Software Requirements Specification (SRS)
for
E-commerce Platform

Menna Abdelmouty Noseer 120210164
Abdelrahman Mohamed Galhom 120210209

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for an e-commerce platform. The system is designed to facilitate online shopping experiences, manage customer relationships, process orders and payments, and handle product inventory.

1.2 Scope

The E-commerce Platform will provide comprehensive functionality for online retail operations, including user management, product catalog browsing, shopping cart operations, order processing, payment handling, and shipment tracking. The system will serve multiple stakeholders including customers, administrators, and support staff.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS** - Software Requirements Specification
- **UI** - User Interface
- **API** - Application Programming Interface
- **CRUD** - Create, Read, Update, Delete
- **EERD** - Enhanced Entity-Relationship Diagram

1.4 References

1. IEEE Standard 830-1998, IEEE Recommended Practice for Software Requirements Specifications
2. E-commerce Platform Database Schema (EERD)

1.5 Overview

The remainder of this document is organized as follows: Section 2 provides an overall description of the system including product perspective, functions, user characteristics, constraints, and assumptions. Section 3 details specific requirements including functional requirements, non-functional requirements, and system interfaces. Section 4 provides supporting information including use cases and class diagrams.

2 Overall Description

2.1 Product Perspective

The E-commerce Platform is a web-based application that will allow customers to browse products, make purchases, and track orders. The system will interface with payment processing services, and shipping service providers. The platform will be accessible through web browsers.

2.2 Product Functions

The primary functions of the E-commerce Platform include:

- User registration and authentication
- User profile management
- Product catalog browsing and searching
- Shopping cart management
- Order placement and tracking
- Multiple payment method processing
- Shipment tracking
- Wishlist management
- Product categorization
- Administrative functions for inventory and user management

2.3 User Characteristics

The system will serve several types of users:

- **Customers**: End-users who browse products, make purchases, and track orders
- **Administrators**: Staff who manage products, categories, orders, and user accounts

2.4 Assumptions and Dependencies

- Users have access to internet-connected devices
- Third-party payment processing services are available and operational
- Shipping carriers provide tracking APIs for integration
- The system will have continuous internet connectivity

3 Specific Requirements

3.1 Stakeholders

Stakeholder	Role	Interests
Customers	End users who purchase products	Easy navigation, secure payments, order tracking
Business Owners	Platform owners	Revenue generation, customer satisfaction, market expansion
System Administrators	IT staff managing the platform	System stability, security, maintainability
Payment Processors	Third-party service providers	Transaction security, API integration

Table 1: Stakeholder Analysis

3.2 Functional Requirements

3.2.1 User Management Requirements

FR-UM1 The system shall allow users to register with email, first name, last name, and password

FR-UM2 The system shall allow users to log in using email and password

FR-UM3 The system shall allow users to update their profile information

FR-UM4 The system shall maintain separate user roles for customers and administrators

FR-UM5 The system shall allow users to view their order history

3.2.2 Product Catalog Requirements

FR-PC1 The system shall display products with images, descriptions, prices, and availability

FR-PC2 The system shall organize products into hierarchical categories

FR-PC3 The system shall allow users to search for products by keywords

FR-PC4 The system shall support filtering products by various attributes

FR-PC5 The system shall display product reviews and ratings

FR-PC6 The system shall allow administrators to add, update, and remove products

3.2.3 Shopping Cart Requirements

FR-SC1 The system shall allow users to add products to a shopping cart

FR-SC2 The system shall allow users to modify quantities in the cart

FR-SC3 The system shall allow users to remove items from the cart

FR-SC4 The system shall save shopping cart contents for registered users

FR-SC5 The system shall calculate subtotals, taxes, and shipping costs

3.2.4 Order Management Requirements

FR-OM1 The system shall generate unique order numbers for each purchase

FR-OM2 The system shall collect shipping and billing addresses during checkout

FR-OM3 The system shall process payments through multiple payment methods

FR-OM4 The system shall allow users to track order status

FR-OM5 The system shall generate invoices for completed orders

3.2.5 Payment Processing Requirements

FR-PP1 The system shall support credit card payments

FR-PP2 The system shall support PayPal integration

FR-PP3 The system shall support bank transfer payments

FR-PP4 The system shall generate payment receipts

3.2.6 Shipping and Delivery Requirements

FR-SD1 The system shall calculate shipping costs based on address and items

FR-SD2 The system shall provide tracking numbers to customers

3.2.7 Wishlist Requirements

FR-WL1 The system shall allow users to add products to wishlists

FR-WL2 The system shall allow users to move items from wishlist to cart

3.2.8 Administrative Requirements

FR-AD1 The system shall provide an administrative dashboard

FR-AD2 The system shall allow administrators to manage user accounts

FR-AD3 The system shall allow administrators to manage product inventory

3.3 Non-functional Requirements

3.3.1 Performance Requirements

NFR-P1 The system shall load product pages within 2 seconds under normal load

NFR-P2 The system shall support at least 1000 concurrent users

NFR-P3 The system shall process checkout operations within 10 seconds

NFR-P4 The system shall be available 99.9% of the time

NFR-P5 The system shall support at least 100,000 product listings

NFR-P6 Database queries shall complete within 500ms

3.3.2 Security Requirements

NFR-S1 The system shall use HTTPS for all communications

NFR-S2 The system shall implement protection against SQL injection

NFR-S3 The system shall enforce password complexity requirements

NFR-S4 Payment information shall not be stored in the system database

3.3.3 Usability Requirements

NFR-U1 The system shall provide clear error messages for user actions

NFR-U2 The checkout process shall be completable in 5 steps or fewer

NFR-U3 The system shall maintain consistent navigation across all pages

3.3.4 Maintainability Requirements

NFR-M1 The system shall follow a modular architecture

NFR-M2 The system code shall be thoroughly documented

4 Use Cases

4.1 Actor Descriptions

1. **Customer**: A registered or unregistered user who browses and purchases products
2. **Administrator**: A user with elevated privileges who manages the system
3. **System**: Automated processes within the e-commerce platform
4. **Payment Gateway**: External service that processes payments
5. **Shipping Service**: External service that handles product delivery

4.2 Use Case Diagrams

4.3 Detailed Use Cases

4.3.1 UC-1: Register Account

- **Actors**: Customer, System
- **Description**: Customer creates a new account in the system
- **Preconditions**: Customer is not logged in
- **Main Flow**:
    1. Customer navigates to registration page

2. Customer enters personal information (name, email, password)
3. System validates input data
4. System creates new account
5. System activates account

- **Alternative Flows**:
  1. Email already registered
- **Postconditions**: Customer account is created and active

4.3.2 UC-2: Place Order

- **Actors**: Customer, System, Payment Gateway, Shipping Service
- **Description**: Customer completes a purchase transaction
- **Preconditions**: Customer has products in shopping cart
- **Main Flow**:
  1. Customer proceeds to checkout
  2. Customer provides/confirms shipping address
  3. Customer selects shipping method
  4. Customer selects payment method
  5. Customer enters payment details
  6. System sends payment request to Payment Gateway
  7. Payment Gateway processes payment
  8. System creates order record
  9. System sends order confirmation
- **Alternative Flows**:
  1. Payment processing fails
  2. Item out of stock
- **Postconditions**: Order is placed, inventory is updated, payment is processed

4.3.3 UC-3: Track Order

- **Actors**: Customer, System, Shipping Service
- **Description**: Customer checks the status of an order
- **Preconditions**: Customer has placed an order
- **Main Flow**:
  1. Customer logs into account
  2. Customer navigates to order history
  3. Customer selects specific order
  4. System retrieves order details
  5. System queries Shipping Service for tracking information
  6. System displays order status and tracking details
- **Alternative Flows**:
  1. Customer uses order tracking number without login
  2. Shipping information not yet available
- **Postconditions**: Customer views current order status

5 Entity-Relationship Diagram (ERD)

This section presents the Entity Relationship Diagram (ERD) for the e-commerce system, detailing the entities, their attributes, and the relationships between them.

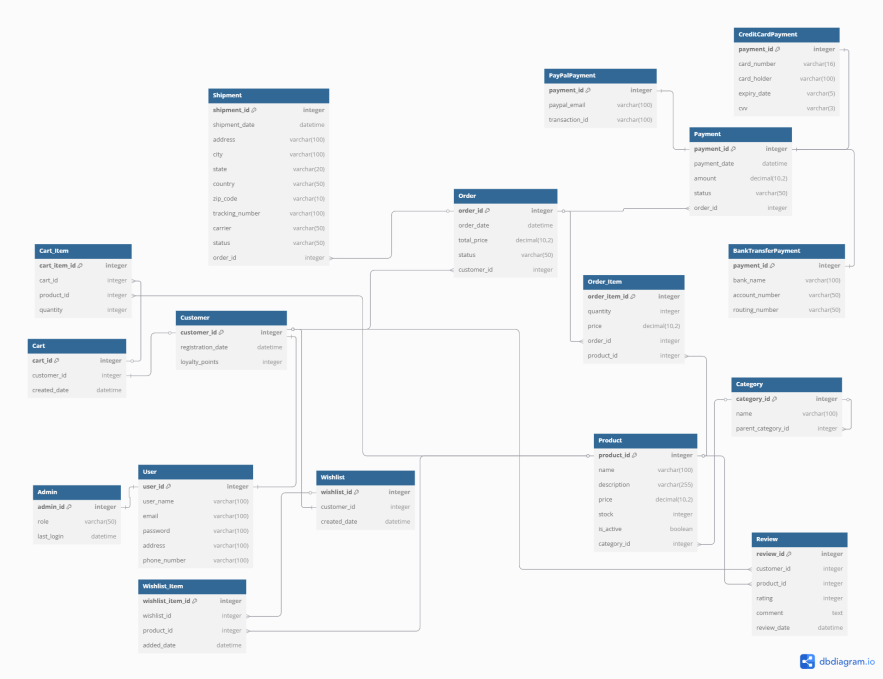

Figure 1: ERD for e-commerce

Note: Due to the simplicity of the data modeling, an EERD and an ERD result in the same diagram.

5.1 Overview of Entities

The system comprises the following main entities:

- **User**: Base entity for system users
- **Customer**: Specialized user who can place orders
- **Admin**: Specialized user who can manage the system
- **Order**: Represents customer purchases
- **Payment**: Base entity for transaction payments
- **Product**: Items available for purchase
- **Category**: Classification of products
- **Cart**: Temporary storage of items before purchase

- **Wishlist**: Customer's saved items for future consideration
- **Shipment**: Delivery information for orders
- **Review**: Customer feedback on products

5.2 Entity Relationships and Cardinalities

5.2.1 User Inheritance Hierarchy

- **User-Customer**: One-to-one relationship. A Customer is a specialized form of User.
  - Cardinality: (1:1) One User record can be associated with at most one Customer record.
  - Implementation: The `customer_id` in the Customer table references `user_id` in the User table.
- **User-Admin**: One-to-one relationship. An Admin is a specialized form of User.
  - Cardinality: (1:1) One User record can be associated with at most one Admin record.
  - Implementation: The `admin_id` in the Admin table references `user_id` in the User table.

5.2.2 Customer Relationships

- **Customer-Order**: One-to-many relationship.
  - Cardinality: (1:N) A Customer can place multiple Orders, but each Order belongs to exactly one Customer.
  - Implementation: The `customer_id` in the Order table references `customer_id` in the Customer table.
- **Customer-Cart**: One-to-one relationship.
  - Cardinality: (1:1) Each Customer has exactly one Cart, and each Cart belongs to exactly one Customer.
  - Implementation: The `customer_id` in the Cart table references `customer_id` in the Customer table.
- **Customer-Wishlist**: One-to-one relationship.

  - Cardinality: (1:1) Each Customer has exactly one Wishlist, and each Wishlist belongs to exactly one Customer.
  - Implementation: The `customer_id` in the Wishlist table references `customer_id` in the Customer table.

- **Customer-Review**: One-to-many relationship.
  - Cardinality: (1:N) A Customer can write multiple Reviews, but each Review is written by exactly one Customer.
  - Implementation: The `customer_id` in the Review table references `customer_id` in the Customer table.

5.2.3 Order Relationships

- **Order-Order_Item**: One-to-many relationship.
  - Cardinality: (1:N) An Order can contain multiple Order Items, but each Order Item belongs to exactly one Order.
  - Implementation: The `order_id` in the Order_Item table references `order_id` in the Order table.

- **Order-Payment**: One-to-many relationship.
  - Cardinality: (1:N) An Order can have multiple Payments (e.g., partial payments), but each Payment is associated with exactly one Order.
  - Implementation: The `order_id` in the Payment table references `order_id` in the Order table.

- **Order-Shipment**: One-to-many relationship.
  - Cardinality: (1:N) An Order can have multiple Shipments (e.g., partial shipments), but each Shipment is associated with exactly one Order.
  - Implementation: The `order_id` in the Shipment table references `order_id` in the Order table.

5.2.4 Payment Inheritance Hierarchy

- **Payment-CreditCardPayment**: One-to-one relationship.
  - Cardinality: (1:1) A CreditCardPayment is a specialized form of Payment.

- Implementation: The `payment_id` in the CreditCardPayment table references `payment_id` in the Payment table.

- **Payment-PayPalPayment**: One-to-one relationship.
  - Cardinality: (1:1) A PayPalPayment is a specialized form of Payment.
  - Implementation: The `payment_id` in the PayPalPayment table references `payment_id` in the Payment table.

- **Payment-BankTransferPayment**: One-to-one relationship.
  - Cardinality: (1:1) A BankTransferPayment is a specialized form of Payment.
  - Implementation: The `payment_id` in the BankTransferPayment table references `payment_id` in the Payment table.

5.2.5 Product Relationships

- **Product-Category**: Many-to-one relationship.
  - Cardinality: (N:1) Multiple Products can belong to one Category, but each Product belongs to exactly one Category.
  - Implementation: The `category_id` in the Product table references `category_id` in the Category table.

- **Product-Order_Item**: One-to-many relationship.
  - Cardinality: (1:N) A Product can appear in multiple Order Items, but each Order Item contains exactly one Product.
  - Implementation: The `product_id` in the Order_Item table references `product_id` in the Product table.

- **Product-Cart_Item**: One-to-many relationship.
  - Cardinality: (1:N) A Product can appear in multiple Cart Items, but each Cart Item contains exactly one Product.
  - Implementation: The `product_id` in the Cart_Item table references `product_id` in the Product table.

- **Product-Wishlist_Item**: One-to-many relationship.

- Cardinality: (1:N) A Product can appear in multiple Wishlist Items, but each Wishlist Item contains exactly one Product.
- Implementation: The `product_id` in the Wishlist_Item table references `product_id` in the Product table.

- **Product-Review**: One-to-many relationship.
  - Cardinality: (1:N) A Product can have multiple Reviews, but each Review is for exactly one Product.
  - Implementation: The `product_id` in the Review table references `product_id` in the Product table.

5.2.6 Category Relationships

- **Category-Category**: Self-referencing one-to-many relationship.
  - Cardinality: (1:N) A Category can have multiple child Categories, but each Category has at most one parent Category.
  - Implementation: The `parent_category_id` in the Category table references `category_id` in the same Category table.

5.2.7 Cart and Wishlist Relationships

- **Cart-Cart_Item**: One-to-many relationship.
  - Cardinality: (1:N) A Cart can contain multiple Cart Items, but each Cart Item belongs to exactly one Cart.
  - Implementation: The `cart_id` in the Cart_Item table references `cart_id` in the Cart table.

- **Wishlist-Wishlist_Item**: One-to-many relationship.
  - Cardinality: (1:N) A Wishlist can contain multiple Wishlist Items, but each Wishlist Item belongs to exactly one Wishlist.
  - Implementation: The `wishlist_id` in the Wishlist_Item table references `wishlist_id` in the Wishlist table.

5.3 Key Design Considerations

5.3.1 Inheritance Implementation

The system uses table-per-type inheritance pattern for both User and Payment hierarchies. This provides clear separation between base and specialized entities while maintaining referential integrity.

5.3.2 Separation of Concerns

Order-related entities (Order, Order_Item, Payment, Shipment) are separated to allow for flexible order processing, including partial shipments and multiple payment methods per order.

5.3.3 Customer Engagement

The system supports customer engagement through Reviews, Wishlist, and Cart functionality, enabling a comprehensive e-commerce experience.

5.3.4 Product Organization

Products are organized in a hierarchical category structure, allowing for flexible classification and easy navigation.

6 Class Diagram

The class diagram represents the object-oriented structure of the E-commerce Platform.

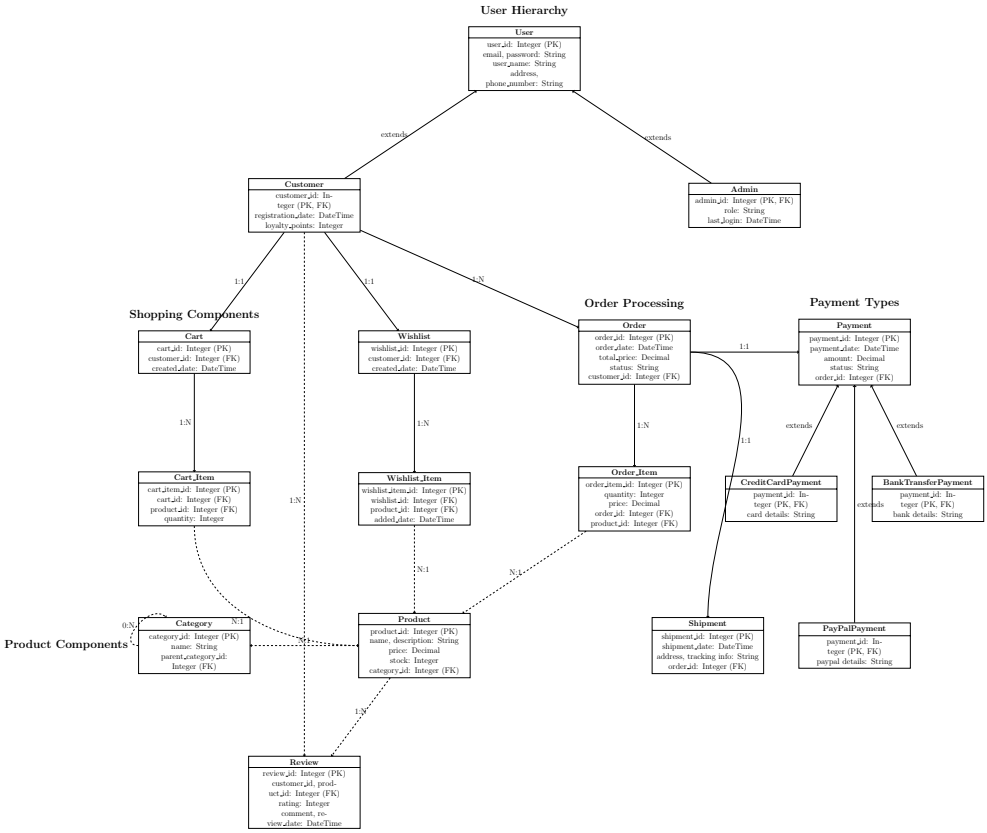

Figure 2: Simplified E-commerce Platform Class Diagram