SOFTWARE REQUIREMENTS SPECIFICATION

[SRS]

for

INVENTRA

– A Shop Inventory Management System

Submitted By: Group No. 10

Under the Supervision of

Dr. Nabendu Chaki, Prof. at University of Calcutta

Mr. Deepanjan Mitra, SRF at University of Calcutta

Contents

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview

2. General Description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 general constraints

2.5 Assumptions and dependencies

3. UCD (Use Case Diagram)

4. Functional Requirements

4.1 Registration of New Businesses

4.2 Log in

4.3 Add branch

4.4 Add Staffs and Cashiers

4.5 Add New Products

4.6 Delete Existing Products

4.7 Admin -- Store Manager Communications Window

4.8 Update Stock Details

4.9 Manage and View Sales Records

4.10 Low Stock Alert

4.11 Placing Order to the Suppliers

4.12 Payment Gateway

4.13 Subscription Window

5. External Interface Requirements

5.1 User Interface

5.2 Hardware and Software Interface

5.3 Communication Interface

6. Non-Functional Requirements

6.1 Performance

6.1 Security

6.2 Usability

6.3 Reliability

6.4 Scalability

6.5 Maintainability

6.6 Integration

6.7 Branding

Index

Appendix

1. Introduction:

1.1. Purpose:

The purpose of this document is to present a detailed description of the Shop Inventory Management System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

1.2. Scope:

INVENTRA facilitates efficient management of shop operations, enabling accurate tracking, updating, and selling of products. The system is designed for administrators, shop managers, sales staff, and cashiers. Its scope includes product catalogue management, stock monitoring, sales processing, financial record-keeping, and branch-level coordination.

1.3. Definitions, acronyms and abbreviations:

- **INVENTRA: A Shop Inventory Management System.**
- **Owner: Owns the whole software system (includes the developers)**
- **Admin/Administrator: Owner of a unique registered business.**
- **Central hub: Primary address of the registered business.**
- **Branch: Business outlets other than the central hub.**
- **Store Manager: Manages a particular branch.**
- **Store Cashier: Works in a particular branch and manages transactions and records sales.**
- **Store Staff: Works in a particular branch and modifies product details.**
- **Product: A distinguishable item, which is being sold in the shop.**
- **Stock: Quantity of a particular product.**
- **Business ID: Every registered business has a unique business ID.**
- **Product ID: Every registered product has a unique Product ID.**
- **Item ID: Every unique item stored in the inventory has a unique item ID.**

1.4. References:

- **Shop_Inventory_Management_System_Use_Case_Diagram_Documentation.docx (Use Case documentation).**
- **SRS.pdf (SRS structure and templates).**
- ***IEEE Std 830-1998* IEEE Recommended Practice for Software Requirements Specifications.** IEEE Computer Society, 1998.

1.5. Overview:

This document describes the functional, non-functional, and interface requirements of the INVENTRA – a Shop Inventory Management System.

2. General Description:

2.1. Product perspective:

The Shop Inventory Management System (SIMS) aims to simplify and automate the process of managing stock, sales, and supplier interactions in retail shops. The system provides real-time product availability, billing support, staff role management, and financial tracking.

2.2. Product functions:

- **Shop Inventory Management**
- **Staff Role Management**
- **Sales, Billing and Payment Management**
- **Report and Analytics Generation**
- **Stock Monitoring and Alerts**
- **Supplier and Order Management**
- **Login and Security Management**
- **Subscription Management**

2.3. User characteristics:

2.3.1 Owner: Controls the overall system, manages subscriptions, and oversees registered businesses.

2.3.2 Administrator (Admin): Registers and manages their shop's inventory, suppliers, and sales records.

2.3.3 Store Manager: Oversees branch-level stock, confirms supplier deliveries, and reviews sales records.

2.3.4 Store Cashier: Handles sales, returns, billing, and payment transactions.

2.3.5 Staff: Updates product information such as expiry dates, manufacturing details, and prices.

2.4. General constraints:

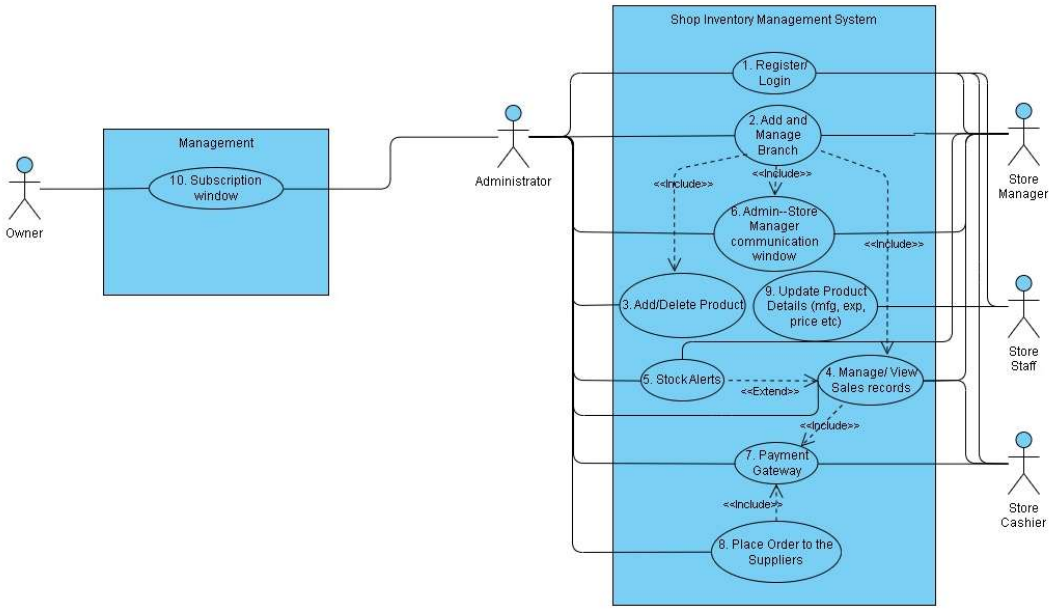
- **Real-time stock updates to ensure accurate inventory tracking across branches.**
- **Secure and scalable cloud-based system to handle multiple shops and transactions.**

- **Role-based access control for Owner, Admin, Store Manager, Cashier, and Stuffs.**
- **Compliance with standard financial transaction security protocols for payments.**
- **System must support multi-branch operations under a single Owner's subscription.**

2.5. Assumption and dependencies:

- **Assumes stable internet connectivity for real-time synchronization of sales and inventory.**
- **Assumes accurate entry of product details and stock data by Stuffs and Cashiers.**
- **Assumes business Owners maintain valid subscriptions for uninterrupted system access.**
- **Depends on third-party payment gateway integration for secure transactions.**

3. UCD (Use Case Diagram)



4. Functional Requirements:

4.1. Registration of New Businesses:

- **Purpose: The purpose of registration is to add a new business into the system by capturing essential details such as a unique email, password, business name, and address. This ensures secure identification and creates a unique Business ID that links all future branches, staff, and activities of that business.**
- **Actors: Admin.**
- **Precondition: The business is not already registered, and new to the system and the system is available for new registrations.**
- **Inputs:**
  1. **Email address**
  2. **Password**
  3. **Name of the business**
  4. **Primary address**
- **Input Table for "Registration of New Businesses" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Email address	String	Required	Must be a valid email address
Password	String	Required	Must be a valid password.
Name of the business	String	Required	Must be a valid and new business to the system entered
Primary address	String	Required	Must be in a valid form.

- **Checks:**
  1. **Email format validation => Show error if invalid.**
  2. **Check uniqueness of email and business name => error if already exists.**
  3. **Password strength check (minimum length, special characters) => prompt user if weak.**
  4. **Ensure business name and address are not empty => error if missing.**
- **Output: Unique Business ID assigned.**

- **Postcondition: A new business entry is stored in the system with unique Business ID and its linked credentials for the admin use.**

4.2. Log in:

- **Purpose: The purpose of login is to authenticate system users (Admin or Store Manager or Store Cashier or Store) or a business by verifying credentials against registered records. It ensures only authorized access to business operations, supports password recovery, and grants role-based access to dashboards, protecting sensitive business and store data from unauthorized use, thus initiates a session that allows authorized users to manage business operations.**
  - **Actors: Admin, Store manager, Store Cashier, Store Staff.**
- **Precondition: The user must be existing user and thus registered and have valid login credentials.**
  - **Inputs:**
    1. **Select category (Admin/ Store manager/ Store Cashier/ Store Staff).**
    2. **Unique Business ID for Admin / Unique Business ID + Store ID for Store Manager / Unique Business ID+ Store ID + Personal ID for Store Cashiers and Store Staffs.**
    3. **Password.**
    4. **Forget Password and Password Recovery.**
  - **Input Table for "Login" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
User Category	String (for example, options will be given such as Admin/Store Manager, cashier, staff) or ENUM	Required	Must choose a valid email category as the role.
Unique Business Id	String	Required	Must be a valid unique id assigned.
Password	String	Required	Must be a valid and matching credentials stored to the system
Forget Password	String	Required	After checking the validity of the previous inputs and on user's requests

			either user can get the recovered password or can set a new password.

- **Checks:**
  1. **Validate Business ID and Store ID and Personal ID => error if not found.**
  2. **Check if entered password matches with stored one => error if incorrect.**
  3. **If "Forget Password" selected => send OTP/email reset link.**
- **Output: Successful login session => dashboard access granted as per respective roles selected by the user.**
- **Postcondition: User is authenticated, a secure session is created and access is granted according to role (Admin or Store Manager or Store Cashier or Store Staff).**

4.3. Add Branch:

- **Purpose: The purpose of adding a branch is to let a registered business expand by creating new store locations. This process records branch location, manager details, and capacity, and generates a Store ID that connects the branch with its parent business for smooth tracking and operations.**

- **Actors: Admin.**
- **Precondition: The Business is already registered and the admin is logged in.**
- **Inputs:**
  1. **Branch Location.**
  2. **Store Manager Details (name, email/phone no.)**
  3. **Store size/ capacity.**
- **Input Table for "Add Branch" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Branch location	String	Required	Must enter a desired branch location and should not be empty
Unique Business Id	String	Required	Must be a valid unique id assigned.
Store manager details (Name,	String (name and any one of	Required	Must enter proper details and field should not be empty.

email/phone no.)	email/ phone no. or both)		
Store size/ capacity	Real	Required	After checking the validity of the previous inputs, user should enter size/capacity of the store for successful adding of the store to the system.

- **Checks:**
  1. **Verify branch location is not already existing one => error if duplicate.**
  2. **Validate store manager email/phone format => error if invalid.**
  3. **Ensure capacity is a numeric value => invalid if non-numeric.**
- **Output: Unique Store ID assigned to that respective branch.**
- **Postcondition: Branch record created and stored with Store ID, linked to the parent Business ID and a unique ID is assigned to the Store Manager.**

4.4. Add Staffs and Cashiers:

- **Purpose: The purpose of adding staff is to allow store managers of respective branches to register new employees (cashiers and other staffs) for their branches. This process captures staff details, generates a Staff ID, and associates them with the store, ensuring workforce management and tracking of responsibilities.**

  - **Actors: Store Manager.**
  - **Precondition: Store Manager is logged in with valid Store ID.**
  - **Inputs: Staff or cashier details (name, email and/or phone no.)**
  - **Input Table for "Add Staffs and Cashiers" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Staff or Cashier details (Name, email/phone no.)	String	Required	Must enter a new and non-existing email id or phone number for getting registered to the system

- **Checks:**
    1. **Validate email/ phone format => error if invalid.**
    2. **Ensure staff contact is unique => error if already exists.**
    3. **Ensure name field is not empty => error if missing.**
- **Output: Unique Staff ID is generated for each new employee.**
- **Postcondition: Staff member successfully registered, stored in the Staff Table and linked to the corresponding Store ID.**

4.5. Add New Products:

- **Purpose: The purpose of the *Add New Product* feature is to enable the Administrator to introduce new items into the inventory based on customer demand, market trends, or business requirements. This functionality ensures that the product catalog remains up to date, supports customer needs, and allows the shop to expand its offerings, thereby improving sales opportunities and customer satisfaction.**

- **Actors: Admin**
- **Precondition:**
    1. **The admin is logged in into the system.**
- **Inputs:**
    1. **Product Name**
    2. **Product Category**
    3. **Brand name**
    4. **Unit Price**
    5. **Initial Stock Quantity**
    6. **Expiration Date (if applicable)**
    7. **Supplier Details**
- **Input Table for "Add New Product" Functional Requirement:**

Input Element	Type	Optional/	Rules

		Required	
Product Name	**String**	**Required**	❖ **Cannot be NULL.** ❖ **Must be unique.**
Product Brand	**String**	**Required**	❖ **Cannot be NULL.**
Product Category	**String**	**Required**	**Cannot be NULL.**
	Real	**Required**	**Must be greater than 0.**
Initial Stock Quantity	**Integer**	**Required**	❖ **Must be ≥ 0.** ❖ **Whole numbers only.**
Expiration Date	**Date**	**Optional**	❖ **Can be NULL (if product has no expiry).** ❖ **If provided, must be greater than or equal to current date.**
Supplier Details	**String**	**Optional**	**Can be NULL (if no supplier is predefined).**

- **Checks:**

  1. **Product Name must be unique and not null.**
  2. **Product Category must be a valid category and not null.**
  3. **Unit Price must be greater than 0.**
  4. **Initial Stock Quantity must be greater than or equal to 0.**
  5. **Expiration Date (if provided) must be today's date or a future date.**
  6. **The product details must not exist in the active business database preiously.**

- **Output: A new product entry is created in the inventory database.**
- **Postcondition:**

  1. **The product catalog is updated with the newly added product.**
  2. **The product becomes available for order, sale, or further inventory operations.**

4.6. Delete Existing Products:

- **Purpose: The purpose of the Delete Existing Product feature is to allow the Administrator to remove products from the inventory that are no longer sold, have become obsolete, or are irrelevant to customer demand. This helps maintain a clean, accurate, and up-to-date inventory database while also optimizing storage space by eliminating unsold, expired, or discontinued items.**
- **Actors: Admin**
- **Precondition:**

1. **The Administrator must be authenticated and authorized.**
2. **The product must already exist in the inventory database.**
3. **The product must not have any pending or active customer orders linked to it.**

- **Inputs:**

1. **Product Name**
2. **Product Category**
3. **Reason for deletion (e.g., obsolete, expired, unsold, discontinued).**

- **Input Table for "Delete Existing Product" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Product Name	String	Required	❖ Cannot be NULL. ❖ Must be unique in the product catalog.
Product Category	ENUM	Required	❖ Cannot be NULL. ❖ Must match a valid category.
Reason for deletion	Text or ENUM	Optional	❖ The field may be NULL (optional). ❖ If a value is provided, it must match one of the allowed options in the ENUM (e.g., 'Expired', 'Obsolete', 'Not Selling', 'Discontinued').

- **Checks:**

  1. **Verify that the product exists in the database.**
  2. **Confirm that the product has no active or pending orders.**

- **Output: The product is removed from the inventory database.**
- **Postcondition: The inventory database no longer contains the deleted products stock details.**

4.7. Admin – Store Manager Communication Window:

- **Purpose: The Store manager can request admin to replenish stocks, remove or add any items etc. The admin can notify the Store Manager about delivery details.**

  The shop-level order request allows the Store Manager to formally request additional stock from the admin when product quantities are low. This ensures timely replenishment, prevents stockouts, and helps maintain smooth business operations by specifying the exact item and quantity required.

  The remove request enables the Store Manager to notify the admin to withdraw items from inventory that are either unsold for a long period or have reached their expiration date. This helps optimize storage space, reduce losses due to expired products, and maintain product quality for customers.

  Admin may reject the request or may grant the exact request made by the manager or may change the request according to sales trends. After placing order to the suppliers, the admin shares all the supplier, delivery and order related information with the Store Managers.

  After receiving the orders, the Store Manager notifies the admin that the delivery is received and the store staffs updates the necessary details of the received items.

- **Actors: Admin and Store Manager.**
- **Precondition: Admin and Store Manager must be logged in into the System.**

- **Inputs:**
  1. **Product Category**
  2. **Product Name**
  3. **Product Brand**
  4. **Product ID**
  5. **Quantity Requested**
  6. **Add to stock/Remove from stock**
  7. **Supplier contact information (Name, contact details (phone no./ email) etc.)**
  8. **Quantity Ordered**
  9. **Date of placing Order to the supplier**
  10. **Expected Delivery date**
  11. **Acknowledgements**
- **Input Table for "Admin – Store Manager Communication Window" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Product Name	**String**	**Required**	❖ **Cannot be NULL.** ❖ **Must be unique in the product catalog.**
Product Category	**String or ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must match a valid category (if using ENUM or foreign key reference to a Categories table).**
Product Brand	**String or ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must match a valid Brand (for ENUM).**
Product ID	**String or ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must be unique.**
Quantity Requested	**Integer**	**Required**	❖ **Cannot be NULL.** ❖ **Quantity > 0.**
Add to Stock/ Remove from Stock	**Boolean**	**Required**	❖ **Cannot be NULL.** ❖ **While removing Quantity <= available in stock and while ordering Quantity <= maximum capacity of the branch.**

Supplier Contact information (Name, contact details (phone no./ email) etc.)	**String**	**Required if ordered**	❖ **Details must be valid.**
Quantity Ordered	**Integer**	**Required if ordered**	❖ **Quantity > 0.**
Date of placing order to the supplier	**Date or string**	**Required if ordered**	❖ **Must be the date of placing the order.**
Expected Delivery Date	**Date or string**	**Optional**	❖ **Must be a valid date >= present date.**
Acknowledgement	**Boolean**	**Required**	❖ **Must check the box after receiving the delivery.**

- **Checks:**
    1. **Verify that the all the product details exist in the database.**
    2. **Check if the request is for removal or addition of items in the stock.**
- **Output: Stock replenishes.**
- **Postcondition: Data of the received items must be entered into the system by the staffs.**

4.8. Update Stock Details:

- **Purpose: After receiving the delivery from suppliers, store staffs enter the details of all the received items into the database. This is a reliable and easy approach to maintain the overall stock reports of the store and of the whole business. For perishable goods this method provides an easy way to maintain manufacturing and expiry date of the items. Store Manager can review and verify the details entered by the staffs to check if any inconsistency has occurred.**
- **Actors: Store Staffs, Store Manager**
- **Precondition: Store staff and/or Store Manager must be logged in into the system.**
- **Inputs:**
    1. **Product Category**
    2. **Product Name**

3. **Product Brand**
4. **Product ID**
5. **Item No.**
6. **Manufacture date**
7. **Expiry date**
8. **Price**
9. **Date of adding the product in stock**

- **Input Table for "Update Stock Details" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Product Name	**String**	**Required**	❖ **Cannot be NULL.** ❖ **Must be unique in the product catalog.**
Product Category	**String or ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must match a valid category (if using ENUM or foreign key reference to a Categories table).**
Product Brand	**ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must match a valid Brand.**
Product ID	**String or ENUM**	**Required**	❖ **Cannot be NULL.** ❖ **Must be unique.**
Item No.	**Integer**	**Required**	❖ **Must be unique for each item entered at that day (starting from 0 to the max quantity added to the stock)**
Manufacture date	**Date or string**	**Required**	❖ **Cannot be NULL.** ❖ **Must be <= present date.**
Expiry date	**Date or string**	**Required (for perishable goods)**	❖ **Must be >= present date.**
Price	**Real**	**Required**	❖ **Price must be >0**
Date of adding the product in stock	**Date or string**	**Required**	❖ **Present date (auto captured by the system)**

- **Checks:**
    1. **The product details must exist in the store database.**
    2. **The current stock is less than the maximum capacity of that product.**

- **Output:**
    1. **System generates a specific Item ID for each and every item based on product category, product name, product ID, product brand and the unique Item No. entered etc.**
    2. **The items are added in the database.**
    3. **It will notify the staffs and Store Manager a fixed time (set by the Store Manager) before the expiry date of the perishable goods.**
- **Postcondition: The inventory database now has the updated stock details of the added products.**

4.9. Manage and View Sales Records:

- **Purpose: Store Cashiers records each sale or return in the system, which automatically updates the stock details. It allows store manager to review and verify the sales records of the respective branch, while the admin is allowed to view overall sales records across all branches. Updated stock information based on sales can also trigger low-stock alerts.**
- **Actors: Admin, Store Manager and Store Cashier**
- **Precondition:**
    1. **The admin, store manager or the cashier must be logged into the system.**
    2. **The branch where the transaction is recorded must already exist in the system.**
    3. **The cashier must be active, registered employee under that branch.**
    4. **The product (Item ID) must be registered in the system and have stock available (for sales).**
    5. **A valid payment method must be configured in the system (for sales).**
- **Inputs:**
    1. **Specific Item ID, Branch ID, Cashier ID, Order ID, Customer Name, Status**
    2. **Date Sold**
    3. **Date Returned**
    4. **Price, Discount, Other Charges and Net Price**
    5. **Payment Method**
- **Input Table for "Manage and View Sales Records" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Branch ID	String	Required	Must match registered Branch ID in the system.
Cashier ID	String	Required	Must be a registered cashier/ employee ID.
Specific Item ID	String	Required	Must match the stored item ID for particular store.
Customer Name	String	Required (Optional for return)	• Must be a non-empty string containing only valid alphabetic characters (A–Z, a–z) and common separators (space, hyphen, apostrophe). • Minimum length: 2 characters. • Maximum length: 100 characters.
Order ID	String	Required	Must be unique for a particular item.
Status	ENUM (Sold, Returned)	Required	Value must be one of the predefined states {Sold, Returned}.
Date Sold	Date	Required (for items sold)	Must be a valid date.
Date Returned	Date	Required (for items returned)	Must be a valid date within the last return date for the product. Return date >= date sold.
Payment Method	ENUM (Cash, UPI, Net Banking, Debit Card etc.)	Required	Must match available payment methods configured in the system.
Selling Price	Real	Required	Must be greater than 0. Must match the listed price of the item at the time of sale.
Discount	Real	Optional	Must be ≥ 0 and ≤ Selling Price. If percentage, must be between 0–100.
Other Charges	Real	Optional	Must be ≥ 0. Represents additional costs like packaging, delivery, GST etc.
Net Price	Real	Required	Must be calculated as (Selling Price – Discount + Other Charges). Must be ≥ 0.

- **Checks:**
  1. **Verify that the Item ID exists in the system and belongs to the logged-in branch.**
  2. **Validate that the Order ID is unique for that transaction.**
  3. **Ensure the Customer Name meets defined input rules.**
  4. **Confirm that Date Sold or Date Returned is a valid date and logically consistent (i.e. return date ≥ sold date, and within return policy period).**
  5. **Check that the Selling Price matches the current listed price for the item.**

6. **Verify that Discount and Other Charges follow the defined rules (no negative or illogical values).**
7. **Validate that Net Price is correctly calculated.**
8. **Ensure that the selected Payment Method is one of the allowed methods in the system.**
9. **On return, check that the item was actually sold earlier (can't return a non-sold item).**

- **Output:**
  1. **Stocks are updated.**
  2. **Sales record is stored.**
  3. **Graphical views of the sales records are generated.**
- **Postcondition: The specific item is marked as sold on sell or as available or defected on return.**

4.10. Low Stock Alert:

- **Purpose: A low stock alert, generated by the system, serves as an early warning mechanism that notifies the store manager and administrator when an item's quantity drops below a predefined threshold. This feature helps maintain an optimal inventory balance by preventing both stockouts. Timely restocking ensures product availability, reduces the risk of customer dissatisfaction due to unavailability, and provides managers with real-time insights for informed replenishment and purchasing decisions.**

- **Actors: Store Manager and Admin.**
- **Precondition: A threshold value for minimum stock has been predefined for each product.**
- **Inputs:**
  1. **Current Stock Quantity.**
  2. **Threshold value.**
  3. **Product Details.**
- **Input Table for "Low Stock Alerts" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Current Stock Quantity	**Integer**	**Required**	❖ **Must be ≥ 0 (cannot be negative).** ❖ **Typically, a whole number.** ❖ **Should support a reasonable maximum**
Threshold	**Integer**	**Required**	❖ **Must be ≥ 0 (cannot be negative).**

Value			❖ **Should be less than or equal to the maximum stock capacity of the product.** ❖ **Defined once per product by Store Manager.** ❖ **Can have a default value (e.g., 5 or 10 units) but should be customizable.**
Product Details	**String**	**Required**	**Must match the stored Product ID for particular store.**
User Details	**String**	**Required**	**Notifications must be mapped to the respective user IDs (Admin, Store Manager) to ensure delivery to the correct recipients.**

- **Checks:**

1. **Ensure the product's current stock quantity < threshold value**.
2. **Confirm that the threshold value is defined in the database**.
3. **Verify that the recipient user ID (Admin/Store Manager) exists in the system.**
4. **Ensure stock quantity and threshold values are non-negative integers**.

- **Output: When the above check is true then A low stock alert notification containing product details is sent to the Admin and Store Manager (via system dashboard, email, or SMS depending on implementation).**
- **Postcondition: The item is flagged as "Low Stock" in the inventory system until replenished.**

4.11. Placing Order to the Suppliers:

- **Purpose: The Administrator/Admin places purchase orders to the suppliers for stock replenishment or new product launches. It is integrated with the Payment Gateway to process payments securely and ensures smooth resourcing of stocks.**
  - **Actors: Admin.**
  - **Precondition:**
    1. **The admin must be logged into the system.**
    2. **The supplier must be registered and active in the system.**

- **Inputs:**
  1. **Order ID**
  2. **Product Category**
  3. **Product Name**
  4. **Product Brand**
  5. **Quantity**
  6. **Supplier Name**
  7. **Cost Price**
  8. **Delivery Locations**

- **Input Table for "Place Order to the Suppliers" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Order ID	String	Required	Must be unique.
Product Category	String	Required	Must be a valid category.
Product Name	String	Required	Must be a valid product name.
Product Brand	String	Required	Must be a valid brand name.
Quantity	Integer	Required	Must be available at the supplier end.
Supplier Name	String	Required	Must be a registered supplier for that Business.
Payment Method	ENUM (Cash, UPI, Net Banking, Debit Card etc.)	Required	Must match available payment methods configured in the system.
Cost Price	Real	Required	Must be a valid amount, fixed by the supplier.
Delivery Locations	String	Required	Must be a registered branch (for that specific business) location.

- **Checks:**
  1. **Verify that Order ID is unique.**
  2. **Validate that Supplier Name exists, is active, and is linked to the product category.**
  3. **Ensure Quantity requested ≤ available supplier stock.**
  4. **Confirm Cost Price matches supplier's registered price list.**
  5. **Validate Delivery Locations as registered branches.**
  6. **Confirm Payment Method and process securely via payment gateway.**

- **Output:**
    1. **System generates Purchase Order Receipt with order details and expected delivery date.**
    2. **Admin gets an order confirmation receipt.**

- **Postcondition: Admin informs the store managers of the exact branches about the supplier details, products to be delivered and delivery date.**

4.12. Payment Gateway:

- **Purpose: The integrated payment gateway supports transactions from the two ends of the business, i.e., from customers to business (Customer <--> Cashier) and from business to suppliers (Admin <--> Supplier). This gateway felicitates an easy way to maintain overall transaction details (and sales records). This gateway is integrated with other globally accepted payment systems (like PayPal, UPI, Net Banking, Debit card, Cash etc.) to provide a smooth payment experience. This payment gateway combinedly helps to maintain the sales records.**
- **Actors: Admin and Store Cashier**
- **Precondition: Admin or Store Cashier must be logged in into the system.**
- **Inputs:**
    1. **Amount to be paid**
    2. **Payment method**
    3. **Customer/Supplier details (name, phone no./email)**
    4. **Store Cashier's unique ID**
    5. **Payment date**
- **Input Table for "Payment Gateway" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Amount to be paid	Real	Required	Must be >0
Payment Method	String	Required	Cannot be NULL. Must be a valid payment method
Customer/Supplier details	String	Required	Valid details must be entered

Payment Date	Date or string	Required	Present date, captured from system
Store Cashier's unique ID	String	Required for customer side transactions	Auto captured from the System

- **Checks: Amount paid must be greater than 0 and if paid other than cash method, the user must be a valid user of that third party application.**
- **Output:**
  1. **A unique transaction ID.**
  2. **Generates invoice for the customers.**
- **Postcondition: The payment received must be verified by the Store manager and the Admin for any inconsistencies.**

4.13. Subscription Window:

- **Purpose: Owner of a registered business (shop/store) purchases and activates a subscription plan (monthly or yearly). The subscription grants access to the inventory management facilities provided by the system. It ensures owner can view and keep track of the active subscribers.**
- **Actors: Admin, Owner**
- **Precondition:**
  1. **The business must be registered to the system.**
  2. **The admin must be logged into the system.**
  3. **Available subscription plans (monthly, yearly) must be predefined in the system.**
- **Inputs:**
  1. **Subscription Type**
  2. **Start and end dates**
  3. **Business ID**
  4. **Payment Method**
  5. **Transaction ID**
- **Input Table for "Subscription Window" Functional Requirement:**

Input Element	Type	Optional/ Required	Rules
Subscription Type (monthly/ yearly)	String	Required	Must be predefined in the system.
Start and end dates	Date or string	Required	Auto generated.
Business ID	String	Required	Must be registered business ID.
Payment Method	ENUM (Cash, UPI, Net Banking, Debit Card etc.)	Required	Must match available payment methods configured in the system.
Transaction ID	String	Required	Must be unique.

- **Checks:**

1. **Checks if the Business ID is registered or not.**
2. **Ensures that the subscription type is predefined in the system.**
3. **Check Payment Details securely via integrated payment gateway.**
4. **Confirm Transaction ID is unique.**

- **Output:**

1. **The subscription is successfully activated and confirmation message is sent.**
2. **Digital Invoice is generated.**
3. **Account status is updated. "Active Subscription" with validity period shown in the dashboard.**

- **Postcondition:**

1. **The records the subscription start date and expiry date.**
2. **The user gains access to all facilities/features allowed by the chosen subscription plan.**
3. **The digital invoice is sent to the user (email/ SMS/ notification).**

5.External Interface Requirements:

5.1 User Interface:

- **A web-based interface that allows Admins, Store Managers, Store Cashier and Store Staffs to interact with the system.**
- **The interface must be intuitive and responsive, providing real-time data on product availability, payment processing.**

5.2 Hardware and Software Interface:

- **Compatible with cloud-based infrastructure for secure, scalable data storage and processing.**
- **Payment gateways integration (e.g., Stripe, PayPal).**

5.3 Communication Interface:

- **Secure HTTPS for all web interactions.**
- **Integration with email/SMS gateways for notifications (Stock alerts, delivery updates).**

6. Non-Functional Requirements:

6.1 Performance

- **ID: NFR-001**
- **Category: Performance**
- **Description: The system shall respond to inventory updates, billing, and stock alerts within 2 seconds for 95% of transactions under a load of 500 concurrent users.**
- **Rationale: To provide a fast and responsive user experience under expected usage levels.**
- **Acceptance Criteria: Load testing should demonstrate that response times do not exceed 2 seconds for 95% of transactions with 500 concurrent users.**
- **Priority: High**

6.2 Security

- **ID: NFR-002**
- **Category: Security**
- **Description: The system shall ensure secure payment processing by integrating a trusted and PCI-DSS compliant payment gateway. All financial transactions must use encrypted communication (HTTPS/TLS), and sensitive customer payment data shall never be stored in plain text. Role-based access control will still be applied to protect operational and inventory data.**
- **Rationale: To protect customer payment information and company financial data from fraud, unauthorized access, and data breaches.**
- **Acceptance Criteria:**
    - **All payment transactions are processed through a secure, verified payment gateway.**
    - **Payment details are encrypted in transit and not stored locally.**
    - **Security audits confirm no vulnerabilities in the payment process.**
- **Priority: High**

6.3 Usability

- **ID: NFR-003**
- **Description: The system shall provide a simple and intuitive interface for all actors:**
    - **Cashiers: A Point of Sale (POS) screen with barcode scanning support for quick billing.**
    - **Store Managers: A dashboard to view stock levels, approve orders, and generate sales/stock reports.**
    - **Admins: An inventory management interface for adding/removing products and branch.**

The interface must be responsive and optimized for both desktop and mobile devices (e.g., tablets at billing counters).

- **Rationale: To ensure that all actors—cashiers, managers, and admins—can operate the system efficiently without advanced technical knowledge.**
- **Acceptance Criteria: It demonstrates that users —cashiers, managers, and admins— can complete common tasks efficiently and without errors.**
- **Priority: Medium**

6.4 Reliability

- **ID: NFR-004**
- **Category: Reliability**
- **Description: Ensure 99.9% uptime to avoid disruptions in service availability.**
- **Rationale: To minimize downtime and ensure that the system is always available for users to use.**
- **Acceptance Criteria: System monitoring should track uptime and identify any issues that could lead to downtime.**
- **Priority: High**

6.5 Scalability

- **ID: NFR-005**
- **Category: Scalability**

- **Description: The system should be able to scale seamlessly to handle an increasing number of users, products, suppliers, and transaction records without performance degradation. It must support expansion across new regions, multiple business units, and higher data volumes as the organization grows.**
- **Rationale: To accommodate future growth and expansion of the business.**
- **Acceptance Criteria: Performance testing should demonstrate that the system can handle increased load without degradation in performance.**
- **Priority: Medium**

6.6 Maintainability

- **ID: NFR-006**
- **Category: Maintainability Description: The system should be easy to maintain and update.**
- **Description: The system should be designed in a modular and well-documented manner so that bug fixes, feature enhancements, and updates can be applied with minimal disruption. It should follow coding standards and provide clear error logging to aid troubleshooting.**
- **Rationale: To reduce the cost and effort required to maintain and update the system.**
- **Acceptance Criteria: Code reviews and regular updates should be conducted to ensure the system's maintainability.**
- **Priority: Medium**

6.7 Integration

- **ID: NFR-007**
- **Category: Integration**
- **Description: The system should be able to integrate with other systems, such as payment gateways.**
- **Rationale: To streamline operations and provide a seamless experience for customers, suppliers and ease in receiving payments for cashier.**

- **Acceptance Criteria: Integration testing should be conducted to ensure that the system can integrate with other systems successfully.**
- **Priority: Medium**

6.8 Branding

- **ID: NFR-009**
- **Category: Branding**
- **Description: The system should reflect the business, company's branding and identity.**
- **Rationale: To create a consistent and professional look and feel for the system.**
- **Acceptance Criteria: The system should incorporate the company's logo, colors, and fonts.**
- **Priority: Low**

Index

A

- **Acronyms & Abbreviations – Section 1.3**
- **Add Branch – Section 4.3**
- **Add Staffs and Cashiers – Section 4.4**
- **Administrator – Section 1.3, 2.3 (2)**
- **Admin – Store Manager Communication Window – Section 4.7**
- **Assumptions and Dependencies – Section 2.5**

B

- **Branch – Section 1.3 (Definition)**
- **Branding – Section 6.8**

C

- **Cashier – Section 1.3 (Definition)**
- **Communication Interface – Section 5 (3rd point)**

D

- **Delete Existing Products – Section 4.6**
- **Definitions, Acronyms & Abbreviations – Section 1.3**

E

- **External Interface Requirements – Section 5.3**

F

- **Functional Requirements – Section 4**

G

- **General Constraints – Section 2.4**
- **General Description – Section 2**

I

- **Index – Final Section**
- **Inventory Management – Section 2.2 (Product Functions)**
- **Introduction – Section 1**

L

- **Login – Section 4.2**
- **Low Stock Alert – Section 4.10**

M

- **Maintainability – Section 6.6**
- **Manage and View Sales Records – Section 4.9**

N

- **Non-Functional Requirements – Section 6**

O

- **Overview – Section 1.5**
- **Owner – Section 1.3 (Definition)**

P

- **Payment Gateway – Section 4.12**

- **Performance – Section 6.1**
- **Place Order to the Suppliers – Section 4.11**
- **Product Functions – Section 2.2**
- **Purpose – Section 1.1, and repeated in Functional Requirements**

R

- **References – Section 1.4**
- **Registration of New Businesses – Section 4.1**
- **Reliability – Section 6.4**
- **Role-based Access – Section 2.4 (Constraints)**

S

- **Scalability – Section 6.5**
- **Scope – Section 1.2**
- **Security – Section 6.2**
- **Store Manager – Section 1.3 (Definitions)**
- **Stock Monitoring – Section 5.1 (Product Functions)**
- **Subscription Window – Section 4.13**

U

- **Update Stock Details – Section 4.8**
- **Usability – Section 6.3**
- **User Characteristics – Section 2.3**
- **UCD (Use Case Diagram) – Section 3**

Appendix

-Supporting Information

A.1 Sample Input/Output Formats

1. Registration of New Business (Input Example):

Email: ravishaw_freshmart@example.com

Password: [Encrypted String] *******

Business Name: "FreshMart"

Primary Address: "12, Green Market Road, City A"

Output: Business ID = BIZ-1024

2. Add New Product (Input Example):

Product Name: "Amul Butter 500g"

Brand Name: “Amul”

Product Category: "Dairy"

Unit Price: 250

Capacity: 100

Expiry Date: 15-12-2025

Supplier: "Amul Pvt Ltd"

Output: Product ID = PRD-DAIRY-001

3. Sales Transaction (Output Example):

Customer: "Chetan Gupta"

Order ID: ORD-2891

Items Sold: [PRD-DAIRY-001 x2]

Net Price: ₹500.00

Payment Mode: UPI

Output: Invoice INV-2891 generated.

A.2 Supporting / Background Information

- **Retail shops often face overstocking or understocking issues leading to losses.**
- **Manual inventory systems result in errors, inefficiency, and fraud risks.**
- **Cloud-based solutions like SIMS ensure real-time synchronization across branches, improving decision-making.**

A.3 Problem Description (to be solved by SIMS)

- **Difficulty in tracking product expiry dates in perishable goods.**
- **Lack of a centralized system for multi-branch operations.**
- **Manual handling of supplier orders, payments, and stock alerts.**
- **Poor visibility of sales trends without automated reports.**

A.4 Special Instructions

- **Security: All customer and payment data must be encrypted (HTTPS/TLS).**
- **Export Requirements: Reports should be downloadable in PDF/Excel format.**
- **Initial Loading: Admin should preload product categories and supplier details for smooth onboarding.**
- **Data Backup: Daily automated backup to ensure no data loss.**

A.5 Future Enhancements (Planned)

- **Barcode / QR Code Integration: Auto-scanning for faster billing.**
- **AI-powered Sales Forecasting: Predict demand based on seasonal and historical sales.**
- **Mobile App Extension: For store managers to approve stock requests remotely.**
- **Offline Sync Mode: Allow transactions during internet downtime, sync later.**

A.6 Special Instructions

- **Security: No sensitive payment data stored locally; encryption is mandatory.**
- **Audit Trail: All stock updates, sales, and deletions must be logged with timestamps.**
- **Backup: Automated daily backup stored in cloud servers.**

- **User Training: Initial onboarding documentation and video tutorials for staff.**