# 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 MySQL for data storage using Java, Hibernate, and Spring Boot.

## 1.3 Definitions, Acronyms, and Abbreviations

• \*\*SRS\*\*: Software Requirements Specification

• \*\*CRUD\*\*: Create, Read, Update, Delete

• \*\*MySQL\*\*: Structured Query Language for databases

• \*\*Spring Boot\*\*: Java framework for building enterprise applications

# 2. System Functional Requirements

* • Add New Medicine: Pharmacists can add medicines with details (name, category, price, expiry date, stock level).
* • Update Medicine Information: Pharmacists can update existing medicine details.
* • Delete Expired Medicines: Store managers can remove expired medicines.
* • Monitor Stock Levels: Store managers receive notifications for low-stock medicines.
* • Check Stock Before Sale: Cashiers can check the availability of medicines before processing sales.
* • Generate Sales and Inventory Reports: Store managers can generate reports on stock levels and sales trends.
* • Send Low Stock Notifications: System notifies store managers when stock is running low.
* • Role-Based Access Control: The system ensures only authorized personnel can access specific functionalities.

# 3. System Non-Functional Requirements

## 3.1 Product Requirements

• The system should support up to 1000 inventory records.

• The UI must be intuitive and user-friendly for non-technical staff.

## 3.2 Organizational Requirements

• Role-based access control (Pharmacists, Store Managers, Cashiers).

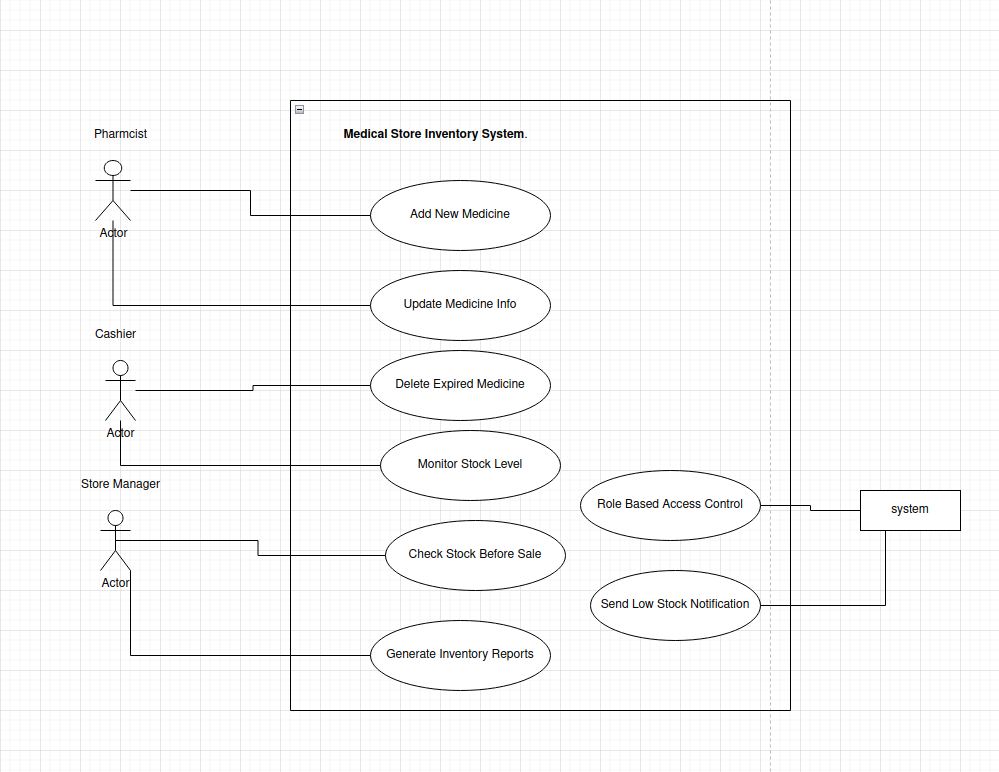
• Compliance with industry regulations for medicine sales.

## 3.3 External Requirements

• Integration with external databases (MySQL).

• The system should be accessible 24/7 with minimal downtime.

# 4. Use Case Diagram



# 5. User Stories

#### **.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 medicine details
* 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
* 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 Generate Inventory Reports**

* Role: Store Manager
* Goal: Generate reports on stock levels and sales
* Reason: Enable better decision-making.

Pre-conditions:

* The store manager must be logged into the system.
* The system must contain recent sales and inventory data.

Post-conditions:

* The generated report provides insights on inventory levels and sales trends.
* The store manager can make data-driven decisions regarding restocking.

#### **5.7 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.

#### **5.8 Implement Role-Based Access Control**

* Role: System Administrator
* Goal: Restrict and grant access based on roles
* Reason: Maintain security.

Pre-conditions:

* The system administrator must be logged into the system.
* Users must be assigned roles within the system.

Post-conditions:

* Users are granted access only to permitted functionalities.
* Unauthorized users are restricted from performing specific actions.

# 6. Sequence Diagrams

(\*Refer to the attached diagrams in the document submission\*)

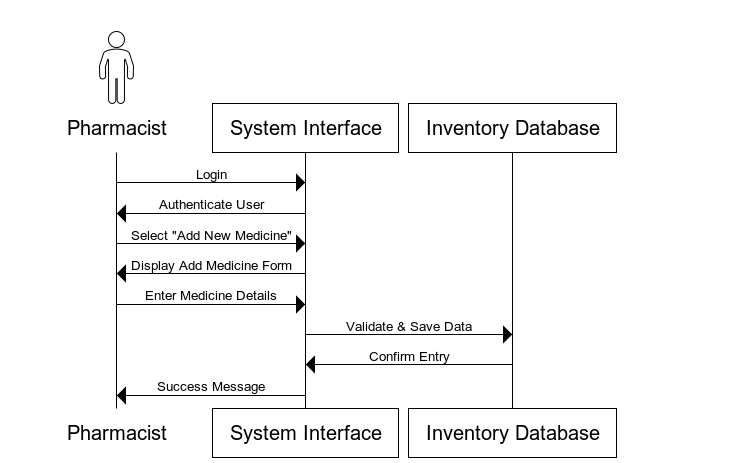
• Adding a New Medicine

### **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.



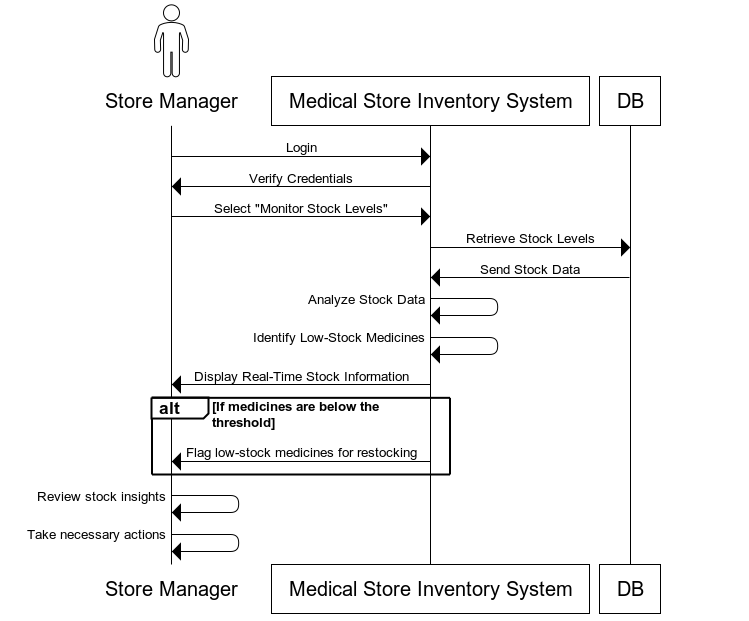
• Monitoring Stock Levels

### **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.



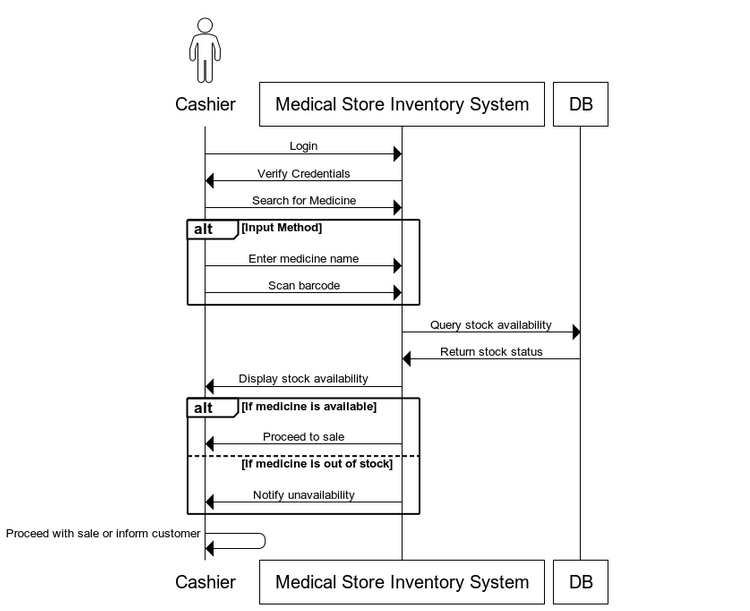
• Checking Stock Before Sale

### **Actors:**

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

### **Flow of Events:**

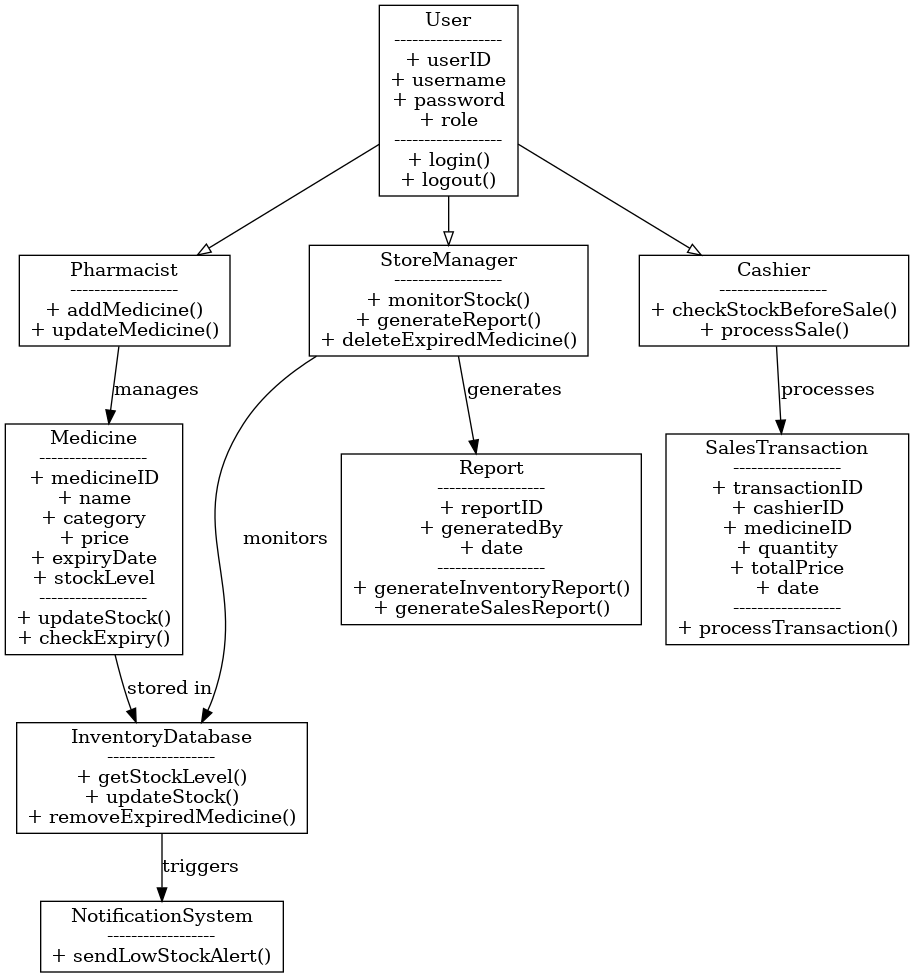
1. Cashier logs into the system.
   * System verifies login credentials.
2. Cashier searches for the requested medicine.
   * Inputs medicine name or scans barcode.
3. System queries the Inventory Database for stock availability.
4. Database returns stock status to the system.
5. System displays stock 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 Java, Hibernate, Spring Boot, and MySQL.