Retail Shop Management System - SRS

Revision History

Name	Date	Reason For Changes	Version
Muhammad Arham	13/02/2025	Initial Draft	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the **Retail Shop Management System (RSMS)**. This system aims to streamline the operations of a retail shop, including inventory management, sales tracking, customer management, and reporting.

1.2 Document Conventions

This document follows standard formatting with:

- Bold for key terms.
- Bullet points for list.
- Tables where applicable.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers**: To understand system architecture and requirements.
- **Project Managers**: To track deliverables and system functionality.
- **End Users**: To comprehend system capabilities.
- **QA Testers**: To validate functionality against requirements.

1.4 Project Scope

The Retail Shop Management System is designed to assist shop owners in managing inventory, processing sales, generating invoices, and maintaining customer and supplier details. The system will enhance efficiency and reduce manual work by automating core business operations.

1.5 References

- IEEE 830-1998 Software Requirements Specification Standard.
- Relevant APIs and system documentation.

2. Overall Description

2.1 Product Perspective

The system will be a **web-based application** that can be accessed through any modern browser. It will integrate with barcode scanners and printers for efficient sales processing.

2.2 Product Features

- Inventory management.
- Sales and billing.
- Customer and supplier management.
- Role-based access control.
- Reports and analytics.

2.3 User Classes and Characteristics

- **Shop Owner/Admin**: Full access to the system, including product management and financial reports.
- **Cashiers**: Limited access for processing sales and managing customers.
- **Customers**: Can view available products and view product prices.

2.4 Operating Environment

- Web-based (accessible on Windows, macOS, Linux)
- Database: PostgreSQL
- Backend: Node.js
- Frontend: React

2.5 Design and Implementation Constraints

- Must use an SQL-based relational database.
- System must support real-time inventory updates.
- Data encryption required for user authentication.

2.6 User Documentation

- A **detailed user manual** will be provided.
- System documentation will be maintained on **Notion** and uploaded to **GitHub**.

2.7 Assumptions and Dependencies

- Assumes stable internet connection for cloud-based features.
- Requires integration with third-party payment gateways.

3. System Features

3.1 User Authentication and Access Control

- Users must log in with a username and password.
- Role-based access control (Admin, Cashier, Customer).

3.2 Inventory Management

- Add, update, and delete products.
- Track stock levels and generate low-stock alerts.

- Categorization of products with SKU and barcode support.

3.3 Sales and Billing

- Process sales through a Point of Sale (POS) system.
- Generate invoices for customers.
- Apply discounts and manage promotions.

3.4 Customer and Supplier Management

- Store customer details (name, contact, purchase history).
- Manage supplier information and product sourcing.

3.5 Reports and Analytics

- Generate daily, weekly, and monthly sales reports.
- Analyze customer buying patterns and inventory trends.

4. External Interface Requirements

4.1 User Interfaces

- Web-based dashboard for shop owners.
- Point of Sale (POS) system for cashiers.
- Wall mounted interface for customer interactions.

4.2 Hardware Interfaces

- Compatible with barcode scanners and receipt printers.
- Works with standard POS hardware.

4.3 Software Interfaces

- Integration with external payment gateways.

- API-based communication with inventory suppliers.

4.4 Communications Interfaces

- Secure HTTPS-based communication.
- Email/SMS notifications for order updates.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- System should support **100+ transactions per second**.
- Inventory updates should reflect within **2 seconds**.

5.2 Safety Requirements

- Data backups performed daily.
- Error logging and alerts for failures.

5.3 Security Requirements

- **User authentication with encryption**
- **Role-based access control** to prevent unauthorized access
- **Data encryption** for sensitive information like customer details and transactions

5.4 Software Quality Attributes

- **Reusability**: Built using an object-oriented programming paradigm.
- **Maintainability**: Well-documented codebase with a simple admin panel.
- **Reliability**: Ensures **99.9% uptime** with automatic backups.
- **Scalability**: Supports multiple locations and high transaction volumes.

6. Other Requirements

- Compliance with relevant retail regulations.
- Localization support for multiple languages.

7. Project Management

- **Jira** will be used for task tracking.
- Development updates and version control will be maintained in **GitHub**.