Software Requirements Specification

for

Shop Management System

Version 1.0

Prepared by

Group Number: 6

Ayaan Sajid Nalakath	**B230638CS**	**ayaan_b230638cs@nitc.ac.in**
Edwin Thomas	**B230294CS**	**edwin_b230294cs@nitc.ac.in**
Johan John Koshy	**B230049CS**	**johan_b230049cs@nitc.ac.in**
Fahad Ahmed Mahdi	**B230103CS**	**fahad_b230103cs@nitc.ac.in**

Instructor:	**Pranesh Das**
Course:	**Database Management Systems**
Date:	**28-02-2025**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Edwin Thomas & Ayaan Sajid	Completed working on sections 1 and 2. Implemented a brief sketch of database schema and design.	23/02/25
1.0	Fahad Ahmed & Johan John	Worked on sections 3 and 4. Planning of the final implementation of project, tools, and tech stack needed.	25/02/25
1.0	Fahad Ahmed & Johan John	Creation of diagram and continued with sections 3 and 4.	27/02/25
1.0	Edwin Thomas & Ayaan Sajid	Final revision and corrected all inconsistencies with the sections. Added references and diagrams. And finished the final touch up.	01/03/25

1 Introduction

The Shop Management System (SMS) is a software solution designed to streamline the operations of small to medium-sized retail shops. This SRS outlines the requirements for a comprehensive system that manages inventory, tracks sales, generates bills, and coordinates with suppliers. This section provides an overview of the document's purpose, scope, audience, key definitions, relevant references to guide the reader as well as acknowledgements.

1.1 Document Purpose

This document specifies the software requirements for the Shop Management System, version 1.0. It details the functional and non-functional requirements for a system that automates inventory management, sales processing, and billing for retail shops. This SRS covers the entire system, including its core modules for product, category, inventory, sales, billing, and supplier management, ensuring a complete solution for shop owners.

1.2 Product Scope

This document provides a comprehensive **Software Requirements Specification (SRS)** for the **Shop Management System**. It defines the system's functionalities, scope, and technical requirements necessary for efficient shop operations. The document serves as a reference for **developers, stakeholders, and testers**, ensuring a clear understanding of the system's capabilities and constraints.

The **Shop Management System** is designed to enhance the efficiency of retail shop operations by providing real-time inventory updates, accurate sales tracking, and automated billing. The system helps prevent stockouts and overstocking through automated alerts while streamlining sales processing and ensuring data integrity with a relational database design. Secure transaction recording and efficient billing processes contribute to financial accuracy, while the system's scalability allows for future enhancements, such as analytics and loyalty programs. Additionally, procurement history and sales reports support better decision-making, enabling businesses to optimize their inventory and sales strategies effectively.

1.3 Intended Audience and Document Overview

This document is intended for developers, testers, business owners, and stakeholders involved in the development and operation of the **Shop Management System**. It also serves as a reference for the client and the professor to assess the project's scope, requirements, and implementation details. Each reader will find relevant information to understand the system's objectives, functionalities, and constraints.

The **Software Requirements Specification (SRS)** is structured to provide a clear and detailed outline of the system. It begins with an **Introduction**, offering an overview of the document's purpose, audience, and scope. The **Overall Description** elaborates on the system's features, user interactions, and dependencies. The **Specific Requirements** section defines the functional and non-functional aspects necessary for implementation, while the **Non-functional Requirements** section details performance, security, and usability considerations. Finally, the **Other Requirements** section includes any additional constraints or external factors affecting the system.

For an optimal understanding of the document, readers should start with the **Introduction** to grasp the system's objectives and scope. Developers and testers should focus on the **Specific Requirements** and **Non-functional Requirements** to ensure proper implementation and testing. Business owners and stakeholders may find the **Overall Description** and **Other Requirements** more relevant for understanding system capabilities and constraints.

1.4 Definitions, Acronyms and Abbreviations

- **SRS**: Software Requirements Specification
- **SMS**: Shop Management System
- **DBMS** - Database Management System
- **POS** - Point of Sale
- **SKU** - Stock Keeping Unit
- **SMS**: Shop Management System
- **UML**: Unified Modeling Language
- **GUI**: Graphical User Interface
- **API:** Application Programming Interface
- **CSV**: Comma Separated Values
- **COMET**: Concurrent Object Modeling and Architectural Design Method
- **Inventory Transaction**: Any stock movement (e.g., purchase, sale, return)
- **Reorder Level**: Stock threshold triggering a restock alert

1.5 Document Conventions

This SRS adheres to IEEE formatting standards. Text is written in Arial font, size 11, single-spaced, with 1" margins. Section and subsection titles follow the template's bold and numbered style. Italics denote comments or placeholders. Naming conventions use camelCase for variables (e.g., stockLevel), and functional requirements are prefixed with "F" (e.g., F1).

2 Overall Description

2.1 Product Overview

The **Shop Management System** is a **centralized retail management application** designed to streamline **sales tracking, inventory management, supplier coordination, and billing processes**. It serves as an all-in-one solution for shop owners, providing real-time updates on stock levels, automating sales transactions, and ensuring seamless invoice generation. The system replaces traditional manual record-keeping methods with a **digitized, efficient, and error-free** approach to shop management.

The system is a **self-contained** product but can also integrate with external APIs for **tax calculations, payment processing, and barcode scanners**. It operates as a **standalone application** but has the potential for expansion into a **multi-store network** through cloud-based deployment. The system ensures smooth interactions between different functional modules, such as product inventory, sales processing, and reporting, making it an essential tool for businesses looking to optimize their operations.

Below is a diagram illustrating the **major components of the system**, their interactions, and external interfaces:

```
                              +----------------------------+
                              | Shop Owner/Admin |
                              +----------------------------+
                                           |
                                           V
          +-------------------------------------------------------------------------+
          |                        Shop Management System                           |
          +-------------------------------------------------------------------------+
             |                 |                          |                   |
             V                 V                          V                   V
+-------------------+  +-------------------+  +---------------------+  +--------------------+
| Inventory       |  | Sales & Billing|  | Supplier Mgmt   |  |  Report and    |
| Management  |  | Management  |  |      System        |  |    Analytics     |
+------------------+  +-------------------+  +---------------------+  +--------------------+
             |                 |                       |                 |
             V                 V                       V                 V
        +-----------------------------------------------------------------------+
        |     External Interfaces (Barcode Scanners, Payment       |
        |            Gateways, Tax APIs, Reporting Tools)                |
        +-----------------------------------------------------------------------+
```

2.2 Product Functionality

The **Shop Management System** provides essential functionalities to streamline retail operations and improve efficiency. The major functions of the system include:

- **Product Management:** Enables users to add, update, and remove products while managing pricing and stock levels.
- **Sales Management:** Records sales transactions and updates inventory in real time.
- **Billing System:** Automatically generates invoices based on sales transactions.
- **Supplier Management:** Maintains supplier details and purchase records for efficient restocking.
- **Inventory Tracking:** Monitors stock movements, including purchases, returns to ensure accurate inventory control.
- **Report and Analytics:** Generates reports of daily, weekly and monthly (according to date) sales, profit etc and provides best selling products of that time period.

2.3 Design and Implementation Constraints

The **Shop Management System** is subject to several constraints that will influence its design and development. The system must be built using the **COMET (Collaborative Object Modeling and Enterprise Transformation) method** for software design, ensuring structured object-oriented analysis and modeling. Additionally, **UML (Unified Modeling Language)** must be used for system modeling, allowing clear visualization of system components and interactions.

The system will utilize **MySQL/PostgreSQL** as the database and frontend using **Python's tkinter/PyQt** for a responsive desktop **GUI**. It must support integration with **barcode scanners** for faster product identification and implement security features such as **user authentication and role-based access control**. The system should be designed for scalability to accommodate future enhancements like analytics and loyalty programs.

References:

- Gomaa, H. (2005). **Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures**. Addison-Wesley.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). **The Unified Modeling Language User Guide**. Pearson Education.

2.4 Assumptions and Dependencies

- Assumes a stable internet for supplier communication and payment processing.
- Assumes staff are trained to use basic software interfaces.
- Depends on a third-party database engine (e.g., MySQL) for data management.
- Assumes no significant changes in payment method standards (e.g., UPI) during development.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The SMS features a desktop interface with a dashboard and pages for inventory, sales, and billing. Key elements include:

1. Login Page:
    a. A basic login page where an employee has to enter a username and password to access the shop management system.
2. Dashboard:
    a. Provides navigation to other screens like product management, sales & billing and report and analytics
3. Product Management Page:
    a. Page has options to view current stock levels and details of each product.
    b. Employees can add/edit/delete products as he pleases or start an automatic updation when new stock arrives.
    c. Page also will show if any stock is low.
4. Sales & Billing:
    a. This page is to be used when a customer is going to make a purchase.
    b. Products can be added to cart either by searching their name/product id or by simply scanning a barcode.
    c. Generates an invoice/receipt (printable).
5. Report & Analytics:
    a. Gives sales report (daily/monthly/weekly as per date) and also shows the best selling products

3.1.2 Hardware Interfaces

1. Barcode Scanner
    - Used to scan product barcodes for quick and accurate sales entry.
    - The scanner interacts with the system via a standard input interface, simulating keyboard entry.
2. Printer
    - Outputs bills, invoices, and receipts for customers.
    - Supports standard paper sizes and prints in English units (e.g., USD for currency).
3. PC (Retail Computer)
    - The primary device running the Shop Management System (SMS).
    - Handles all processing, data storage, and user interactions.

3.1.3 Software Interfaces

1. Database System
    - The SMS interacts with a local database (e.g., SQLite, MySQL, or PostgreSQL) to store and retrieve data related to products, sales, stock, suppliers, and customers.
    - SQL queries are used for data manipulation and retrieval.

2. Operating System
    - The SMS runs on a Windows or Linux-based operating system and interacts with system-level APIs for file handling, printing, and hardware communication.
3. Billing & Printing Services
    - The software sends formatted invoice and receipt data to a printer service for generating customer receipts.
    - May integrate with PDF generation tools to allow for digital invoice storage.
4. Barcode Scanner Software
    - The system processes input from the barcode scanner using standard input methods (keyboard emulation or API integration) to fetch product details quickly.
5. Backup & Export Tools
    - The software may offer data export in CSV or PDF format for record-keeping and reporting purposes.
    - Automated or manual backups can be implemented to prevent data loss.

3.2 Functional Requirements

3.2.1 F1 : The system shall maintain detailed product records

The system shall store and manage comprehensive product information in a relational database, including product name (string, max 30 characters), category (string, max 20 characters), price (decimal, INR), stock level (integer), and supplier ID (foreign key linking to supplier table). Users shall have the ability to view these records through the interface.

3.2.2 F2 : The system shall track discount for each product

The system shall store a discount percentage (0-100%) for each product in the Products table as DiscountPercentage, editable by staff, displayed in product and sales screens, and auto-applied during sales unless overridden. Discounts default to 0% and are validated to be 0-100%.

3.2.3 F3 : The system shall allow users to add, update, and delete products.

The system shall provide functionality for authorized users (shop staff) to:

- **Add**: Create new product entries with all required fields (name, category, price, stock level, supplier).
- **Update**: Modify existing product details (e.g., adjust price or stock level).
- **Delete**: Remove products from the database, with a confirmation prompt to prevent accidental deletion.

3.2.4 F4 : The system shall maintain real-time inventory updates after each sale.

Following each sale transaction, the system shall immediately deduct the sold quantities from the corresponding product stock levels and commit the update to the database. This ensures that inventory data reflects the latest state, preventing overselling or stock discrepancies.

3.2.5 F5 : The system shall generate invoices for customers.

The system shall automatically create itemized invoices based on sale transactions, including:

- SalesID (unique identifier),
- Date and time of sale,
- List of products sold (with quantities and unit prices),
- Total amount,
- Amount paid,
- Balance due (if applicable)

Invoices shall be printable and stored in the database for future reference.

3.2.6 F6 : The system shall provide sales reports and analytics.

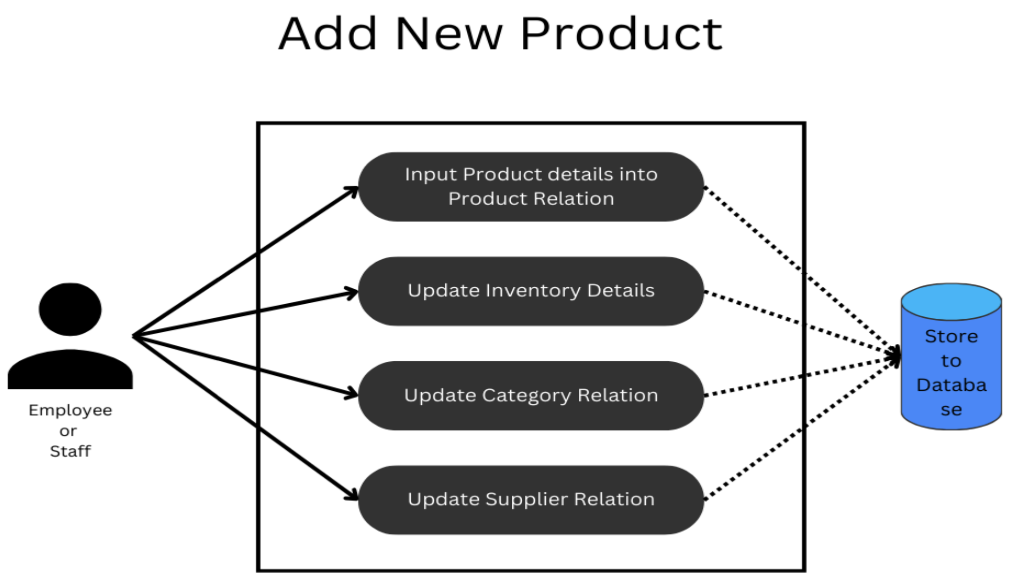
The system shall generate sales reports based on recorded transactions, offering:

- Daily, weekly, or monthly summaries of total sales (by amount and volume).
- Breakdown by product, category, etc.
- Trends analysis (e.g., best-selling items).

Reports shall be exportable in PDF or CSV format and accessible via the dashboard.

3.3 Use Case Model

3.3.1 Use Case #1 (U1: Add New Product)



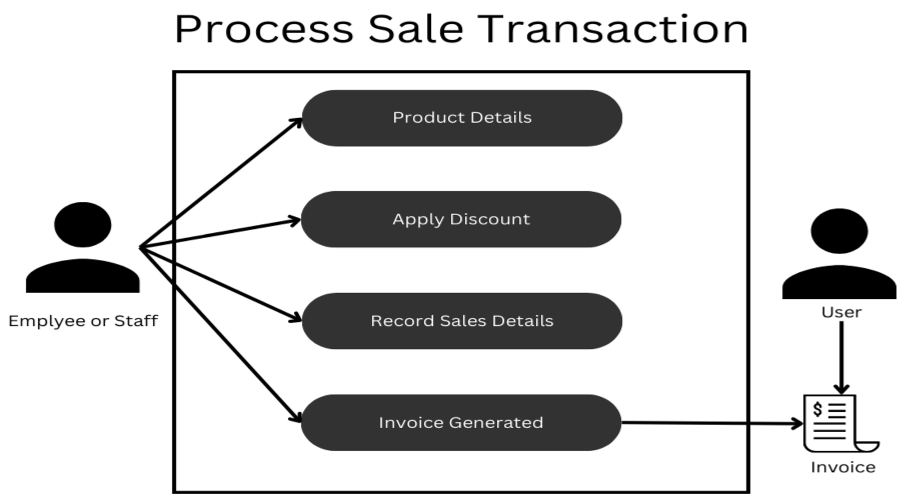
Author: Ayaan Sajid
Purpose: Allow shop staff to add a new product to the system to expand the inventory.
Requirements Traceability: F1 (maintain product records), F3 (add/update/delete products)
Priority: High – Essential for inventory management and business growth.
Preconditions: User is logged in as authorized staff; database is accessible.
Postconditions: New product record is stored in the Products table with all required fields.

Actors: Shop Staff (initiates), Database (stores data)
Extends: None
Flow of Events:

1. *Basic Flow*:
   - Staff navigates to the "Add Product" screen.
   - Staff enters product details (name, category, price, stock level, supplier ID).
   - System validates inputs and saves the record to the database.
   - System logs the action with a timestamp.
   - Staff receives confirmation of success.
2. *Alternative Flow*:
   - If a field is optional (e.g., discount), staff leaves it blank, and it defaults to 0%.
3. *Exceptions*:
   - Invalid input (e.g., negative stock) → System displays an error and prompts re-entry.
   - Database unavailable → System alerts staff and aborts the operation.

Includes: None
Notes/Issues: Ensure validation prevents duplicate product names if required by the client.

3.3.2 Use Case #2 (U2: Process Sale Transaction)

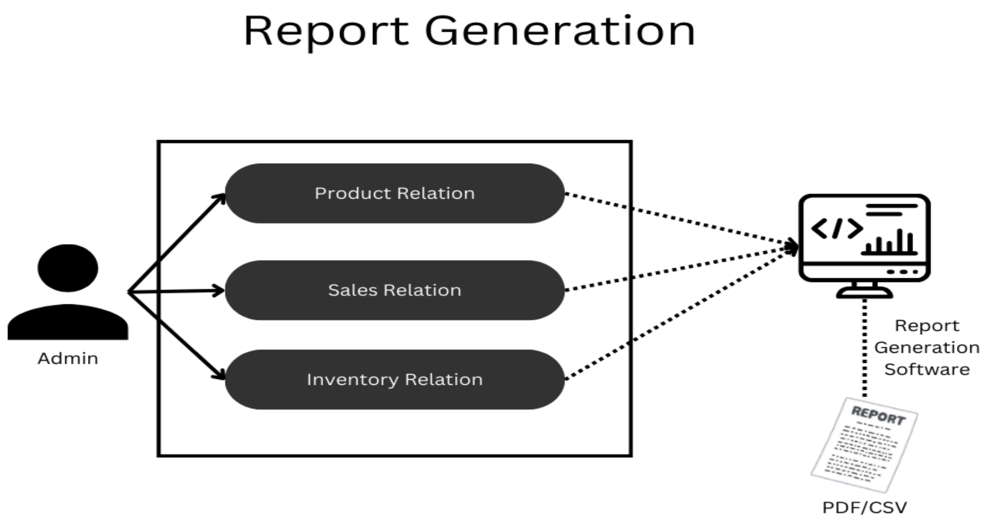


Author: Fahad Ahmed
Purpose: Record a customer sale, update inventory, and generate an invoice.
Requirements Traceability: F1 (product records), F2 (track discount), F4 (real-time inventory updates), F5 (generate invoices)
Priority: High – Core functionality for daily operations.
Preconditions: Products exist in inventory with sufficient stock; staff is logged in.
Postconditions: Sale is recorded, stock levels are updated, and an invoice is generated and stored.
Actors: Shop Staff (initiates), Customer (provides input), Database (stores data), Printer (outputs invoice)
Extends: None
Flow of Events:

1. *Basic Flow*:
    - ○ Staff enters product IDs and quantities sold into the sales screen.
    - ○ System retrieves product details (price, discount) and calculates the total.
    - ○ Staff selects a payment method (manual verification assumed).
    - ○ System deducts quantities from stock, records the sale, and generates an invoice.
    - ○ Invoice is displayed and optionally printed.
2. *Alternative Flow*:
    - ○ Staff overrides the default discount with a sale-specific value.
3. *Exceptions*:
    - ○ Insufficient stock → System alerts staff and halts the sale.
    - ○ Database error → System rolls back and notifies staff.

Includes: None
Notes/Issues: Ensure invoice includes discount details for transparency.

3.3.3 Use Case #3 (U3: Generate Sales Report)



Author: Edwin Thomas
Purpose: Provide staff with sales insights to analyze performance and trends.
Requirements Traceability: F6 (sales reports and analytics)
Priority: Medium – Valuable for decision-making but not immediate operational need.
Preconditions: Sales data exists in the database; staff is logged in.
Postconditions: A sales report is generated and available for viewing/exporting.
Actors: Shop Staff (initiates), Database (provides data)
Extends: None
Flow of Events:

1. *Basic Flow*:
    - Staff selects report type (daily, weekly, monthly) and parameters (e.g., by category).
    - System queries the database and compiles sales data (amount, volume, discounts).
    - Report is displayed on the dashboard with export options (PDF/CSV).

- Staff reviews or exports the report.

2. *Alternative Flow*:
    - Staff filters for best-selling items only, focusing on trends.
3. *Exceptions*:
    - No data for the period → System displays a “No results” message.

Includes: None
Notes/Issues: Test report generation time with large datasets to meet performance goals (e.g., P4).

4 Other Non-functional Requirements

4.1 Performance Requirements

Performance requirements ensure the Shop Management System (SMS) operates efficiently under various conditions, supporting real-time operations critical for retail environments. These requirements are tied to specific functions (Section 3.2) to guide developers in optimizing system responsiveness and resource usage.

- **P1**: The system shall update inventory stock levels within 2 seconds of a sale transaction being recorded.
  Rationale: Real-time inventory tracking (F3, F4) is essential to prevent overselling and ensure accurate stock visibility for staff. A 2-second update ensures minimal delay during peak sales periods, maintaining smooth operations.
- **P2**: The system shall generate an invoice for a sale of up to 20 items within 10 seconds.
  Rationale: Fast invoice generation (F5) enhances customer experience by reducing wait times at checkout. Five seconds accommodates typical retail transactions while allowing for database queries and printing.
- **P3(optional)**: The system shall process and validate a payment (Cash, Card, or UPI) within 3 seconds under normal network conditions.
  Rationale: Quick payment processing (F6) is critical for efficient sales management, especially during high-traffic periods. The 3-second limit assumes integration with external payment gateways and accounts for network latency.
- **P4**: The system shall generate a sales report for a month's data (up to 10,000 transactions) within 10 seconds.
  Rationale: Sales analytics (F6) must be readily available for shop owners to make timely decisions. A 10-second limit ensures usability even with moderate data volumes, balancing performance with database complexity.

4.2 Safety and Security Requirements

These requirements address potential risks of data loss, unauthorized access, or operational harm, ensuring the system protects sensitive business and customer information. Safeguards are defined to comply with general retail standards and hypothetical client expectations.

- **S1**: The system shall require user authentication via username and password for all staff accessing the system.
  Rationale: Prevents unauthorized access to inventory, sales, and supplier data, protecting against internal misuse (e.g., altering stock levels or voiding sales). Passwords must be at least 8 characters with a mix of letters and numbers.
- **S2**: The system shall automatically back up the database daily to a secure local or cloud storage location.
  Rationale: Protects against data loss due to hardware failure or accidental deletion, ensuring business continuity. Backups must be encrypted and retained for at least 30 days.
- **S3**: The system shall log all user actions (e.g., product updates, sales) with timestamps and user IDs, accessible only to administrators.

  Rationale: Enables auditing to trace errors or fraudulent activities (e.g., unauthorized stock adjustments), enhancing accountability and safety of operations.
- **S4**: The system shall implement try-except blocks in all database operations to handle errors gracefully and prevent data corruption.
  Rationale: Unhandled exceptions (e.g., database connection failures, invalid inputs) could lead to incomplete transactions (e.g., stock updated but sale not recorded), compromising data integrity. Try-except blocks must catch specific exceptions (e.g., sqlite3.IntegrityError, ValueError) and roll back transactions if errors occur, ensuring no partial updates. For example, during a sale, if stock cannot be updated, the entire transaction must fail safely.

4.3 Software Quality Attributes

These attributes define additional quality characteristics beyond functionality and performance, focusing on aspects critical to customers (shop owners) and developers. Two key attributes—**Reliability** and **Adaptability**—are detailed with specific, verifiable measures and implementation strategies.

4.3.1 Reliability

Requirement: The system shall achieve 99% uptime during operational hours (e.g., 8 AM to 10 PM), excluding scheduled maintenance.

How Achieved:

- Implement redundant database connections to handle server failures.
- Use error-handling mechanisms (e.g., try-catch blocks) to gracefully recover from crashes during sales or inventory updates.
- Conduct stress testing with 500 concurrent transactions to validate stability.

Rationale: High reliability ensures the system is available during peak sales times, minimizing disruptions to shop operations. A 99% uptime allows for roughly 8 minutes of downtime per day, which is acceptable for small retail setups.

4.3.2 Adaptability

Requirement: The system shall support the addition of new product categories or inventory management features (e.g., batch tracking, expiration dates) within 2 weeks of development effort, without requiring a full system redesign.

How Achieved:

- Design a modular database schema with extendable tables (e.g., Products and Categories tables use generic fields that can accommodate new attributes via additional columns or related tables).
- Implement a flexible GUI framework (e.g., tkinter with dynamic widget generation) to allow new fields or screens to be added via configuration rather than hard coded layouts.
- Maintain a well-documented codebase with comments and a schema reference file (e.g., schema.md) to guide developers in extending the system.

Rationale: Retail shops frequently need to adapt to new product types (e.g., seasonal items) or inventory requirements (e.g., tracking perishable goods), and this adaptability ensures the system can evolve to meet these needs without significant overhaul. A 2-week development window is feasible for small enhancements, balancing flexibility with minimal disruption to existing operations. Since transaction verification is handled manually, this requirement focuses on inventory and product management, which are core to the SMS's functionality.

Appendix A – Data Dictionary

This section tracks all key variables, states, and functional requirements described in the document. It includes constants, state variables, inputs, and outputs along with their descriptions and related operations.

Variable/State	Description	Possible Values	Related Operations
productID	Unique identifier for each product	Integer (Auto-increment)	Add, Update, Delete Product
productName	Name of the product	String (Max 50 chars)	Add, Update Product
category	Category under which the product falls	String (Max 30 chars)	Assign Category, Update Product
price	Cost of a single unit of product	Decimal (e.g., 10.99)	Set Price, Update Product
stockLevel	Current stock quantity	Integer	Update after Sale, Restock
supplierID	Unique identifier for supplier	Integer	Add, Update Supplier
transactionID	Unique ID for sales transactions	Integer (Auto-increment)	Record Sale, Generate Invoice
discountPercentage	Discount applied to a product	Integer (0-100%)	Apply Discount, Update Discount
totalAmount	Final cost after discounts and tax	Decimal	Calculate Total, Process Payment
paymentMethod	Payment type used in transactions	Cash, Card, UPI	Process Payment

Appendix B - Group Log

This section contains meeting minutes, group activities, and relevant details regarding the effort put into producing this document.

Date	Participants	Activities	Remarks
20-02-2025	All Members	Initial discussion on system requirements	Defined scope and objectives
23-02-2025	Edwin, Ayaan	Completed sections 1 and 2	Drafted database schema
25-02-2025	Fahad, Johan	Worked on sections 3 and 4	Finalized functional requirements
27-02-2025	Fahad, Johan	Created use case diagrams	Added necessary revisions
01-03-2025	Edwin, Ayaan	Final review and corrections	Completed document submission