# Software Requirements Specification (SRS) Document

# 1. Introduction

## 1.1 Purpose

The Medical Store Inventory System is designed to manage stock efficiently by allowing pharmacists, store managers, and cashiers to track inventory, update stock, remove expired medicines, and monitor sales transactions.

### **1.2 Scope**

The system provides functionalities such as:  
• Adding, updating, and deleting medicines.  
• Tracking expiry dates and managing low-stock alerts.  
• Role-based access control for pharmacists, store managers, and cashiers.  
• Integration with SQLite for data storage using Flask.

Technology Stack

* Backend: Python, Flask
* Database: SQLAlchemy (SQLite)
* Frontend: HTML, CSS, JavaScript, Bootstrap 5
* Visualization: Chart.js
* Authentication: Flask-Login

### **1.3 Definitions, Acronyms, and Abbreviations**

• Flask: A micro web framework for Python used for building web applications.  
• SQLite: A lightweight, serverless, self-contained SQL database engine used for data storage.

### **2. System Functional Requirements**

#### **Inventory Management (Pharmacist & Store Manager)**

2.1 Add New Medicine  
• Functional Requirement:  
The system must allow pharmacists to add new medicines to the inventory. The system should capture all necessary details such as name, category, price, quantity, and expiry date.

2.2 Update Medicine Information  
• Functional Requirement:  
The system must allow pharmacists to update the information of existing medicines in the inventory. This includes updating the price, quantity, category, or other relevant details.

2.3 Delete Expired Medicines  
• Functional Requirement:  
The system must allow store managers to identify expired medicines and remove them from the inventory. The system should automatically update stock levels when expired items are deleted.

2.4 Monitor Stock Levels  
• Functional Requirement:  
The system must allow store managers to monitor current stock levels for all medicines. It should show stock information in real-time and flag low-stock medicines for reordering.

2.5 Check Stock Before Sale  
• Functional Requirement:  
The system must allow cashiers to check the availability of a particular medicine before completing a sale. If the medicine is unavailable, the cashier should be able to inform the customer and prevent the sale.

2.6 View Entire Inventory  
• Functional Requirement:  
The system must allow pharmacists to view the entire inventory, including all available medicines with their details, such as stock level, price, and expiration date.

2.7 View Stock Status Indicators  
• Functional Requirement:  
The system must visually indicate the status of stock for each product (e.g., well-stocked, low-stock, or out of stock) to help users quickly assess inventory health.

#### **Sales Transactions (Cashier & Store Manager)**

2.8 Process Sales Transaction  
• Functional Requirement:  
The system must support the process of completing a sale by updating stock levels, generating invoices, and recording transaction details. The system must validate the availability of stock and provide a detailed sales invoice.

2.9 Generate Sales and Inventory Reports  
• Functional Requirement:  
The system must generate detailed reports on both sales and inventory levels based on specified time periods (daily, weekly, monthly). Sales reports should include total sales, revenue, and product demand data to help the store manager analyze business performance. Inventory reports should include product quantities, stock status, and restocking recommendations, enabling the store manager to make informed restocking decisions and optimize inventory management.

2.10 Send Low Stock Notifications  
• Functional Requirement:  
The system must notify store managers automatically when stock levels of a particular medicine fall below a defined threshold. Notifications should be delivered in real-time.

2.11 Export Inventory and Sales Data  
• Functional Requirement:  
The system must allow the export of both inventory and sales data to a CSV file. The inventory data should include product details, stock levels, and categories, enabling external analysis and reporting. The sales data should include transaction details, revenue, and product sales information, supporting financial analysis and reporting purposes.

#### **Performance & Security**

2.12 View Monthly Revenue Data  
• Functional Requirement:  
The system must generate and display monthly revenue data for the store manager. It should include total revenue and product sales information for the selected period.

2.13 View Top-Selling Products  
• Functional Requirement:  
The system must display a list of top-selling products based on sales data. The store manager should be able to use this data for informed purchasing and restocking decisions.

2.14 View Sales by Category  
• Functional Requirement:  
The system must allow store managers to view sales by product category, helping to identify trends and make decisions based on market demand.

### **System Non-Functional Requirements**

These are requirements that focus on how the system should perform in terms of usability, security, performance, reliability, and other aspects.

#### **Usability Requirements**

1. Responsive Interface
   * System must be responsive for efficient navigation.
2. Dark Mode Option
   * Provide a dark mode option for user comfort.
3. Visual Cues
   * Clear visual indicators (color coding, badges) for stock status.
4. Helpful Error Messages
   * Error messages should clearly explain issues.
5. Confirmation Messages
   * System provides confirmation after successful actions.

#### **Performance Requirements**

1. Page Load Performance
   * Pages should load within 3 seconds.
2. Report Generation Speed
   * Reports must generate within 5 seconds.
3. Concurrent User Handling
   * System supports at least 100 concurrent users without performance loss.

#### **Reliability Requirements**

1. Data Validation
   * Perform validation checks to ensure data correctness.
2. Atomic Transactions

* Database transactions must be atomic.

1. Automatic Data Backups

* Regular automatic backups to prevent data loss.

#### **Security Requirements**

1. Password Hashing

* Passwords stored with hashing for protection.

1. Session Timeout

* Automatic session timeout after inactivity period.

1. Granular Access Controls

* Precise, role-based data access controls.

#### **Maintainability Requirements**

1. Modular Code Structure

* Ensure modularity for future enhancements.

1. Clear Documentation

* Maintain clear documentation of system components.

1. Consistent Naming Conventions

* Follow consistent naming conventions for readability.

#### **Compatibility Requirements**

1. Cross-browser Compatibility

* Support major browsers (Chrome, Firefox, Safari, Edge).

1. Responsive Design

* Interface is responsive across different screen sizes.

#### **Scalability Requirements**

1. Inventory Scalability

* System handles expanding inventory size without performance issues.

1. Transaction Volume Scalability

* System manages growing transactions volume efficiently.

### **Product Requirements**

These are requirements that define the system's functionality, how it interacts with users, and its role in the product.

* All the Functional Requirements (from your initial prompt) are classified under Product Requirements. This includes the actions users can perform, such as logging in, adding medicines, generating reports, managing inventory, etc.

### **Organizational Requirements**

These requirements are more related to the organization's internal processes, code structure, documentation, and development practices.

* Modular Code Structure
* Clear Documentation
* Consistent Naming Conventions

These organizational practices ensure that the development process is efficient, maintainable, and scalable in the long term.

### **External Requirements**

These requirements are typically about interacting with external systems, software, or services. In this case, they mostly focus on how the system must integrate with external platforms or perform in an external environment.

* Cross-browser Compatibility
  + Support for major browsers (Chrome, Firefox, Safari, Edge).
* Responsive Design
  + Ensure the interface adapts to various screen sizes (e.g., mobile, tablet, desktop).

# 4. Use Case Diagram

# 

### **5 USER STORIES**

### **5.1 Add New Medicine**

* Role: Pharmacist
* Goal: Add new medicines to the inventory
* Reason: Track stock availability and update pricing.

Pre-conditions:  
• The pharmacist must be logged into the system.  
• The pharmacist must have the necessary permissions to add medicines.

Post-conditions:  
• The new medicine is added to the inventory database.  
• The stock level of the new medicine is updated.

### **5.2 Update Medicine Information**

* Role: Pharmacist
* Goal: Update the information of existing medicines in the inventory
* Reason: Ensure accurate stock information.

Pre-conditions:  
• The pharmacist must be logged into the system.  
• The medicine must already exist in the inventory.

Post-conditions:  
• The updated medicine information is stored in the system.  
• Any discrepancies in pricing or stock levels are corrected.

### **5.3 Delete Expired Medicines**

* Role: Store Manager
* Goal: Remove expired medicines from the inventory
* Reason: Ensure outdated medicines are not sold.

Pre-conditions:  
• The store manager must be logged into the system.  
• Expired medicines must be identified in the system.

Post-conditions:  
• Expired medicines are removed from the inventory.  
• The stock count is updated accordingly.

### **5.4 Monitor Stock Levels**

* Role: Store Manager
* Goal: Track stock availability
* Reason: Prevent shortages by restocking in time.

Pre-conditions:  
• The store manager must be logged into the system.  
• The inventory system must contain up-to-date stock information.

Post-conditions:  
• The store manager receives real-time stock level insights.  
• Low-stock medicines are flagged for restocking.

### **5.5 Check Stock Before Sale**

* Role: Cashier
* Goal: Verify stock before completing a sale
* Reason: Ensure requested medicines are available.

Pre-conditions:  
• The cashier must be logged into the system.  
• The medicine must be available in stock.

Post-conditions:  
• The cashier confirms stock availability before processing the sale.  
• If stock is unavailable, the cashier informs the customer.

### **5.6 View Entire Inventory**

* Role: Pharmacist
* Goal: View the entire inventory
* Reason: Track all available medicines.

Pre-conditions:  
• The pharmacist must be logged into the system.

Post-conditions:  
• The pharmacist has access to the full inventory list.  
• The inventory is displayed with relevant information.

### **5.7 View Stock Status Indicators**

* Role: User
* Goal: See clear stock status indicators (well-stocked, low-stock, out of stock)
* Reason: Quickly assess inventory conditions.

Pre-conditions:  
• The user must be logged into the system.  
• The inventory data must be up-to-date.

Post-conditions:  
• Stock status indicators are displayed for each product.  
• The user can easily identify stock availability at a glance.

### **5.8 Process Sales Transaction**

* Role: Cashier
* Goal: Complete a sales transaction
* Reason: Ensure medicines are sold with proper billing and stock updates.

Pre-conditions:  
• The cashier must be logged into the system.  
• The medicines being sold must be available in stock.  
• The customer details (if required) must be entered.

Post-conditions:  
• The system generates an invoice for the sale.  
• The stock levels are updated accordingly.  
• The transaction details are stored in the system for reporting purposes.

### **5.9 Generate Sales and Inventory Reports**

* Role: Store Manager
* Goal: Generate reports on both sales and inventory levels
* Reason: Analyze business performance and make informed decisions.

Pre-conditions:  
• The store manager must be logged into the system.  
• The system must have recorded sales and inventory data.

Post-conditions:  
• The system generates detailed reports on sales and inventory.  
• Sales reports include total sales, revenue, and product demand data.  
• Inventory reports include product quantities, stock status, and restocking recommendations.

### **5.10 Send Low Stock Notifications**

* Role: System
* Goal: Notify store managers when stock is low
* Reason: Ensure timely restocking.

Pre-conditions:  
• The system must have predefined stock threshold levels.  
• The store manager must be registered to receive notifications.

Post-conditions:  
• The system automatically notifies the store manager when stock is below the threshold.  
• The store manager can take action to reorder medicines.

### **5.11 Export Inventory and Sales Data**

* Role: Store Manager
* Goal: Export both inventory and sales data to CSV
* Reason: Facilitate financial analysis and external reporting.

Pre-conditions:  
• The store manager must be logged into the system.  
• The system must have up-to-date inventory and sales data.

Post-conditions:  
• The inventory and sales data are exported into CSV files.  
• The store manager receives the exported data for further analysis.

### **5.12 View Monthly Revenue Data**

* Role: Store Manager
* Goal: View monthly revenue data
* Reason: Track financial performance regularly.

Pre-conditions:  
• The store manager must be logged into the system.  
• The system must have recorded sales transactions.

Post-conditions:  
• Monthly revenue data is displayed.  
• The store manager can assess the revenue trends for the month.

### **5.13 View Top-Selling Products**

* Role: Store Manager
* Goal: Identify top-selling products
* Reason: Inform purchasing and restocking decisions.

Pre-conditions:  
• The store manager must be logged into the system.  
• The system must have sales data available.

Post-conditions:  
• A list of top-selling products is displayed.  
• The store manager can make informed decisions about future purchases.

### **5.14 View Sales by Category**

* Role: Store Manager
* Goal: Analyze sales performance by category
* Reason: Understand market demand patterns.

Pre-conditions:  
• The store manager must be logged into the system.  
• The system must have categorized sales data.

Post-conditions:  
• Sales by category are displayed for analysis.  
• The store manager can see product demand patterns by category.

# 6. Sequence Diagrams

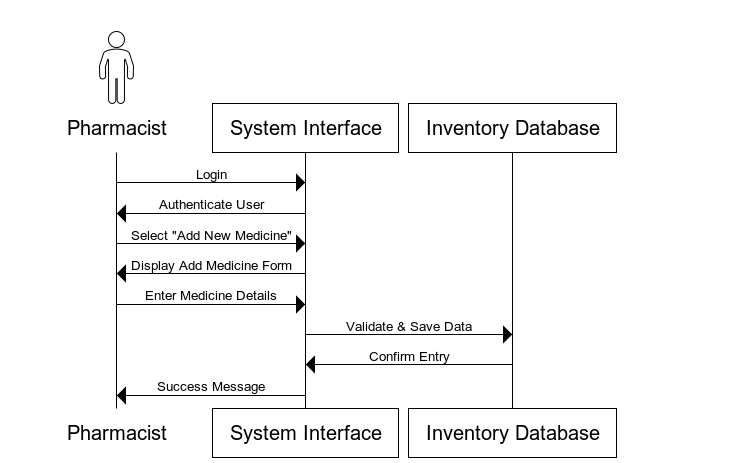
# • Adding a New Medicine

### **Actors:**

1. Pharmacist (User)
2. System (Medical Store Inventory System)
3. Inventory Database (Database where medicines are stored)

### **Flow of Events:**

1. Pharmacist logs into the system.
   * System verifies login credentials.
2. Pharmacist selects the "Add New Medicine" option.
3. Pharmacist enters medicine details (name, category, price, expiry date, stock level).
4. System validates the entered details.
   * If valid, proceed to step 5.
   * If invalid, system prompts for corrections.
5. System sends data to the Inventory Database.
6. Database stores the new medicine information and updates stock levels.
7. System confirms successful addition and notifies the pharmacist.

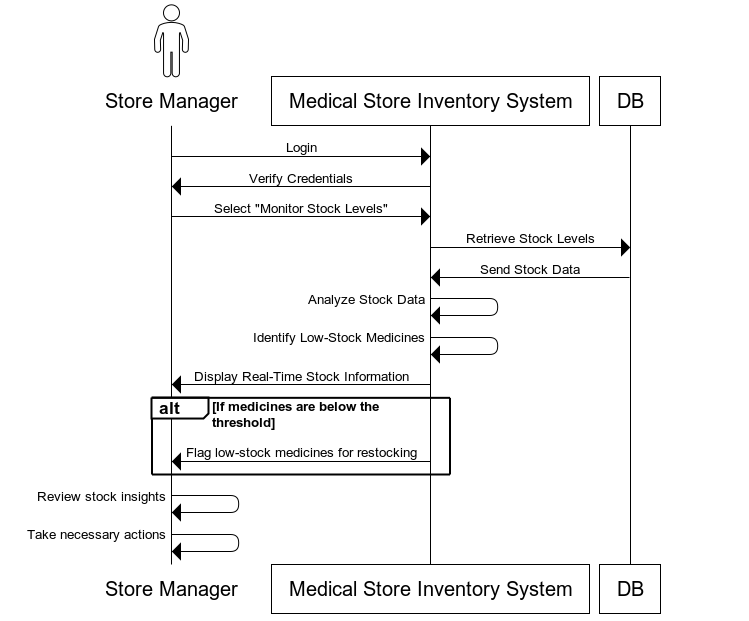
 **Monitoring Stock Levels**

### **Actors:**

1. Store Manager (User)
2. System (Medical Store Inventory System)
3. Inventory Database (Stores stock information)

### **Flow of Events:**

1. Store Manager logs into the system.
   * System verifies login credentials.
2. Store Manager selects the "Monitor Stock Levels" option.
3. System retrieves stock levels from the Inventory Database.
4. System analyzes stock data and identifies low-stock medicines.
5. System displays real-time stock information to the Store Manager.
   * If any medicines are below the threshold, they are flagged for restocking.
6. Store Manager reviews stock insights and takes necessary actions.



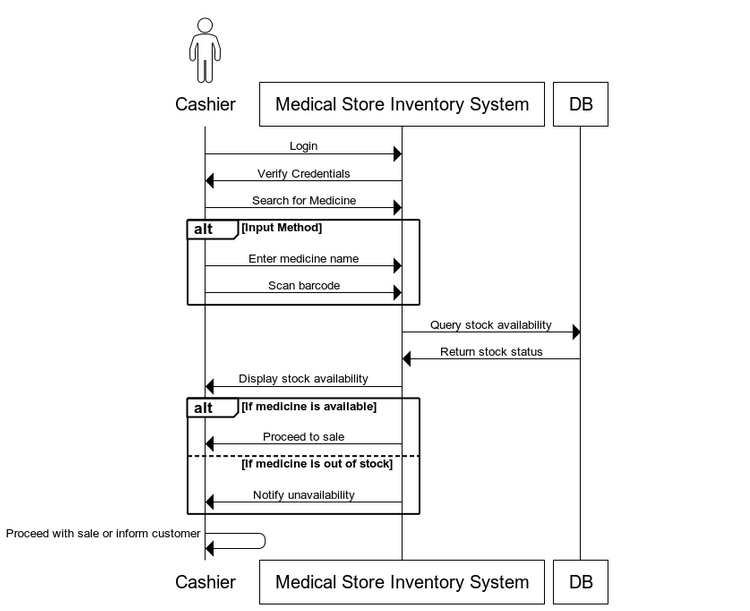
**• Checking Stock Before Sale**

### **Actors:**

1. Cashier (User)
2. System (Medical Store Inventory System)
3. Inventory Database (Stores stock information)

### **Flow of Events:**

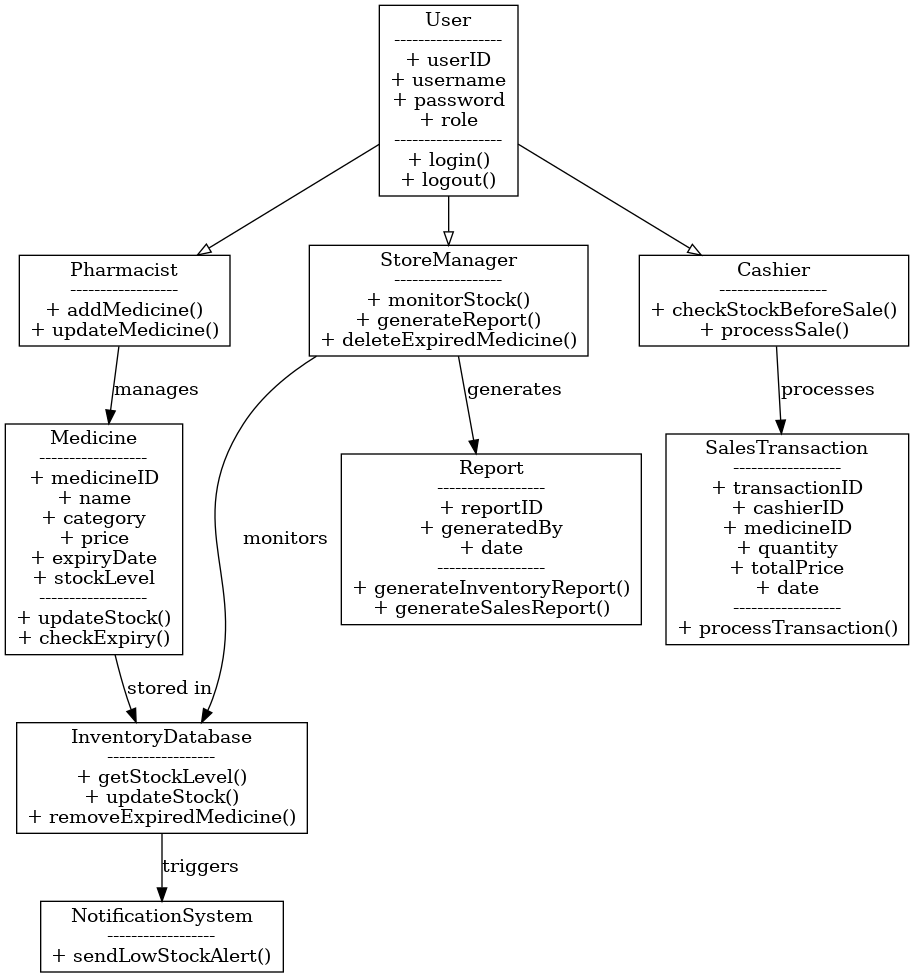
1. Cashier logs into the system.
   * System verifies login credentials.
2. Cashier searches for the requested medicine.
   * Inputs medicine name or scans barcode.
3. System queries the Inventory Database for stock availability.
4. Database returns stock status to the system.
5. System displays stock availability to the cashier.
   * If medicine is available, proceed to sale.
   * If medicine is out of stock, notify the cashier.
6. Cashier proceeds with the sale or informs the customer about unavailability.



# 7. Class Diagram

### **Identified Classes:**

1. User (Superclass)
   * Attributes: userID, username, password, role
   * Methods: login(), logout()
2. Pharmacist (Inherits from User)
   * Methods: addMedicine(), updateMedicine()
3. StoreManager (Inherits from User)
   * Methods: monitorStock(), generateReport(), deleteExpiredMedicine()
4. Cashier (Inherits from User)
   * Methods: checkStockBeforeSale(), processSale()
5. Medicine
   * Attributes: medicineID, name, category, price, expiryDate, stockLevel
   * Methods: updateStock(), checkExpiry()
6. InventoryDatabase
   * Methods: getStockLevel(), updateStock(), removeExpiredMedicine()
7. SalesTransaction
   * Attributes: transactionID, cashierID, medicineID, quantity, totalPrice, date
   * Methods: processTransaction()
8. NotificationSystem
   * Methods: sendLowStockAlert()
9. Report
   * Attributes: reportID, generatedBy, date
   * Methods: generateInventoryReport(), generateSalesReport()



# 8. Conclusion

The Medical Store Inventory System provides essential inventory management functionalities while ensuring security, efficiency, and scalability. The role-based access control ensures that pharmacists, store managers, and cashiers have appropriate permissions to manage medicines, check stock, and prevent sales of expired drugs. The system will be implemented using Python, Flask ,SQLAlchemy (SQLite), HTML, CSS, JavaScript, Bootstrap 5, Chart.j and Flask-Login.

Github link:

https://github.com/saadnadeem554/SE-Project