# PharmaBright Pharmacy System Documentation

## Table of Contents

# 1 Introduction

This document outlines the design, implementation, and usage of the PharmaBright Pharmacy System, developed using Apache NetBeans. The system facilitates the management of a pharmacy's medication inventory, allows customers to browse and purchase medications, and ensures real-time updates of stock levels in a MySQL database. The system is designed to be user-friendly, secure, and reliable, catering to both administrators and customers.

# 2 System Requirements

## 2.1 Purpose and Objectives

The primary objective of the PharmaBright system is to streamline the management of pharmacy operations by:

- Providing a secure and intuitive platform for customers to browse and purchase medications.
- Allowing administrators to efficiently manage medication inventory.
- Ensuring real-time updates to stock levels to maintain accurate inventory records.

## 2.2 Functionalities and Features

- **Customer Interface:** Login, browse medications, add to cart, and purchase.
- **-Administrator Interface:** Upload and manage medication details, view sales, and monitor stock levels.
- **Database Integration:** Use of MySQL for data storage and management.
- **Real-Time Stock Updates**: Automatic decrement of stock levels upon customer purchase.
- **Secure Login Mechanism:** Role-based access control for customers and administrators.

## 2.3 Data Types and User Roles

**Data Types:**

- Medication information, customer details, purchase records.

**User Roles:**

- Administrator: Manages medication data, views sales records.
- Customer: Browses medications, makes purchases.

## 2.4 Hardware and Software Requirements

- **Hardware:** Standard computer with internet access.
- **Software:** Java Development Kit (JDK)
- Apache NetBeans IDE
- MySQL Database Server
- JDBC Driver for MySQL

# 3 Database Schema Design

## 3.1 Entities and Relationships

The database schema is designed to manage medication inventory, customer data, and purchase records efficiently. The key entities and their relationships are as follows:

**Medication**: Represents the medication available in the pharmacy.

- **Fields:** id, name, description, price, stock_level, last_updated_date
- **Relationships:** One-to-many with Purchases

**Customer:** Represents the customers who can browse and purchase medications.

- **Fields:** id, username, password, name, contact_details
- **Relationships:** One-to-many with Purchases

**Purchase:** Represents the purchase transactions made by customers.

- **Fields:** id, customer_id, medication_id, quantity, total_price, purchase_date
- **Relationships:** Many-to-one with Customers and Medications

## 3.2 Tables and Fields

**Medications:**

- \`id\` (Primary Key): Unique identifier for each medication.
- \`name\`: Name of the medication.
- \`description\`: Description of the medication.
- \`price\`: Price per unit of the medication.
- \`stock_level\`: Current stock level of the medication.
- \`last_updated_date\`: Date when the medication details were last updated.

**Customers:**

- \`id\` (Primary Key): Unique identifier for each customer.
- \`username\`: Unique username for customer login.
- \`password\`: Encrypted password for customer login.
- \`name\`: Full name of the customer.
- \`contact_details\`: Contact details of the customer.

**Purchases:**

- \`id\` (Primary Key): Unique identifier for each purchase.
- \`customer_id\` (Foreign Key): Identifier for the customer making the purchase.

- `medication_id` (Foreign Key): Identifier for the purchased medication.
- `quantity`: Quantity of medication purchased.
- `total_price`: Total price for the purchase.
- `purchase_date`: Date of the purchase.

### 3.3 Keys and Constraints

**Primary Keys:** Ensure unique identification of records in each table.

- **Foreign Keys:** Establish relationships between tables to maintain data integrity.
- **Constraints:** Ensure valid data entry (e.g., non-negative stock levels, valid dates).

# 4 Application Structure

## 4.1 Organization of Packages and Classes

The application is organized into several packages and classes to ensure modularity and ease of maintenance.

**Entities:** Represents the database entities.

- `Medication`: Class representing the medication entity.
- `Customer`: Class representing the customer entity.
- `Purchase`: Class representing the purchase entity.

**Database Access:** Handles database connectivity and operations.

- `DatabaseConnection`: Class to establish and manage the database connection.
- `MedicationDAO`: Data Access Object (DAO) for medication-related operations.
- `CustomerDAO`: DAO for customer-related operations.
- `PurchaseDAO`: DAO for purchase-related operations.

**Business Logic:** Implements business rules and logic.

- `InventoryManager`: Manages inventory-related operations.
- `PurchaseManager`: Manages purchase-related operations.

**User Interfaces:** Manages graphical user interfaces.

- `LoginUI`: User interface for login.
- `CustomerUI`: User interface for customer interactions.
- `AdminUI`: User interface for administrator interactions.

### 4.2 Coding Style and Naming Conventions

- **Consistency:** Consistent coding style for readability and maintainability.
- **Naming Conventions:** Meaningful names for classes, methods, and variables.
- **Documentation:** Inline comments and documentation for clarity.

## 5 Database Connectivity

### 5.1 Connection Establishment

- **JDBC:** Java Database Connectivity (JDBC) is used to connect to the MySQL database.
- **Configuration:** Database connection settings are configured in the `DatabaseConnection` class.

### 5.2 CRUD Operations

- **Create:** Insert new records into the database.
- **Read:** Retrieve records from the database.
- **Update:** Modify existing records in the database.
- **-Delete:** Remove records from the database.

### 5.3 Exception Handling

- **Robust Handling:** Mechanisms to handle database connectivity issues and other exceptions.
- **Logging:** Log errors and exceptions for troubleshooting.

## 6 User Interfaces

### 6.1 Design and Implementation

- **JavaFX:** Used for designing intuitive and user-friendly interfaces.
- **Forms and Dialogs:** Created for data input and display.
- **Event Handlers:** Manage user interactions and trigger appropriate actions.

### 6.2 Integration with Database

- **Seamless Integration:** User interfaces are integrated with database access methods to ensure data consistency and real-time updates.

## 7 Stock Management

### 7.1 Real-Time Updates

- **Automatic Decrement:** Stock levels decrease automatically in the database when customers purchase medication.
- **Synchronization:** Ensure data consistency between the application and the database.

### 7.2 Administrator Uploads

- **Upload Functionality**: Administrators can upload and update medication information.
- **Validation:** Ensure valid data entry during uploads.

# 8 Testing and Validation

## 8.1 Testing Scenarios

- **Unit Testing:** Test individual components and methods.
- **Integration Testing:** Test the interaction between different components.
- **User Acceptance Testing:** Ensure the system meets user requirements.

## 8.2 Data Validation

- **Input Validation:** Ensure accurate and valid data input.
- **Output Validation:** Ensure correct data retrieval and display.

## 8.3 Bug Identification and Resolution

- **Bug Tracking:** Identify and log bugs for resolution.
- **Continuous Improvement:** Regular updates and bug fixes to improve system reliability.

# 9 User Manual

## 9.1 Installation Process

I. **Install Java Development Kit (JDK):** Ensure JDK is installed on the system.
II. **Set Up MySQL Database:** Install MySQL and set up the database.
III. **Configure Database Connection:** Set database connection settings in the `DatabaseConnection` class.
IV. **Run the Application:** Open the project in Apache NetBeans and run the application.

## 9.2 System Usage

**Login:**

- **Administrator:** Upload and manage medication information.
- **Customer:** Browse and purchase medications.
- **Medication Browsing:** Customers can browse the available medications.

- **Stock Management:** Real-time stock level updates upon purchase.
- **Administrator Functions:** Upload, update, and view medication details and sales records.

# 10 Future Improvements

## 10.1 Enhanced User Interfaces

- **Improved Design:** Enhance the visual appeal and usability of the user interfaces.
- **Mobile Compatibility:** Develop a mobile-friendly version of the application.

## 10.2 Advanced Data Analytics

- **Sales Reports:** Implement detailed sales reports for better business insights.
- **Customer Analytics:** Analyze customer behavior and preferences.

## 10.3 Expanded Functionalities

**Customer Management:** Include more detailed customer management features.

**Reporting Features:** Implement comprehensive reporting features for administrators.