nbsp;2. System validates each product against pending orders separately.   
&nbsp;&nbsp;3. Only order-free products are deleted; others generate individual alerts.   
\*\*D. Reference Integrity Failure:\*\*   
&nbsp;&nbsp;1. If deletion violates category relationships (UC-07), system rolls back operation.   
&nbsp;&nbsp;2. System displays "Category dependency error: Reassign products first".   
  
---  
\*Note: Aligns with Product/Order/Shopping Cart entities. Soft delete (archiving) preserves historical order integrity. Requires validation against active transactions.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Browse Products   
\*\*Use Case ID:\*\* UC-19   
\*\*Actors:\*\*   
- Customer   
- Administrator (optional for monitoring)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Product catalog contains at least one active product.   
3. Customer has access to the product interface (no login required).   
  
\*\*Postconditions:\*\*   
1. Product listings are displayed according to applied criteria.   
2. No modifications are made to product data (read-only operation).   
3. Browse activity is optionally recorded in analytics logs.   
  
\*\*Main Flow:\*\*   
1. Customer accesses the product catalog interface.   
2. System retrieves and displays initial product listings (name, thumbnail, price) with default sorting/filters.   
3. Customer optionally enters search keywords or selects a product category.   
4. System updates display to show matching products.   
5. Customer browses paginated results.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Advanced Filtering:\*\*   
&nbsp;&nbsp;1. Customer applies price range/availability filters.   
&nbsp;&nbsp;2. System dynamically updates product list.   
\*\*B. Sort Modification:\*\*   
&nbsp;&nbsp;1. Customer selects sort option (e.g., "Price: Low to High").   
&nbsp;&nbsp;2. System reorders list accordingly.   
\*\*C. Empty Results:\*\*   
&nbsp;&nbsp;1. If no products match criteria, system displays "No products found".   
&nbsp;&nbsp;2. Customer adjusts search/filters.   
\*\*D. Quick View:\*\*   
&nbsp;&nbsp;1. Customer hovers over product for summary details.   
&nbsp;&nbsp;2. System displays pop-up with key attributes.   
\*\*E. Admin Monitoring:\*\*   
&nbsp;&nbsp;1. Administrator views live catalog from dashboard.   
&nbsp;&nbsp;2. System shows customer-facing interface without edit controls.   
  
---  
\*Note: Aligns with Product/Category entities. Read-only operation supports product discovery before View Product Details (UC-16). Default display includes pagination and thumbnail previews.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Product Categories   
\*\*Use Case ID:\*\* UC-07   
\*\*Actors:\*\*   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "category management" privileges.   
  
\*\*Postconditions:\*\*   
1. Product category is created, modified, or deactivated per request.   
2. Category changes propagate to all associated products.   
3. Audit log records modification details.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Categories" in admin console.   
2. System displays hierarchical category tree.   
3. Administrator selects target category and action (Edit/Create/Delete).   
4. For edits: System shows current details (name/description/parent).   
5. Administrator modifies fields and confirms.   
6. System validates inputs (name uniqueness/hierarchy rules).   
7. System propagates changes to linked products.   
8. System confirms successful operation.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Create New Category:\*\*   
&nbsp;&nbsp;1. At step 3, select "Add New".   
&nbsp;&nbsp;2. System displays empty category form.   
&nbsp;&nbsp;3. Administrator defines mandatory fields (name/hierarchy).   
&nbsp;&nbsp;4. System generates unique category ID.   
\*\*B. Delete Category:\*\*   
&nbsp;&nbsp;1. At step 3, select "Delete".   
&nbsp;&nbsp;2. System verifies no assigned products exist.   
&nbsp;&nbsp;3. Administrator confirms; system archives category.   
\*\*C. Reassign Products:\*\*   
&nbsp;&nbsp;1. If deletion blocked (products assigned), prompt reassignment.   
&nbsp;&nbsp;2. Administrator selects new category for affected products.   
&nbsp;&nbsp;3. System migrates products before deletion.   
\*\*D. Hierarchy Conflict:\*\*   
&nbsp;&nbsp;1. At validation (step 6), if parent-child violation occurs.   
&nbsp;&nbsp;2. System rejects change and suggests valid hierarchy options.   
  
---  
\*Note: Aligns with Category/Product entities. Ensures catalog consistency through change propagation. Hierarchical management supports multi-level navigation.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add Category   
\*\*Use Case ID:\*\* UC-20   
\*\*Actors:\*\*   
- Administrator (with "category management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "category management" privileges.   
3. Category management interface is accessible.   
  
\*\*Postconditions:\*\*   
1. New category record is created with unique ID.   
2. Category is available for immediate product assignment.   
3. Creation event is audited (timestamp, administrator ID).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Category Management" > "Add New Category" in admin console.   
2. System displays category creation form (name, description, parent category selector).   
3. Administrator populates mandatory fields and specifies hierarchy position.   
4. Administrator submits the form.   
5. System validates inputs (name uniqueness, hierarchy rules).   
6. System generates unique category ID and stores record.   
7. System confirms successful creation and displays new category details.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 5, if name conflicts with existing category or violates hierarchy rules.   
&nbsp;&nbsp;2. System rejects submission, highlights errors, and retains form data.   
\*\*B. Parent Category Invalidation:\*\*   
&nbsp;&nbsp;1. If selected parent category is deleted during form completion.   
&nbsp;&nbsp;2. System auto-refreshes hierarchy options and prompts reselection.   
\*\*C. Hierarchy Depth Limit:\*\*   
&nbsp;&nbsp;1. When nesting would exceed maximum depth (e.g., 5 levels).   
&nbsp;&nbsp;2. System blocks submission and suggests alternative placement.   
\*\*D. Draft Saving:\*\*   
&nbsp;&nbsp;1. Administrator saves incomplete form as draft.   
&nbsp;&nbsp;2. System stores partial data with "Draft" status for later completion.   
  
---  
\*Note: Aligns with Category entity and hierarchical structure requirements. Enforces uniqueness within hierarchy levels. Integrates with Product assignment workflows (UC-15).\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Categories   
\*\*Use Case ID:\*\* UC-21   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Administrator: Must be logged in with "view catalog" privileges.   
3. At least one category exists in the system (handled in alternative flow if not met).   
  
\*\*Postconditions:\*\*   
1. Category list is displayed in appropriate interface format.   
2. Access event is recorded in audit logs for administrative views.   
3. No modifications are made to category data.   
  
\*\*Main Flow:\*\*   
1. Actor navigates to category interface:   
 - Customer: Browses public catalog menu   
 - Administrator: Selects "View Categories" in admin console   
2. System retrieves active categories from database with hierarchy data.   
3. System displays category list:   
 - Customer: Active categories only with visual hierarchy   
 - Administrator: All categories (active/inactive) with status indicators   
4. Actor browses or searches categories (if search function invoked).   
  
\*\*Alternative Flow:\*\*   
\*\*A. No Categories Found:\*\*   
&nbsp;&nbsp;1. At step 2, if no categories exist, system displays:   
&nbsp;&nbsp;&nbsp;&nbsp;- Customer: "Categories coming soon" placeholder   
&nbsp;&nbsp;&nbsp;&nbsp;- Administrator: "No categories found. [Create New Category]" prompt   
\*\*B. Category Search (Administrator):\*\*   
&nbsp;&nbsp;1. Administrator enters search term in category filter.   
&nbsp;&nbsp;2. System retrieves categories matching name/description.   
&nbsp;&nbsp;3. Display updates to show filtered results.   
\*\*C. Restricted Access:\*\*   
&nbsp;&nbsp;1. If customer attempts admin category view via URL manipulation, system blocks access.   
&nbsp;&nbsp;2. System logs security violation and displays "Access Denied".   
\*\*D. Inactive Category Access:\*\*   
&nbsp;&nbsp;1. If customer accesses inactive category via direct link, system shows "Category unavailable".   
&nbsp;&nbsp;2. System suggests similar active categories.   
  
---  
\*Note: Aligns with Category entity and catalog requirements. Customer view excludes inactive categories. Hierarchical display supports navigation to Browse Products (UC-19).\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Category   
\*\*Use Case ID:\*\* UC-22   
\*\*Actors:\*\*   
- Administrator (with "category management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "category management" privileges.   
3. Target category exists in the system.   
4. Valid category hierarchy rules are defined (e.g., no circular references).   
  
\*\*Postconditions:\*\*   
1. Category attributes are updated as requested.   
2. Changes propagate to all associated products and navigation structures.   
3. Update event is recorded in audit logs (timestamp, modified fields, administrator ID).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Categories" in admin console.   
2. System displays hierarchical category tree with edit icons.   
3. Administrator selects target category and chooses "Edit".   
4. System loads current category details (name, description, parent category).   
5. Administrator modifies editable fields and confirms submission.   
6. System validates inputs (name uniqueness, hierarchy integrity).   
7. System updates category record and synchronizes all linked products.   
8. System confirms successful update and refreshes category tree.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 6, if name conflicts or hierarchy violates rules, system rejects changes.   
&nbsp;&nbsp;2. System highlights errors with correction suggestions; administrator resubmits.   
\*\*B. Parent Category Deletion:\*\*   
&nbsp;&nbsp;1. If selected parent category is deleted during edit, system auto-refreshes options.   
&nbsp;&nbsp;2. Administrator must reselect valid parent category.   
\*\*C. Concurrent Update Conflict:\*\*   
&nbsp;&nbsp;1. If category was modified by another administrator, system detects version mismatch.   
&nbsp;&nbsp;2. System reloads latest data and prompts re-application of changes.   
\*\*D. Impact Warning:\*\*   
&nbsp;&nbsp;1. When changing hierarchy, system calculates affected product count.   
&nbsp;&nbsp;2. If >100 products impacted, system requires explicit confirmation before propagation.   
  
---  
\*Note: Aligns with Category entity and UC-07/UC-20. Ensures catalog consistency through automatic propagation. Hierarchy validation prevents structural conflicts.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete Category   
\*\*Use Case ID:\*\* UC-23   
\*\*Actors:\*\*   
- Administrator (with "category management" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Administrator is logged in with "category management" privileges.   
3. Target category exists in the system.   
4. Category management interface is accessible.   
  
\*\*Postconditions:\*\*   
1. Category is archived and removed from active catalog.   
2. Products previously assigned to the category are either reassigned or handled per policy.   
3. Deletion event is recorded in audit logs (category ID, timestamp, administrator ID).   
4. Navigation structures are updated across the system.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Categories" in admin console.   
2. System displays hierarchical category tree with action controls.   
3. Administrator selects target category and chooses "Delete".   
4. System verifies no active products are assigned to the category.   
5. System displays confirmation dialog showing impact summary.   
6. Administrator confirms deletion request.   
7. System archives category record and updates hierarchy relationships.   
8. System propagates changes to all dependent modules (catalog, products).   
9. System notifies administrator of successful deletion and updates audit log.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Products Assigned to Category:\*\*   
&nbsp;&nbsp;1. At step 4, if active products exist, system blocks deletion.   
&nbsp;&nbsp;2. System displays "Reassign required: [X] products assigned" alert.   
&nbsp;&nbsp;3. Administrator either:   
&nbsp;&nbsp;&nbsp;&nbsp;a) Cancels deletion OR   
&nbsp;&nbsp;&nbsp;&nbsp;b) Selects "Reassign Products" to initiate transfer workflow.   
\*\*B. Subcategory Deletion Cascade:\*\*   
&nbsp;&nbsp;1. If target category has subcategories, system identifies nested structure.   
&nbsp;&nbsp;2. System requires explicit confirmation to delete entire hierarchy.   
&nbsp;&nbsp;3. Administrator verifies subcategory impacts before final confirmation.   
\*\*C. Orphaned Products Handling:\*\*   
&nbsp;&nbsp;1. During product reassignment (flow A3b), system auto-suggests similar categories.   
&nbsp;&nbsp;2. Administrator selects new target category; system migrates all products.   
&nbsp;&nbsp;3. Migration completion triggers category deletion (resumes at step 6).   
\*\*D. Concurrent Modification Conflict:\*\*   
&nbsp;&nbsp;1. If category was modified during deletion process, system detects version mismatch.   
&nbsp;&nbsp;2. System reloads current category state and prompts restart of deletion flow.   
  
---  
\*Note: Aligns with Category/Product entities. Requires empty category per UC-07 validation rules. Soft delete preserves referential integrity for historical order data.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Product Categories   
\*\*Use Case ID:\*\* UC-24   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer has access to the catalog interface (no login required).   
3. For Administrator: Must be logged in with "view catalog" privileges.   
4. At least one active category exists (handled in alternative flow).   
  
\*\*Postconditions:\*\*   
1. Product categories are displayed in hierarchical format.   
2. Access event is recorded in audit logs for administrative views.   
3. No modifications are made to category data.   
  
\*\*Main Flow:\*\*   
1. Actor navigates to category interface:   
 - Customer: Accesses public catalog menu   
 - Administrator: Selects "Product Categories" in admin console   
2. System retrieves active categories with parent-child hierarchy data.   
3. System displays category tree:   
 - Customer: Visual hierarchy with category names   
 - Administrator: Detailed view with status indicators (active/inactive)   
4. Actor browses categories or expands/collapses hierarchy nodes.   
  
\*\*Alternative Flow:\*\*   
\*\*A. No Categories Exist:\*\*   
&nbsp;&nbsp;1. System displays "No categories available" placeholder.   
&nbsp;&nbsp;2. Administrator sees "Create New Category" prompt with action link.   
\*\*B. Search Filtering (Administrator):\*\*   
&nbsp;&nbsp;1. Administrator enters search term in category filter.   
&nbsp;&nbsp;2. System highlights matching categories in hierarchy.   
\*\*C. Inactive Category Access:\*\*   
&nbsp;&nbsp;1. Customer accesses inactive category via direct URL.   
&nbsp;&nbsp;2. System displays "Category unavailable" with suggested active alternatives.   
\*\*D. Breadcrumb Navigation:\*\*   
&nbsp;&nbsp;1. Actor selects leaf category.   
&nbsp;&nbsp;2. System displays hierarchical path (e.g., Electronics > Mobile > Accessories).   
  
---  
\*Note: Aligns with Category entity and UC-07/UC-21. Customer view excludes inactive categories. Hierarchical display supports navigation to Browse Products (UC-19).\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Order History   
\*\*Use Case ID:\*\* UC-25   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account.   
3. For Administrator: Logged in with "view orders" privilege.   
4. Order records exist in the system database.   
  
\*\*Postconditions:\*\*   
1. Order history records are displayed in read-only format.   
2. Access event (viewer ID, timestamp, customer ID) is recorded in audit logs.   
3. No modifications are made to order data.   
  
\*\*Main Flow:\*\*   
1. Actor accesses order history interface:   
 - Customer: Navigates to "Order History" in account dashboard.   
 - Administrator: Selects "Manage Orders" > "View History" in admin console.   
2. For Administrator:   
 a. Enters search criteria (customer email, order ID, date range).   
 b. System returns matching customer records.   
 c. Administrator selects target customer.   
3. System retrieves order history for target customer (auto-selected for Customer actor).   
4. System displays order summaries (order ID, date, total amount, status) with pagination.   
5. Actor selects an order to view details.   
6. System displays detailed order information (products, quantities, payment method, shipping address).   
  
\*\*Alternative Flow:\*\*   
\*\*A. No Orders Found:\*\*   
&nbsp;&nbsp;1. At step 3, if no orders exist, system displays "No order history found".   
&nbsp;&nbsp;2. Actor may adjust search criteria (Administrator) or exit interface.   
\*\*B. Unauthorized Access:\*\*   
&nbsp;&nbsp;1. If Customer attempts to access another customer's orders via URL manipulation, system blocks request.   
&nbsp;&nbsp;2. System logs security violation and displays "Access Denied".   
\*\*C. Restricted Data View:\*\*   
&nbsp;&nbsp;1. For Customer view, system masks sensitive fields (full payment card numbers, security codes).   
&nbsp;&nbsp;2. Displays partial payment method (e.g., "Visa •••• 1234").   
\*\*D. Export Request (Administrator):\*\*   
&nbsp;&nbsp;1. At step 4, Administrator selects "Export to CSV".   
&nbsp;&nbsp;2. System generates downloadable file with full order details for selected range.   
  
---  
\*Note: Aligns with Order/Payment Information/Customer entities. Integrates with Browse Products (UC-19) and Payment workflows. Access control separates customer self-service from administrative oversight.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Order Details   
\*\*Use Case ID:\*\* UC-26   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Target order exists in the system.   
3. For Customer: Logged in with active account and order belongs to them.   
4. For Administrator: Logged in with "view orders" privilege.   
  
\*\*Postconditions:\*\*   
1. Detailed order information is displayed in read-only format.   
2. Access event (viewer ID, order ID) is recorded in audit logs.   
3. No modifications are made to order data.   
  
\*\*Main Flow:\*\*   
1. Customer selects an order from "My Orders" history OR Administrator searches for order via "Order Management".   
2. Actor selects target order from results.   
3. System retrieves order details:   
 - Order ID, date, status   
 - Products (name, quantity, price)   
 - Payment method and amount   
 - Shipping/billing addresses   
4. System formats display:   
 - Customer view: Masks sensitive payment details   
 - Administrator view: Shows full payment information   
5. System displays comprehensive order summary page.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Order Not Found:\*\*   
&nbsp;&nbsp;1. At step 3, if order is invalid/deleted, system shows "Order unavailable".   
&nbsp;&nbsp;2. Actor returns to order list interface.   
\*\*B. Unauthorized Customer Access:\*\*   
&nbsp;&nbsp;1. If Customer attempts to view another customer's order, system blocks access.   
&nbsp;&nbsp;2. System logs security violation and displays "Access Denied".   
\*\*C. Payment Data Masking:\*\*   
&nbsp;&nbsp;1. For Customer view, system displays partial payment details (e.g., "Visa •••• 1234").   
\*\*D. Order Status Conflicts:\*\*   
&nbsp;&nbsp;1. If order is in "Processing" state during retrieval, system shows real-time status banner.   
&nbsp;&nbsp;2. Actor may refresh page for updates.   
  
---  
\*Note: Aligns with Order/Payment Information/Product entities. Customer view restricts sensitive data per privacy policies. Integrates with Order History (UC-25) for contextual navigation.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Order Status   
\*\*Use Case ID:\*\* UC-27   
\*\*Actors:\*\*   
- Administrator   
- Customer (for cancellations only)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Target order exists in the system with valid status.   
3. Administrator is logged in with "manage orders" privilege (for admin updates).   
4. Customer is logged in and owns the order (for cancellations only).   
  
\*\*Postconditions:\*\*   
1. Order status is updated to the new value.   
2. Status change is recorded in audit logs (actor, timestamp, old/new status).   
3. Dependent processes are triggered (e.g., restock on cancellation, shipment notification).   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Orders" in admin console.   
2. System displays orders with search filters (ID, customer, status).   
3. Administrator locates target order and selects "Update Status".   
4. System shows current status and valid transitions (e.g., "Processing" → "Shipped").   
5. Administrator selects new status and confirms.   
6. System validates status transition rules.   
7. System updates order status and triggers dependent actions:   
 - For "Shipped": Generates tracking email   
 - For "Cancelled": Restocks inventory   
8. System logs change and confirms success.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Customer-Initiated Cancellation:\*\*   
&nbsp;&nbsp;1. Customer selects "My Orders" and chooses target order.   
&nbsp;&nbsp;2. Customer clicks "Cancel Order" (visible only if cancellable).   
&nbsp;&nbsp;3. System verifies cancellation window (e.g., <24h after payment).   
&nbsp;&nbsp;4. Customer confirms; system updates status to "Cancelled" and restocks items.   
\*\*B. Invalid Status Transition:\*\*   
&nbsp;&nbsp;1. At step 6, if transition violates rules (e.g., "Delivered" → "Processing"), system blocks update.   
&nbsp;&nbsp;2. System displays allowed transitions and error details.   
\*\*C. Notification Failure:\*\*   
&nbsp;&nbsp;1. If shipment email fails at step 7, system retries twice and logs alert.   
&nbsp;&nbsp;2. Status update remains valid despite notification failure.   
\*\*D. Restock Exceptions:\*\*   
&nbsp;&nbsp;1. During cancellation, if products are discontinued, system skips restock.   
&nbsp;&nbsp;2. System logs inventory exception for manual review.   
  
---  
\*Note: Aligns with Order/Payment Information/Product entities. Status transitions enforce business rules (e.g., no cancellation post-shipment). Integrates with inventory management and notification systems.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete Order   
\*\*Use Case ID:\*\* UC-28   
\*\*Actors:\*\*   
- Administrator (with "manage orders" privileges)   
- Customer (for own orders under cancellable conditions)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Target order exists in the system.   
3. Order is in cancellable state (e.g., "Pending" or "Processing" status).   
4. Administrator is logged in with "manage orders" privileges OR Customer is logged in and owns the order.   
  
\*\*Postconditions:\*\*   
1. Order is archived with "Cancelled" status.   
2. Ordered products are restocked to inventory.   
3. Payment refund is initiated if applicable.   
4. Deletion event is recorded in audit logs.   
  
\*\*Main Flow:\*\*   
1. Administrator selects "Manage Orders" in admin console OR Customer navigates to "My Orders".   
2. Actor searches/locates target order using filters (ID/customer/date).   
3. Actor selects "Delete" action.   
4. System displays order details and deletion consequences (restocking, refund).   
5. Actor confirms deletion request.   
6. System verifies order cancellability (status/time constraints).   
7. System updates order status to "Cancelled" and archives record.   
8. System restocks products to inventory.   
9. If payment was captured, system triggers refund process.   
10. System logs deletion event and confirms success.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Non-Cancellable Order:\*\*   
&nbsp;&nbsp;1. At step 6, if order status prohibits deletion (e.g., "Shipped"), system blocks action.   
&nbsp;&nbsp;2. System displays "Order cannot be deleted" error with reason.   
\*\*B. Refund Failure:\*\*   
&nbsp;&nbsp;1. At step 9, if refund initiation fails, system notifies support team.   
&nbsp;&nbsp;2. Order cancellation proceeds but refund status marked "Pending Manual Review".   
\*\*C. Partial Restock:\*\*   
&nbsp;&nbsp;1. If ordered products are discontinued, system skips restock.   
&nbsp;&nbsp;2. System logs inventory exception for manual resolution.   
\*\*D. Timeout Conflict:\*\*   
&nbsp;&nbsp;1. If order status changes during confirmation, system reloads data.   
&nbsp;&nbsp;2. Actor must restart deletion process with updated status.   
  
---  
\*Note: Aligns with Order/Payment Information/Product entities. Soft delete preserves historical records. Restocking logic integrates with inventory management.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add to Shopping Cart   
\*\*Use Case ID:\*\* UC-29   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Target product exists and is available for purchase.   
  
\*\*Postconditions:\*\*   
1. Selected product is added to the customer’s shopping cart.   
2. Cart item count and total value are updated.   
3. Product inventory is temporarily reserved (if applicable).   
  
\*\*Main Flow:\*\*   
1. Customer browses products and selects a target product.   
2. System displays product details page with "Add to Cart" option.   
3. Customer selects desired quantity and clicks "Add to Cart".   
4. System verifies product availability and inventory levels.   
5. System adds the product with selected quantity to the customer’s shopping cart.   
6. System updates cart summary (item count, subtotal) in the UI.   
7. System displays confirmation message with cart access options.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Insufficient Stock:\*\*   
&nbsp;&nbsp;1. At step 4, if inventory < requested quantity, system displays "Only [X] units available".   
&nbsp;&nbsp;2. Customer adjusts quantity or cancels action.   
\*\*B. Existing Item in Cart:\*\*   
&nbsp;&nbsp;1. If product already exists in cart, system merges quantities.   
&nbsp;&nbsp;2. System displays "Quantity updated: now [Y] in cart".   
\*\*C. Session Expiry:\*\*   
&nbsp;&nbsp;1. If session times out during process, system requires re-authentication.   
&nbsp;&nbsp;2. Cart state is preserved after successful login.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Inventory reservation prevents overselling. Integrates with Browse Products (UC-19) and Checkout workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Shopping Cart Contents   
\*\*Use Case ID:\*\* UC-30   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Shopping cart exists for the customer (initialized during registration per UC-02).   
  
\*\*Postconditions:\*\*   
1. Shopping cart contents are displayed in read-only format.   
2. No modifications are made to cart items or quantities.   
3. Cart access event is recorded in audit logs.   
  
\*\*Main Flow:\*\*   
1. Customer selects "View Cart" from navigation interface.   
2. System retrieves current shopping cart items (product IDs, names, quantities, prices).   
3. System calculates subtotal, taxes, and applicable discounts.   
4. System displays cart contents with product thumbnails, details, and total summary.   
5. Customer reviews items and exits cart view.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Empty Cart:\*\*   
&nbsp;&nbsp;1. If no items exist in cart, system displays "Your cart is empty" message.   
&nbsp;&nbsp;2. System suggests popular products with "Add to Cart" prompts.   
\*\*B. Product Unavailable:\*\*   
&nbsp;&nbsp;1. If stored product is discontinued or out-of-stock, system flags item with "Not Available" warning.   
&nbsp;&nbsp;2. System disables checkout option until issue is resolved.   
\*\*C. Price Change Conflict:\*\*   
&nbsp;&nbsp;1. If product price differs from stored cart value, system displays both prices with update prompt.   
&nbsp;&nbsp;2. Customer must manually refresh cart to adopt current pricing.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Read-only operation supports cart management workflows. Availability checks prevent downstream checkout failures.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Shopping Cart   
\*\*Use Case ID:\*\* UC-31   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Shopping cart exists and contains at least one item.   
4. Products in cart are available for modification.   
  
\*\*Postconditions:\*\*   
1. Shopping cart contents are updated per customer request.   
2. Cart totals (quantity, subtotal) are recalculated and persisted.   
3. Product inventory reservations are adjusted accordingly.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "Shopping Cart" view.   
2. System displays cart contents (products, quantities, prices).   
3. Customer modifies quantity of a specific item or selects "Remove" action.   
4. System validates inventory availability for new quantities.   
5. System updates cart item quantities or removes selected items.   
6. System recalculates cart totals and refreshes display.   
7. System confirms successful update.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Remove Item:\*\*   
&nbsp;&nbsp;1. At step 3, Customer selects "Remove Item".   
&nbsp;&nbsp;2. System removes item immediately without quantity validation.   
&nbsp;&nbsp;3. Proceed to step 6.   
\*\*B. Clear Entire Cart:\*\*   
&nbsp;&nbsp;1. Customer selects "Empty Cart" option.   
&nbsp;&nbsp;2. System clears all items after confirmation.   
&nbsp;&nbsp;3. System displays "Cart is empty" status.   
\*\*C. Inventory Shortage:\*\*   
&nbsp;&nbsp;1. At step 4, if requested quantity > available stock.   
&nbsp;&nbsp;2. System auto-adjusts to maximum available quantity.   
&nbsp;&nbsp;3. Displays warning with stock limit.   
\*\*D. Concurrent Modification:\*\*   
&nbsp;&nbsp;1. If cart was updated by another session during process.   
&nbsp;&nbsp;2. System reloads latest cart state and prompts re-application.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Real-time inventory validation prevents over-selling. Integrates with Checkout workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete from Shopping Cart   
\*\*Use Case ID:\*\* UC-32   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Shopping cart exists and contains at least one item.   
  
\*\*Postconditions:\*\*   
1. Selected item(s) are permanently removed from the shopping cart.   
2. Cart totals (quantity and subtotal) are recalculated and persisted.   
3. Inventory reservations for removed items are released.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "Shopping Cart" view.   
2. System displays cart contents (products, quantities, prices).   
3. Customer selects "Remove" action for a specific item.   
4. System permanently deletes the item from the cart.   
5. System recalculates cart totals and refreshes display.   
6. System shows "Item removed" confirmation message.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Remove Multiple Items:\*\*   
&nbsp;&nbsp;1. Customer selects multiple items using checkboxes.   
&nbsp;&nbsp;2. Customer clicks "Remove Selected Items".   
&nbsp;&nbsp;3. System deletes all selected items and recalculates totals.   
\*\*B. Clear Entire Cart:\*\*   
&nbsp;&nbsp;1. Customer selects "Empty Cart" option.   
&nbsp;&nbsp;2. System deletes all items after confirmation.   
&nbsp;&nbsp;3. System displays "Cart is empty" status.   
\*\*C. Concurrent Removal Conflict:\*\*   
&nbsp;&nbsp;1. If item was already removed by another session.   
&nbsp;&nbsp;2. System displays "Item no longer available" warning.   
&nbsp;&nbsp;3. Cart view refreshes to show current state.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Inventory reservations release prevents stock blockage. Integrates with Update Shopping Cart (UC-31) and Checkout workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add Payment Information   
\*\*Use Case ID:\*\* UC-33   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account.   
3. For Administrator: Logged in with "manage payment methods" privileges.   
4. Customer profile exists in the system (if Administrator is acting).   
  
\*\*Postconditions:\*\*   
1. New payment method is securely stored and associated with the customer.   
2. Payment method is available for future transactions.   
3. Addition event is recorded in audit logs (actor, timestamp, method type).   
  
\*\*Main Flow:\*\*   
1. Actor initiates payment method addition:   
 - Customer: Navigates to "Payment Methods" in account settings   
 - Administrator: Selects target customer in admin console > "Add Payment Method"   
2. System displays secure entry form (card number, expiration, CVV, billing address).   
3. Actor enters valid payment details and submits.   
4. System validates format (Luhn algorithm, expiration date) and performs anti-fraud checks.   
5. System tokenizes sensitive data via payment gateway.   
6. System stores token and masked details (e.g., "Visa •••• 1234") with customer profile.   
7. System confirms successful addition and displays new payment method.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 4, if data invalid (e.g., expired card), system rejects submission.   
&nbsp;&nbsp;2. Highlights specific errors; actor corrects and resubmits.   
\*\*B. Duplicate Detection:\*\*   
&nbsp;&nbsp;1. If identical payment method exists, system blocks storage.   
&nbsp;&nbsp;2. Displays "Payment method already registered" warning.   
\*\*C. Gateway Connection Failure:\*\*   
&nbsp;&nbsp;1. At step 5, if payment gateway unreachable, system retries twice.   
&nbsp;&nbsp;2. After failures, displays "Temporarily unavailable; try later".   
\*\*D. Administrator Security Override:\*\*   
&nbsp;&nbsp;1. If Administrator adds without CVV (e.g., phone order), system requires dual approval.   
&nbsp;&nbsp;2. Second administrator verifies identity before forcing storage.   
  
---  
\*Note: Aligns with Payment Information/Customer entities. Tokenization minimizes PCI compliance scope. Integrates with Order (UC-25) and Shopping Cart (UC-29) workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* View Payment Details   
\*\*Use Case ID:\*\* UC-34   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account and has saved payment methods.   
3. For Administrator: Logged in with "view payment details" privilege.   
4. Payment information records exist for the target customer.   
  
\*\*Postconditions:\*\*   
1. Payment details are displayed in read-only format.   
2. Access event (viewer ID, timestamp) is recorded in audit logs.   
3. No modifications are made to payment data.   
  
\*\*Main Flow:\*\*   
1. Actor accesses payment interface:   
 - Customer: Navigates to "Payment Methods" in account settings   
 - Administrator: Selects target customer > "View Payment Details"   
2. System retrieves saved payment methods:   
 - Customer: Own payment records   
 - Administrator: Target customer's payment records   
3. System displays payment details:   
 - Masked card numbers (e.g., "Visa •••• 1234")   
 - Expiration dates and billing addresses   
 - Payment method type (credit/debit/e-wallet)   
4. Actor reviews displayed information.   
  
\*\*Alternative Flow:\*\*   
\*\*A. No Payment Methods Found:\*\*   
&nbsp;&nbsp;1. System displays "No saved payment methods" message.   
&nbsp;&nbsp;2. Customer sees "Add Payment Method" prompt; Administrator sees empty record notification.   
\*\*B. Unauthorized Access:\*\*   
&nbsp;&nbsp;1. Customer attempts to view another customer's payment details via URL manipulation.   
&nbsp;&nbsp;2. System blocks access and logs security violation.   
\*\*C. Expired Methods Highlighting:\*\*   
&nbsp;&nbsp;1. System flags expired payment methods with "Expired" warning icon.   
&nbsp;&nbsp;2. Customer receives "Update Payment Method" suggestion.   
\*\*D. Restricted Administrator View:\*\*   
&nbsp;&nbsp;1. For low-privilege administrators, system hides billing addresses.   
&nbsp;&nbsp;2. Displays "Restricted: Contact senior admin for full details".   
  
---  
\*Note: Aligns with Payment Information/Customer entities. Strict masking protects sensitive data per PCI compliance. Integrates with Add Payment Information (UC-33) for holistic payment management.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Update Payment Information   
\*\*Use Case ID:\*\* UC-35   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account and has saved payment methods.   
3. For Administrator: Logged in with "update payment methods" privileges.   
4. Target payment method exists and is associated with a valid customer.   
  
\*\*Postconditions:\*\*   
1. Payment method details (expiration date, billing address) are updated.   
2. Audit log records modification (timestamp, actor, modified fields).   
3. Updated payment method remains available for transactions.   
4. Sensitive data (card number) remains tokenized and secure.   
  
\*\*Main Flow:\*\*   
1. Actor accesses payment interface:   
 - Customer: Navigates to "Payment Methods" in account settings   
 - Administrator: Selects target customer > "Payment Methods"   
2. Actor selects a saved payment method and chooses "Edit".   
3. System displays editable fields: expiration date and billing address (card number masked as "•••• 1234").   
4. Actor updates expiration date and/or billing address and submits.   
5. System validates:   
 a. Expiration date format/future date   
 b. Billing address completeness   
6. System updates payment record.   
7. System confirms successful update and displays updated details.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 5, if data invalid (e.g., expired date), system rejects changes.   
&nbsp;&nbsp;2. Highlights errors; actor corrects and resubmits.   
\*\*B. Card Number Update Request:\*\*   
&nbsp;&nbsp;1. If actor attempts full card replacement, system requires new tokenization via Add Payment (UC-33).   
&nbsp;&nbsp;2. Redirects to UC-33 flow after confirmation.   
\*\*C. Concurrent Modification:\*\*   
&nbsp;&nbsp;1. If payment method updated by another session, system detects conflict.   
&nbsp;&nbsp;2. Reloads latest data and prompts re-application of changes.   
\*\*D. Administrator Security Protocol:\*\*   
&nbsp;&nbsp;1. For high-risk updates (e.g., international billing address), system requires 2nd admin approval.   
&nbsp;&nbsp;2. Action pauses until dual verification completes.   
  
---  
\*Note: Aligns with Payment Information entity. Card numbers remain immutable per tokenization security standards. Integrates with Add Payment (UC-33) for full replacements.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Delete Payment Information   
\*\*Use Case ID:\*\* UC-36   
\*\*Actors:\*\*   
- Customer   
- Administrator (with "manage payment methods" privileges)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account and has saved payment methods.   
3. For Administrator: Logged in with "manage payment methods" privileges and target customer exists.   
4. Target payment method exists and is not linked to pending orders/subscriptions.   
  
\*\*Postconditions:\*\*   
1. Payment method is permanently removed from the system.   
2. Audit log records deletion (timestamp, actor, payment method ID).   
3. If deleted method was default payment option, another is assigned (if available).   
4. Associated tokens are revoked with payment gateway.   
  
\*\*Main Flow:\*\*   
1. Actor accesses payment interface:   
 - Customer: Navigates to "Payment Methods" in account settings   
 - Administrator: Selects target customer > "Payment Methods"   
2. System displays saved payment methods with delete icons.   
3. Actor selects payment method and chooses "Delete".   
4. System confirms deletion request showing masked details (e.g., "Visa •••• 1234").   
5. Actor confirms deletion.   
6. System verifies no pending orders/subscriptions use this payment method.   
7. System revokes payment gateway tokens and permanently deletes record.   
8. If method was default, system auto-assigns new default (if other methods exist).   
9. System logs deletion and confirms success to actor.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Pending Orders/Subscriptions:\*\*   
&nbsp;&nbsp;1. At step 6, if payment method is linked to active orders/subscriptions, system blocks deletion.   
&nbsp;&nbsp;2. Displays "Cannot delete: Linked to [X] active transactions".   
&nbsp;&nbsp;3. Actor must resolve dependencies before retrying.   
\*\*B. Last Payment Method:\*\*   
&nbsp;&nbsp;1. If deleted method is customer's only payment option, system skips default reassignment.   
&nbsp;&nbsp;2. Displays warning: "No remaining payment methods".   
\*\*C. Gateway Revocation Failure:\*\*   
&nbsp;&nbsp;1. At step 7, if token revocation fails, system retries twice and logs alert.   
&nbsp;&nbsp;2. Local record deleted regardless, payment gateway notified manually.   
\*\*D. Unauthorized Deletion Attempt:\*\*   
&nbsp;&nbsp;1. If Customer tries deleting another customer's payment data via URL manipulation, system blocks action.   
&nbsp;&nbsp;2. Logs security violation and displays "Access Denied".   
  
---  
\*Note: Aligns with Payment Information entity. Deletion blocked if payment method is transaction-dependent. Gateway integration ensures token cleanup per PCI-DSS requirements.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Manage Payment Information   
\*\*Use Case ID:\*\* UC-37   
\*\*Actors:\*\*   
- Customer   
- Administrator   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. For Customer: Logged in with active account.   
3. For Administrator: Logged in with "manage payment methods" privileges.   
4. Customer profile exists (if Administrator is acting).   
  
\*\*Postconditions:\*\*   
1. Payment method is added, updated, or deleted as requested.   
2. Changes are recorded in audit logs (actor, timestamp, action).   
3. Tokenized payment data remains PCI-compliant.   
  
\*\*Main Flow:\*\*   
1. Actor accesses payment management interface:   
 - Customer: Navigates to "Payment Methods" in account settings   
 - Administrator: Selects target customer → "Manage Payment Methods"   
2. System displays saved payment methods with masked details (e.g., "Visa •••• 1234").   
3. Actor selects action: Add/Edit/Delete.   
4. For Add/Edit: System displays secure input form.   
5. Actor submits changes with valid data.   
6. System validates inputs and processes through payment gateway.   
7. System updates records and confirms success.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Add New Method:\*\*   
&nbsp;&nbsp;1. At step 3, select "Add Payment Method".   
&nbsp;&nbsp;2. Enter card/billing details; system tokenizes via gateway.   
&nbsp;&nbsp;3. Store token + masked details; set as default if first method.   
\*\*B. Update Existing Method:\*\*   
&nbsp;&nbsp;1. At step 3, choose "Edit" on a payment method.   
&nbsp;&nbsp;2. Modify expiration date/billing address (card number immutable).   
&nbsp;&nbsp;3. System validates address format and future expiration.   
\*\*C. Delete Method:\*\*   
&nbsp;&nbsp;1. At step 3, choose "Delete" → confirm with masked details.   
&nbsp;&nbsp;2. System verifies no pending orders/subscriptions depend on method.   
&nbsp;&nbsp;3. Revoke gateway token and delete record.   
\*\*D. Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 6: Invalid data (e.g., expired card) → reject submission.   
&nbsp;&nbsp;2. Highlight errors; preserve form data for correction.   
\*\*E. Dependency Block:\*\*   
&nbsp;&nbsp;1. If payment method linked to active subscriptions → block deletion.   
&nbsp;&nbsp;2. Display: "Cannot delete: Active subscription(s) depend on this method".   
  
---  
\*Note: Aligns with Payment Information entity. Card numbers remain immutable per tokenization standards. Integrates with Order (UC-25) and Subscription workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Add to Cart   
\*\*Use Case ID:\*\* UC-29   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Target product exists and is available for purchase.   
  
\*\*Postconditions:\*\*   
1. Selected product is added to the customer’s shopping cart.   
2. Cart item count and total value are updated.   
3. Product inventory is temporarily reserved.   
  
\*\*Main Flow:\*\*   
1. Customer browses products and selects a target product.   
2. System displays product details page with "Add to Cart" option.   
3. Customer selects desired quantity and clicks "Add to Cart".   
4. System verifies product availability and inventory levels.   
5. System adds the product with selected quantity to the customer’s shopping cart.   
6. System updates cart summary (item count, subtotal) in the UI.   
7. System displays confirmation message with cart access options.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Insufficient Stock:\*\*   
&nbsp;&nbsp;1. At step 4, if inventory < requested quantity, system displays "Only [X] units available".   
&nbsp;&nbsp;2. Customer adjusts quantity or cancels action.   
\*\*B. Existing Item in Cart:\*\*   
&nbsp;&nbsp;1. If product already exists in cart, system merges quantities.   
&nbsp;&nbsp;2. System displays "Quantity updated: now [Y] in cart".   
\*\*C. Session Expiry:\*\*   
&nbsp;&nbsp;1. If session times out during process, system requires re-authentication.   
&nbsp;&nbsp;2. Cart state is preserved after successful login.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Inventory reservation prevents overselling. Integrates with Browse Products (UC-19) and Checkout workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Remove from Cart   
\*\*Use Case ID:\*\* UC-38   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Shopping cart exists and contains at least one item.   
  
\*\*Postconditions:\*\*   
1. Selected item(s) are permanently removed from the shopping cart.   
2. Cart totals (quantity and subtotal) are recalculated and persisted.   
3. Inventory reservations for removed items are released.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "Shopping Cart" view.   
2. System displays cart contents (products, quantities, prices).   
3. Customer selects "Remove" action for a specific item.   
4. System permanently deletes the item from the cart.   
5. System recalculates cart totals and refreshes display.   
6. System shows "Item removed" confirmation message.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Remove Multiple Items:\*\*   
&nbsp;&nbsp;1. Customer selects multiple items using checkboxes.   
&nbsp;&nbsp;2. Customer clicks "Remove Selected Items".   
&nbsp;&nbsp;3. System deletes all selected items and recalculates totals.   
\*\*B. Clear Entire Cart:\*\*   
&nbsp;&nbsp;1. Customer selects "Empty Cart" option.   
&nbsp;&nbsp;2. System deletes all items after confirmation.   
&nbsp;&nbsp;3. System displays "Cart is empty" status.   
\*\*C. Concurrent Removal Conflict:\*\*   
&nbsp;&nbsp;1. If item was already removed by another session.   
&nbsp;&nbsp;2. System displays "Item no longer available" warning.   
&nbsp;&nbsp;3. Cart view refreshes to show current state.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Inventory reservation releases prevent stock blockage. Integrates with Update Shopping Cart (UC-31) and Checkout workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Modify Cart Contents   
\*\*Use Case ID:\*\* UC-39   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with an active account.   
3. Shopping cart exists and contains at least one item.   
  
\*\*Postconditions:\*\*   
1. Cart contents are updated per modification request (quantity change or removal).   
2. Cart totals (item count, subtotal) are recalculated and persisted.   
3. Inventory reservations are adjusted for modified quantities.   
  
\*\*Main Flow:\*\*   
1. Customer navigates to "Shopping Cart" view.   
2. System displays cart contents (products, quantities, prices).   
3. Customer modifies quantity for a specific item or selects "Remove".   
4. For quantity updates:   
 &nbsp;&nbsp;a. System validates new quantity (>0) and inventory availability.   
 &nbsp;&nbsp;b. System updates item quantity.   
5. For removal: System deletes selected item.   
6. System recalculates cart totals and refreshes display.   
7. System confirms modification success.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Clear Entire Cart:\*\*   
&nbsp;&nbsp;1. Customer selects "Empty Cart".   
&nbsp;&nbsp;2. System removes all items after confirmation.   
&nbsp;&nbsp;3. System displays "Cart emptied" status.   
\*\*B. Invalid Quantity:\*\*   
&nbsp;&nbsp;1. At step 4a, if quantity ≤0, system rejects change.   
&nbsp;&nbsp;2. System reverts to original quantity with "Invalid input" error.   
\*\*C. Inventory Shortage:\*\*   
&nbsp;&nbsp;1. At step 4a, if requested quantity > stock, system auto-sets to max available.   
&nbsp;&nbsp;2. Displays "Only [X] units available; quantity adjusted".   
\*\*D. Concurrent Update Conflict:\*\*   
&nbsp;&nbsp;1. If cart modified by another session, system detects conflict.   
&nbsp;&nbsp;2. Reloads latest cart state and prompts re-application.   
  
---  
\*Note: Aligns with Shopping Cart/Product entities. Integrates inventory reservation logic to prevent overselling. Supports both granular updates and bulk operations.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Checkout   
\*\*Use Case ID:\*\* UC-40   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer is logged in with active account.   
3. Shopping cart contains ≥1 available product.   
4. Valid payment method exists on file or can be added.   
  
\*\*Postconditions:\*\*   
1. New order is created with unique ID and "Processing" status.   
2. Shopping cart is cleared.   
3. Payment authorization is processed.   
4. Inventory is reserved for ordered items.   
5. Order confirmation email is sent.   
  
\*\*Main Flow:\*\*   
1. Customer selects "Checkout" from cart view.   
2. System displays checkout steps: Review → Shipping → Payment → Confirm.   
3. Customer verifies cart items/quantities; system validates availability.   
4. Customer selects saved shipping address or enters new valid address.   
5. System calculates shipping costs and taxes.   
6. Customer selects saved payment method or adds new via embedded UC-33 flow.   
7. System validates payment details via tokenization.   
8. Customer reviews order summary (items, costs, total) and confirms.   
9. System reserves inventory and authorizes payment through gateway.   
10. Upon success:   
 a. Creates order record with "Processing" status   
 b. Clears shopping cart   
 c. Sends confirmation email   
11. System displays order confirmation page with ID and details.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Item Unavailable During Checkout:\*\*   
&nbsp;&nbsp;1. At step 3/9, if product stock < cart quantity, system removes item.   
&nbsp;&nbsp;2. Displays "Item unavailable: removed from order" warning.   
&nbsp;&nbsp;3. Customer confirms updated cart or cancels checkout.   
\*\*B. Payment Authorization Failure:\*\*   
&nbsp;&nbsp;1. At step 9, if payment declined, system sets order status to "Payment Failed".   
&nbsp;&nbsp;2. Preserves cart contents; displays "Declined: Try another method".   
&nbsp;&nbsp;3. Customer returns to payment step to retry.   
\*\*C. New Address Validation Failure:\*\*   
&nbsp;&nbsp;1. At step 4, if address incomplete/invalid, system blocks progression.   
&nbsp;&nbsp;2. Highlights errors; customer corrects fields.   
\*\*D. Express Checkout (Saved Preferences):\*\*   
&nbsp;&nbsp;1. If customer has default shipping/payment, system auto-populates steps 4-6.   
&nbsp;&nbsp;2. Customer skips to final confirmation (step 8).   
  
---  
\*Note: Integrates Shopping Cart (UC-29), Payment Information (UC-33), and Order entities. Inventory reservation prevents overselling. Tokenization ensures PCI compliance.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Confirm Order   
\*\*Use Case ID:\*\* UC-41   
\*\*Actors:\*\*   
- Customer   
- Administrator (optional for manual verification)   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Customer has completed checkout process (UC-40) with valid payment authorization.   
3. Order exists in "Processing" status with all required details (products, shipping, payment).   
4. For administrative override: Administrator is logged in with "order verification" privileges.   
  
\*\*Postconditions:\*\*   
1. Order status transitions to "Confirmed".   
2. Inventory is permanently deducted for ordered items.   
3. Payment capture is initiated with payment gateway.   
4. Order confirmation notification is sent to customer.   
5. Order record is finalized in the system.   
  
\*\*Main Flow:\*\*   
1. System automatically validates order integrity after checkout (inventory availability, payment pre-authorization).   
2. System updates order status to "Confirmed".   
3. System permanently deducts inventory quantities for all ordered items.   
4. System initiates payment capture through payment gateway using stored token.   
5. System generates order confirmation number.   
6. System sends email notification to customer with order details and confirmation number.   
7. System logs confirmation timestamp and transitions order to fulfillment workflow.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Manual Verification Required:\*\*   
&nbsp;&nbsp;1. If fraud risk detected (e.g., high-value order, new customer), system flags order for admin review.   
&nbsp;&nbsp;2. Administrator verifies order details and payment legitimacy.   
&nbsp;&nbsp;3. Administrator manually triggers confirmation via admin console.   
\*\*B. Inventory Shortage Post-Checkout:\*\*   
&nbsp;&nbsp;1. If inventory insufficient at confirmation time, system holds order in "Pending" status.   
&nbsp;&nbsp;2. System notifies inventory management and customer about delay.   
&nbsp;&nbsp;3. Confirmation completes automatically when stock replenished.   
\*\*C. Payment Capture Failure:\*\*   
&nbsp;&nbsp;1. If payment gateway capture fails, system retries twice.   
&nbsp;&nbsp;2. After failures, reverts order to "Payment Failed" status.   
&nbsp;&nbsp;3. Notifies customer to update payment method (UC-35).   
\*\*D. Customer Cancellation Request:\*\*   
&nbsp;&nbsp;1. If cancellation request received during confirmation window (<5 minutes), system aborts confirmation.   
&nbsp;&nbsp;2. Processes cancellation (UC-28) instead.   
  
---  
\*Note: Aligns with Order/Payment Information/Product entities. Transactional integrity ensures inventory-payment synchronization. Integrates with Checkout (UC-40) and Fulfillment workflows.\*  
  
### Use Case Description   
\*\*Use Case Name:\*\* Receive Order Confirmation Email   
\*\*Use Case ID:\*\* UC-42   
\*\*Actors:\*\*   
- Customer   
  
\*\*Preconditions:\*\*   
1. System is operational.   
2. Order has been successfully confirmed (UC-41 completed).   
3. Customer's email address is valid and active.   
4. Email delivery service is available.   
  
\*\*Postconditions:\*\*   
1. Order confirmation email is delivered to customer's registered email address.   
2. Email delivery status is recorded in order history.   
3. Customer has access to order details via email.   
  
\*\*Main Flow:\*\*   
1. System composes confirmation email with:   
 - Order ID, date, and summary   
 - Itemized product list with quantities/prices   
 - Shipping address and tracking information (if available)   
 - Payment method (masked) and total amount   
 - Secure link to view order details   
2. System sends email via configured SMTP service.   
3. System updates order history with "Confirmation email sent" status and timestamp.   
4. Customer receives email in their inbox.   
5. Customer views email contents to verify order accuracy.   
  
\*\*Alternative Flow:\*\*   
\*\*A. Email Delivery Failure:\*\*   
&nbsp;&nbsp;1. SMTP service returns bounce/failure notification.   
&nbsp;&nbsp;2. System retries delivery after 5 minutes.   
&nbsp;&nbsp;3. After 2 failures, system flags order with "Email Delivery Failed".   
&nbsp;&nbsp;4. Customer receives in-app notification prompting email verification.   
\*\*B. Customer Requests Resend:\*\*   
&nbsp;&nbsp;1. Customer selects "Resend Confirmation" in order history (UC-25).   
&nbsp;&nbsp;2. System re-sends identical email with "[Resent]" tag in subject line.   
&nbsp;&nbsp;3. New delivery attempt logged in order history.   
\*\*C. Spam Filter Interception:\*\*   
&nbsp;&nbsp;1. Customer reports missing email after 15 minutes.   
&nbsp;&nbsp;2. System provides manual "View Confirmation" option in order history.   
&nbsp;&nbsp;3. Customer marks sender as "Not Spam" in email client.   
  
---  
\*Note: Aligns with Order/Payment Information entities. Secure link integrates with View Order Details (UC-26). Delivery monitoring ensures customer notification compliance.\*