

Unified Business Rules Document (SRS Ready)

Magento Open Source – Platform Independent

## 1. Product & Catalog Domain

- Product is a sellable entity with unique SKU.
- Product must belong to at least one category.
- Product can be enabled, disabled, or hidden.

## 2. Customer Domain

- Customer can be registered or guest.
- Registered customer has persistent profile and order history.
- Customer belongs to a group affecting pricing and promotions.

## 3. Cart / Checkout Domain

- Cart represents purchase intent, not a legal transaction.
- Prices in cart are temporary.
- Cart converts to order on checkout confirmation.

## 4. Sales / Orders Domain

- Order is a legal immutable sales record.
- Supports partial invoices, shipments, and refunds.

## 5. Payments Domain

- Order is paid only after confirmed payment.
- Payment must be idempotent.

## 6. Shipping / Fulfillment Domain

- Shipment created only for confirmed orders.
- Supports partial deliveries.

## 7. Promotions Domain

- Discounts apply based on conditions.
- Cannot result in negative price.

## 8. Reviews & Ratings Domain

- Reviews build trust and may require moderation.

## 9. Notifications Domain

- Notifications are event-driven.
- No duplicate messages allowed.

## 10. Settings / Wallet / Packages

- Each store has isolated configuration.
- Wallet is ledger-based.

## 11. Appearance & App Store

- Themes affect presentation only.
- Plugins must be isolated and permission-based.

### Conclusion:

This document represents the complete business logic foundation for rebuilding Magento into a clean, scalable Django platform.

# Software Requirements Specification (SRS)

Project: Magento Open Source → Django Platform (AI-Assisted Rebuild)

Document Type: Functional & Non-Functional Requirements

Purpose: Define a complete, platform-independent specification for rebuilding the system.

# **1. Introduction**

This document defines the functional and non-functional requirements for rebuilding an e-commerce platform originally based on Magento Open Source into a clean, scalable Django-based system.

The document is business-driven and independent of any specific implementation technology.

## **2. Scope**

The system covers product management, customers, orders, payments, shipping, marketing, notifications, configuration, appearance, and extensibility via plugins.

The initial delivery focuses on an MVP, with phased expansion.

## **3. Definitions & Acronyms**

MVP: Minimum Viable Product

SKU: Stock Keeping Unit

SRS: Software Requirements Specification

## **4. Functional Requirements**

### **4.1 Product & Catalog Management**

The system shall allow creation, update, activation, and deactivation of products.

The system shall enforce unique SKU for each product.

The system shall support product categorization and visibility control.

### **4.2 Customer Management**

The system shall support registered and guest customers.

The system shall maintain customer profiles, addresses, and order history.

The system shall support customer grouping affecting pricing and promotions.

### **4.3 Cart & Checkout**

The system shall allow users to add, update, and remove items from a cart.

The system shall calculate totals dynamically and convert a cart into an order upon confirmation.

### **4.4 Orders & Sales**

The system shall create immutable order records.

The system shall support partial invoicing, shipping, and refunds.

### **4.5 Payments**

The system shall support multiple payment methods.

The system shall confirm payments asynchronously and prevent duplicate charges.

#### **4.6 Shipping & Fulfillment**

The system shall calculate shipping costs based on address and method.

The system shall manage shipment lifecycle and tracking.

#### **4.7 Promotions & Marketing**

The system shall apply discounts based on configurable rules and conditions.

The system shall prevent invalid or conflicting promotions.

#### **4.8 Reviews & Ratings**

The system shall allow customers to submit reviews and ratings.

The system shall support moderation workflows.

#### **4.9 Notifications & Messaging**

The system shall send notifications triggered by business events.

The system shall support multiple communication channels.

#### **4.10 Settings, Packages & Wallet**

The system shall support store-specific configuration.

The system shall manage subscription packages and wallet balances.

#### **4.11 Appearance & App Store**

The system shall allow theme selection without affecting business logic.

The system shall support plugin installation with permission control.

### **5. Non-Functional Requirements**

#### **5.1 Performance**

The system shall respond to API requests within acceptable latency under normal load.

#### **5.2 Scalability**

The system shall support horizontal scaling for increased traffic.

#### **5.3 Security**

The system shall enforce authentication, authorization, and data protection.

Sensitive financial data shall not be stored insecurely.

## **5.4 Reliability**

The system shall ensure consistency of financial and order data.

Critical operations shall be idempotent.

## **5.5 Maintainability**

The system shall follow clean architecture principles.

Business logic shall be isolated from infrastructure concerns.

## **5.6 Audit & Compliance**

The system shall maintain audit logs for financial and critical operations.

# **6. Assumptions & Constraints**

Initial release targets MVP scope.

Advanced enterprise features are planned for later phases.

# **7. Conclusion**

This SRS document serves as the authoritative reference for designing, implementing, and validating the Django-based platform replacing Magento Open Source.

# Software Requirements Specification (SRS)

Section: Actors & Use Cases

Project: Magento Open Source → Django Platform

Purpose: Define system actors and their interactions with the system.

# **1. System Actors**

## **1.1 Store Owner**

Manages store configuration, products, orders, and subscriptions.

## **1.2 Admin**

System-level administrator responsible for platform configuration and monitoring.

## **1.3 Customer (Registered)**

End user with an account who can browse, purchase, and review products.

## **1.4 Customer (Guest)**

End user without an account who can browse and place orders.

## **1.5 Payment Gateway**

External system responsible for processing payments.

## **1.6 Shipping Provider**

External system responsible for shipping and delivery tracking.

## 2. Core Use Cases

### UC-01: Manage Products

Actor: Store Owner

Description: Create, update, activate, or deactivate products.

Outcome: Products become available or unavailable for sale.

### UC-02: Browse Products

Actor: Customer

Description: Browse and search products by category or keyword.

Outcome: Product listings are displayed.

### UC-03: Place Order

Actor: Customer (Registered / Guest)

Description: Add items to cart and complete checkout.

Outcome: Order is created.

### UC-04: Process Payment

Actor: Payment Gateway

Description: Confirm or reject payment for an order.

Outcome: Order payment status updated.

### UC-05: Ship Order

Actor: Store Owner / Shipping Provider

Description: Prepare and ship order items.

Outcome: Shipment created and tracked.

### UC-06: Submit Review

Actor: Customer

Description: Submit rating and review for a purchased product.

Outcome: Review pending or published.

### UC-07: Send Notification

Actor: System

Description: Notify users about order, payment, or shipment events.

Outcome: Message delivered through configured channel.

### UC-08: Install Plugin

Actor: Store Owner

Description: Install or remove platform plugins.

Outcome: Feature enabled or disabled.

### 3. Conclusion

This section complements the SRS by defining actors and use cases, enabling clear understanding of system behavior and responsibilities.

# SRS – Use Case Diagrams (Textual Representation)

Project: Magento Open Source → Django Platform

Purpose: Provide clear use case diagrams in a textual UML-style format suitable for documentation and reviews.

## Diagram 1: Customer Purchase Flow

```
Customer | |--> Browse Products |--> View Product Details |--> Add to Cart |-->
Checkout |--> Select Address |--> Select Shipping |--> Select Payment |-->
Confirm Order |--> Receive Notification
```

## Diagram 2: Store Owner Management Flow

```
Store Owner | |--> Manage Products |--> Manage Categories |--> Manage Orders |-->
Approve Order |--> Create Shipment |--> Issue Refund |--> View Reports |-->
Manage Store Settings
```

### Diagram 3: Payment Processing Flow

```
Customer --> Checkout | v Payment Gateway | [Success | Failure] | v Update Order  
Status | v Send Notification
```

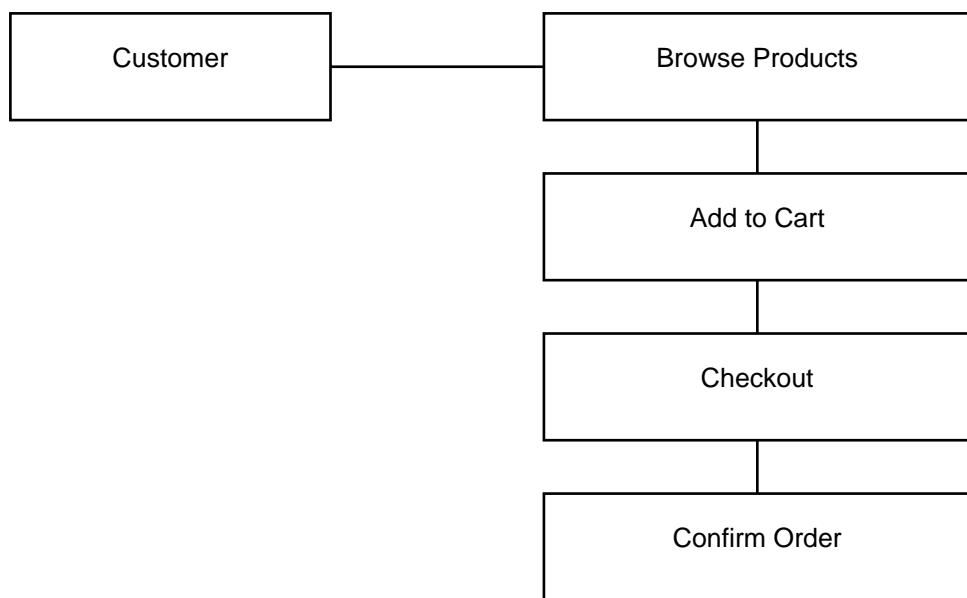
## Diagram 4: Plugin Installation Flow

```
Store Owner | |--> Browse App Store |--> Select Plugin |--> Review Permissions  
|--> Install Plugin |--> Enable / Disable Plugin
```

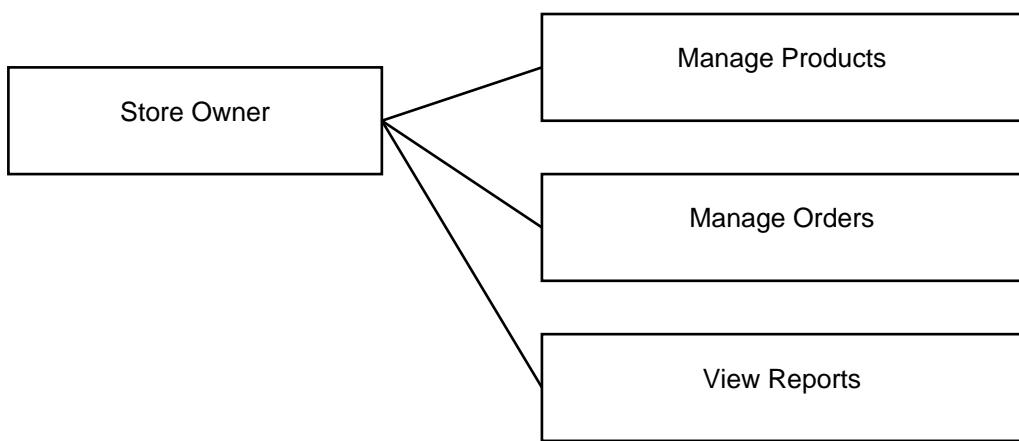
## Conclusion

These textual use case diagrams complement the SRS by visually explaining system interactions in a simple, implementation-independent format.

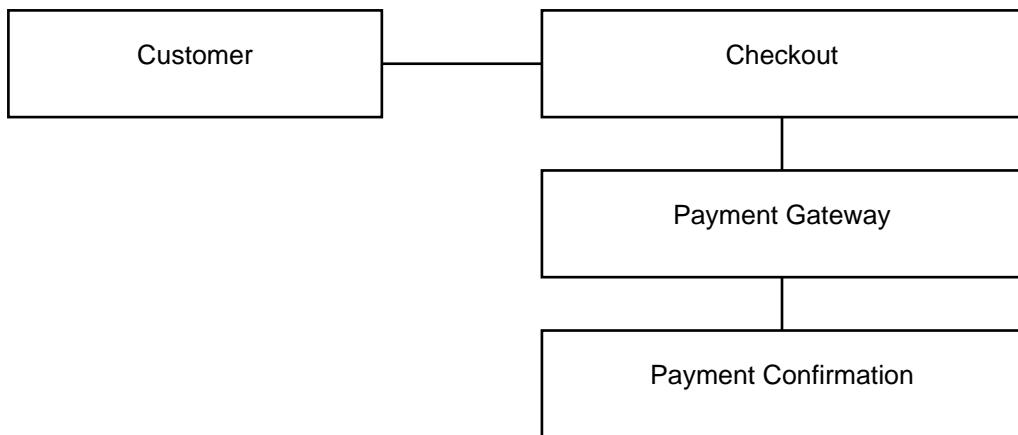
## Use Case Diagram – Customer Purchase Flow



## Use Case Diagram – Store Owner Management



## Use Case Diagram – Payment Processing



# SRS – Sequence Diagrams

## 1. Order Placement Sequence

```
Customer -> Cart : Add items
Customer -> Checkout : Start checkout
Checkout -> Pricing : Calculate totals
Checkout -> Payment : Initiate payment
Payment -> Gateway : Process payment
Gateway -> Payment : Payment confirmation
Payment -> Order : Mark order as paid
Order -> Notification : Send confirmation
```

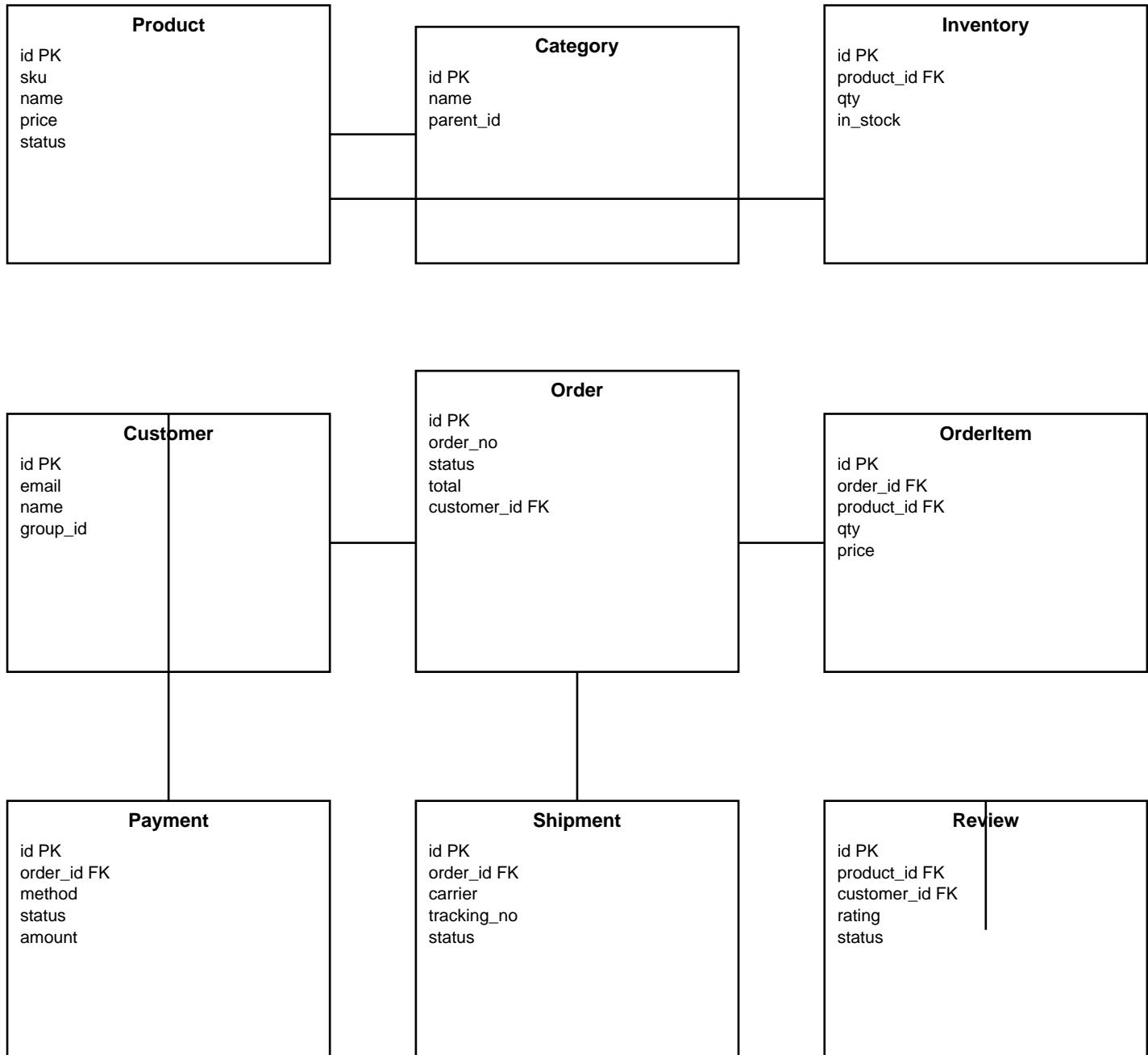
## 2. Payment Failure Sequence

```
Customer -> Checkout : Confirm order
Checkout -> Payment : Initiate payment
Payment -> Gateway : Process payment
Gateway -> Payment : Failure response
Payment -> Order : Keep order pending
Order -> Notification : Send failure message
```

### 3. Shipping Sequence

```
Store Owner -> Order : Prepare shipment  
Order -> Shipping : Create shipment  
Shipping -> Carrier : Send shipment data  
Carrier -> Shipping : Tracking number  
Shipping -> Notification : Send tracking info
```

## ER Diagram – Full Platform (Magento → Django)



## ER Diagram – Wallet, Subscriptions & Plugins

