Software Requirements Specification (SRS)

Smart Shop System

E-Commerce Web Application

Version: 1.0
Date: December 11, 2025
Author: Development Team
Status: Approved
Classification: Internal Use

This document specifies the requirements for the Smart Shop System, a web-based e-commerce platform.

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the requirements for the Smart Shop System, a web-based e-commerce platform that enables online shopping with advanced features including product management, shopping cart functionality, order processing, and promotional offers.

1.2 Scope

The Smart Shop System is designed to:
• Provide a user-friendly web interface for customers to browse and purchase products
• Enable administrators to manage inventory, products, and promotional offers
• Support multiple user roles (Admin and Customer)
• Handle order processing and payment methods
• Implement promotional offers with discounts, gifts, and purchase limits

1.3 Definitions, Acronyms, and Abbreviations

SRS: Software Requirements Specification
UI: User Interface
API: Application Programming Interface
Admin: Administrator user role
Customer: Regular user role
COD: Cash on Delivery
POS: Point of Sale (Visa on Delivery)
MVC: Model-View-Controller
WSGI: Web Server Gateway Interface
RTL: Right-to-Left (text direction)

1.4 References

• Flask Framework Documentation: https://flask.palletsprojects.com/
• Python 3.11 Documentation: https://docs.python.org/3.11/
• Railway Deployment Guide: https://docs.railway.app/

2. Overall Description

2.1 Product Perspective

The Smart Shop System is a standalone web application built using:
• **Backend:** Python 3.11 with Flask framework
• **Frontend:** HTML, CSS, JavaScript
• **Deployment:** Railway cloud platform
• **Architecture:** Model-View-Controller (MVC) pattern

2.2 Product Functions

The system provides the following main functions:
1. User authentication and authorization
2. Product catalog management
3. Shopping cart functionality
4. Order processing
5. Promotional offer management
6. Inventory management
7. Sales reporting

2.3 User Classes and Characteristics

2.3.1 Administrator

• **Responsibilities:** Manage products, inventory, offers, and view sales
• **Access Level:** Full system access
• **Default Credentials:** admin/123 or place/123

2.3.2 Customer

• **Responsibilities:** Browse products, add to cart, place orders
• **Access Level:** Limited to shopping features
• **Registration:** Can create new account

3. System Features

3.1 User Authentication

3.1.1 Login

Priority: High
Description: Users can log in with username and password
Input: Username, Password
Output: Redirect to appropriate dashboard
Validation:
• Username and password must match existing account
• Case-sensitive username

3.1.2 Registration

Priority: High
Description: New customers can create accounts
Input: Username, Password
Output: New user account created, auto-login
Validation:
• Username must be unique
• Username cannot be empty
• Password cannot be empty

3.2 Product Management (Admin)

3.2.1 View Products

Priority: High
Description: Admin can view all products with details
Display: Table format with ID, Name, Price, Stock, Offers

3.2.2 Edit Product

Priority: High
Description: Admin can modify product information
Editable Fields: Name, Price, Stock
Validation:
• Price must be positive number
• Stock must be non-negative integer

3.3 Shopping Cart

3.3.1 Add to Cart

Priority: High
Description: Customers can add products to shopping cart
Input: Product ID, Quantity
Validation:
• Quantity must be positive

• Quantity cannot exceed available stock
• If offer limit exists, total quantity in cart cannot exceed limit
Output: Product added to cart, cart count updated

4. Data Model

4.1 Product Entity

```
Product { id: Integer (Primary Key) name: String price: Float stock: Integer
category: String offer_discount: Float (0-100) offer_gift: String (nullable)
offer_limit: Integer (0 = unlimited) }
```

4.2 User Entity

```
User { username: String (Primary Key) password: String role: String ("Admin" |
"Customer") cart: Array } CartItem { product: Product qty: Integer }
```

4.3 Order Entity

```
Order { order_id: Integer (Primary Key) customer_name: String items_txt: Array
total: Float address: String pay_method: String pay_status: String date: DateTime }
```

5. Security Requirements

5.1 Authentication

• **Method:** Username/password
• **Session:** Flask session management
• **Timeout:** Session expires on browser close
• **Password Policy:** None (for MVP)

5.2 Authorization

• **Role-based:** Admin vs Customer
• **Access Control:** Routes protected by role check
• **Default Admin:** admin/123, place/123

6. Performance Requirements

6.1 Response Times

• **Page Load:** < 2 seconds
• **API Response:** < 500ms
• **Cart Update:** < 300ms
• **Checkout:** < 1 second

6.2 Throughput

• **Concurrent Users:** 50+ users
• **Requests per Second:** 100+ requests
• **Database Operations:** N/A (in-memory)

7. Deployment Requirements

7.1 Deployment Platform

• **Platform:** Railway
• **URL:** https://railway.app
• **Build:** Automatic from GitHub
• **Runtime:** Python 3.11

7.2 Dependencies

Python: 3.11+
Packages:
• Flask==3.0.0
• Werkzeug==3.0.1
• gunicorn==21.2.0

8. Appendices

8.1 Glossary

MVP: Minimum Viable Product
RTL: Right-to-Left (text direction)
WSGI: Web Server Gateway Interface
COD: Cash on Delivery
POS: Point of Sale

8.2 Future Enhancements

1. Database integration (PostgreSQL)
2. User password hashing
3. Email notifications
4. Payment gateway integration
5. Product images
6. Search functionality
7. Product reviews
8. Wishlist feature
9. Order tracking
10. Admin analytics dashboard