PHARMACEUTICAL INVENTORY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

ARUL AMUDHAN G (241001017)

ASHOK BHARATHI (241001020)

DIVYESH (241001058)

in partial fulfilment for the course

CS23332 – DATABASE MANAGEMENT SYSTEM

for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSMI ENGINEERING COLLEGE

THANDALAM

BONAFIDE CERTIFICATE

Certified that this project report titled **"PHARMACEUTICAL INVENTORY MANAGEMENT SYSTEM"** is the Bonafide work of **"ARUL AMUDHAN G (241001017), ASHOK BHARATHI (241001020), DIVYESH S (241001058)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any another candidate.

SIGNATURE
Dr. Shunmuganathan K.L
Dean of The Department
Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE
Dr.P. Valarmathie
Head of The Department
Department of Information Technology
Rajalakshmi Engineering College

Submitted to Project Viva Voce Examination for the course CS23332 Database Management System held on_____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman **Mr.S. Meganathan,B.E., F.I.E** for his sincere endeavour in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan**, for her enthusiastic motivation which inspired us a lot in completing this project and Vice-Chairman Mr. Abhay Shankar Meganathan B.E., M.S., for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college principal, **Dr.S.N.Murugesan M.E., PhD**., for his kind support and facilities to complete our work on time. We extend heartfelt gratitude to **Dr.P.Valarmathie, Professor and Head of the Department of Information Technology** for her guidance and encouragement throughout the work. We are very glad to thank our course faculty **Dr. Shunmuganathan K.L, Dean** of our department for their encouragement and support towards the successful completion of this project. We extend our thanks to our parents, friends, all faculty members, and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

ARUL AMUDHAN G

ASHOK BHARATHI

DIVYESH S

ABSTRACT

The **Pharmaceutical Inventory Management System (IMS)** is designed as a robust database application to address the complexities of managing pharmaceutical stock, which includes tracking critical details such as expiry dates and manufacturing batches. Developed using a centralized **MySQL** relational database and implemented via a Java-based application, the system replaces error-prone manual methods to ensure accurate, real-time inventory visibility. Key functionalities include the management of core master data (Drugs, Suppliers, Locations, Users) and the automation of transactional processes like creating **Purchase Orders (PO)** and recording stock receipt via **Goods Received Notes (GRN)**. The architecture ensures data integrity through enforced foreign key relationships and provides the foundation for accurate reporting and compliance.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**i**
	LIST OF TABLES	**iii**
	LIST OF FIGURES	**iv**
	LIST OF ABBREVIATION	**v**
1	**INTRODUCTION**	**1**
	1.1 MOTIVATION	1
	1.2 EXISTING SYSTEM	1
	1.3 PROJECT OBJECTIVES	2
	1.4 PROPOSED SYSTEM	2
2	**LITERATURE REVIEW**	**3**
	2.1 OVERVIEW	3
	2.2 PHARMACEUTICAL INVENTORY MANAGEMENT SYSTEM	
	2.3 FEATURES AND FUNCTIONALITIES	3
	2.4 RELEVANCE TO PROJECT	4
3	**SYSTEM DESIGN**	**5**
	3.1 SYSTEM OVERVIEW	5
	3.2 SYSTEM ARCHITECTURE	5
	3.2.1 SYSTEM ARCHITECTURE OVERVIEW	5
	3.2.2 USE CASE DIAGRAM OVERVIEW	6
	3.2.3 KEY USE CASES	6
	3.2.4 RELATIONSHIPS	7
	3.2.5 DATA FLOW DIAGRAM	9
	3.3 SYSTEM REQUIREMENTS	10
	3.4 DATABASE DESIGN	11
4	**PROJECT DESCRIPTION**	21
5	**OUTPUT SCREENSHOTS**	23
6	**CONCLUSION AND FUTURE WORKS**	32
	REFERENCES	33
	APPENDIX	34

LIST OF TABLES

TABLE NO:	TABLE NAME	PAGE NO:
1	User_Master	12
2	Supplier_Master	13
3	Drug_Master	14
4	Location_Master	15
5	Purchase_Order	16
6	Goods_Received_Note	17
7	Stock_Inventory	18

LIST OF FIGURES

Figure No.	Title	Page No.
3.2.1	System Architecture Diagram	6
3.2.2	Use Case Diagram	7
3.2.5	Data Flow Diagram (DFD)	9
3.4.1	Database Design & Entity-Relationship (ER) Diagram	20

LIST OF ABBREVIATIONS

Abbreviation	Full Form
IMS	Inventory Management System
DBMS	Database Management System
PO	Purchase Order
GRN	Goods Received Note
GUI	Graphical User Interface
JDBC	Java Database Connectivity
SQL	Structured Query Language
OTC	Over The Counter
GSTIN	Goods and Services Tax Identification Number

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

In the pharmaceutical sector, managing inventory is crucial due to the short shelf life of products and strict regulatory requirements (e.g., Schedule H drugs, specific storage conditions). Manual or obsolete systems risk stock-outs, expiry of expensive drugs, inaccurate counts, and non-compliance. The primary motivation for this project is to develop an automated, efficient, and robust solution to track every unit of stock at the **batch level**, ensuring product integrity and enabling immediate recall and expiry checks.

1.2 EXISTING SYSTEM

The existing system is conceptualized as a legacy or manual record-keeping method (e.g., paper ledgers, spreadsheets). This system suffers from inherent drawbacks such as delayed data entry, high chances of human error in calculations, poor visibility into inventory levels across different locations, and reactive rather than proactive management of crucial parameters like reorder levels and impending expiry dates.

1.3 PROJECT OBJECTIVES

The main objectives of the Pharmaceutical IMS project are:

To establish a well-structured **Relational Database Schema** in MySQL capable of managing all pharmaceutical logistics data.

To accurately maintain **Master Records** for all Drugs, Suppliers, Locations, and Users.

To implement transactional modules for **Purchase Order (PO)** creation and maintenance of **Goods Received Notes (GRN)**.

To ensure **batch-level tracking** and capture essential product data like batch_number and expiry_date at the point of receipt (GRN).

To maintain a **Stock Inventory** that accurately reflects the quantity of each specific batch at a designated physical location.

To provide an intuitive, role-based user interface for managing operations (Inferred from role field in User_Master and presence of GUI classes).

1.4 PROPOSED SYSTEM

The Proposed System is a **Pharmaceutical Inventory Management System** built on a client-server architecture. It utilizes a Java front-end application (GUI) and a MySQL database backend. The system centralizes all inventory data, applying business rules at the database level (e.g., FOREIGN KEY constraints) and the application level (e.g., stock update logic within Java services). It provides role-based access for different user types like 'Admin', 'Pharmacist', and 'Staff'. The core process flow links the procurement cycle (PO) to the receiving cycle (GRN), which finally updates the physical inventory (Stock_Inventory).

CHAPTER 2

LITERATURE REVIEW

2.1 OVERVIEW

Inventory management is a crucial function in logistics and supply chain systems. Literature consistently emphasizes the shift from manual to computerized systems for improved efficiency, cost reduction, and enhanced decision-making capabilities. Modern DBMS solutions are preferred for their ability to enforce data integrity, manage concurrent user access, and handle complex relational data models.

2.2 PHARMACEUTICAL INVENTORY MANAGEMENT SYSTEM

Inventory control in the pharmaceutical industry is distinctly more critical than in general retail due to:

1. **Strict Expiry Dates:** The system must track inventory at the batch level (batch_number, expiry_date in GRN_Item and Stock_Inventory) to minimize financial loss from expired goods and ensure patient safety.

2. **Regulatory Compliance:** The system must track drug categories (like 'Schedule H') and specific **storage conditions** (Drug_Master table) to adhere to pharmaceutical laws.

3. **Traceability:** Complete audit trails are required, linking received stock (GRN) back to the original Purchase Order (PO) and the Supplier for easy recall management.

2.3 FEATURES AND FUNCTIONALITIES

Based on the implemented structure, the core functionalities reviewed and utilized are:

Master Data Maintenance: Centralized records for Suppliers (Supplier_Master), Drugs (Drug_Master), and Storage Locations (Location_Master).

Procurement Cycle: Structured data flow from Purchase_Order creation to Goods_Received_Note generation.

Batch Tracking: The use of the GRN_Item and Stock_Inventory tables ensures that physical inventory reflects stock accurately by material_code, location_code, and unique batch_number.

Control Mechanisms: Setting of **reorder_level** on the Drug_Master enables automated stock monitoring and alerts.

2.4 RELEVANCE TO PROJECT

The existing literature confirms that a robust relational database model is the standard for complex inventory needs, especially when multi-level tracking (drug -> batch -> location) is required. The current project directly implements these best practices by employing an RDBMS schema (MySQL) to model master and transactional entities, enforcing referential integrity and ensuring the storage of critical batch metadata essential for pharmaceutical operations.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM OVERVIEW

The Pharmaceutical IMS operates on a **two-tier architecture**. The system is composed of the Client (Presentation Layer and some Business Logic) and the Database Server (Data Layer). The system is centralized, meaning all clients connect to a single database instance via **JDBC** for all data operations, while the logic is handled by Java classes in the Application layer.

3.2 SYSTEM ARCHITECTURE

3.2.1 SYSTEM ARCHITECTURE OVERVIEW

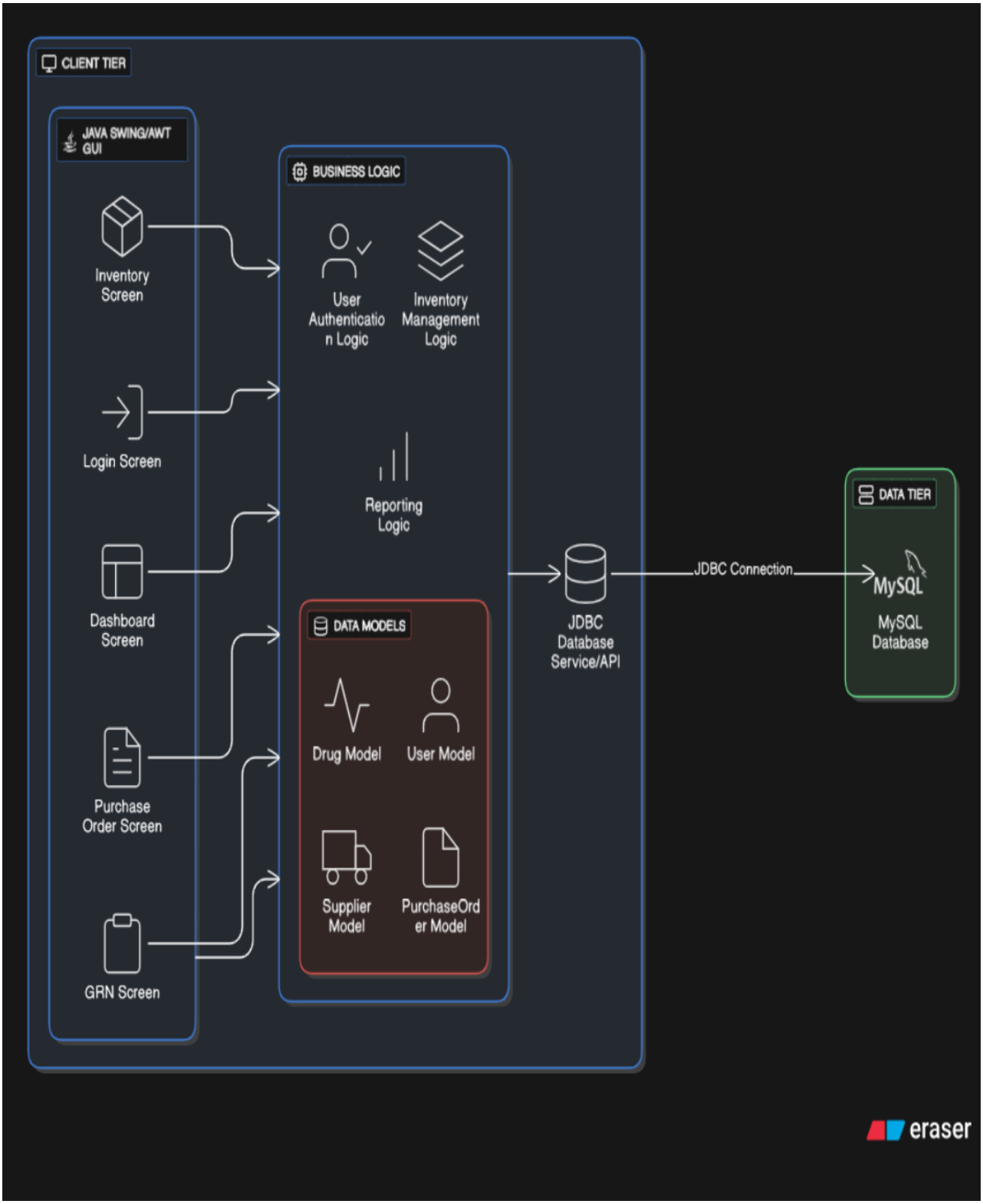
The architecture is layered as follows:

Data Layer (Backend): Consists of the **MySQL Database** (pharma_ims) and the JDBC connector library. This layer handles data storage, retrieval, and persistence, governed by the schema defined in db_setup.sql.

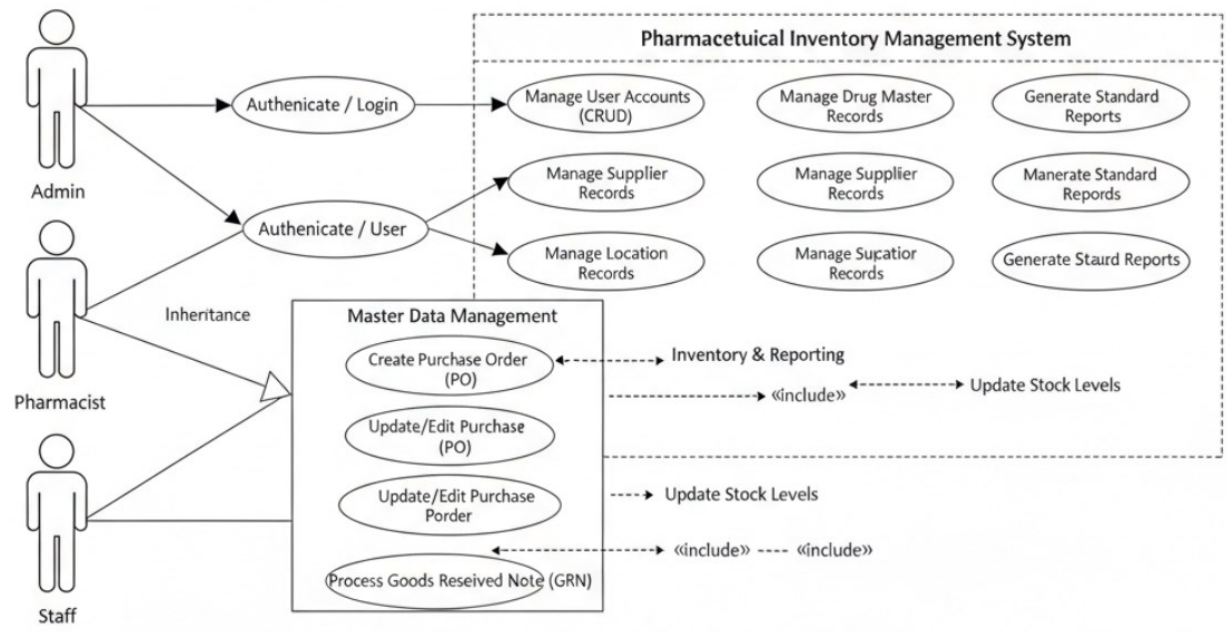
Business/Logic Layer: Implemented in Java classes (e.g., DatabaseService.java, AuthService.java). This layer contains the core application logic, such as validating user input, performing multi-step operations (e.g., creating a GRN and updating stock), and enforcing security rules.

Presentation Layer (Client): Consists of the Java **GUI** components (e.g., LoginGUI.java, DashboardPanel.java, PurchaseOrderPanel.java). This layer provides the user interface for interaction and displays the processed data fetched from the d

3.2.1 System Architecture Diagram:



3.2.2 USE CASE DIAGRAM OVERVIEW



The main actors in the system are **Admin**, **Pharmacist**, and **Staff**. The core processes revolve around: User Authentication; Master Data Management (for Drugs, Suppliers, Locations); Procurement Workflow (PO and GRN creation); and Inventory Query/Reporting.

3.2.3 KEY USE CASES

1. **User Authentication:** Any user (Admin, Pharmacist, Staff) attempts to log in using credentials verified against the User_Master table.

2. **Drug/Master Data Management (Admin/Pharmacist):** Create, Retrieve, Update, or Delete (CRUD) records in Drug_Master, Supplier_Master, and Location_Master.

3. **New Purchase Order (Staff/Pharmacist):** Create an entry in Purchase_Order and associated records in PurchaseOrder_Item, linking to a specific Supplier_Master.

4. **Goods Receipt (Staff/Pharmacist):** Create a Goods_Received_Note linked to a Purchase_Order, capturing detailed batch information in GRN_Item, and subsequently updating the Stock_Inventory table.

5. **Inventory Check (All Users):** View real-time stock levels, batches, expiry dates, and locations from the Stock_Inventory table.

3.2.4 RELATIONSHIPS

The database employs a highly normalized relational structure with explicit foreign keys:

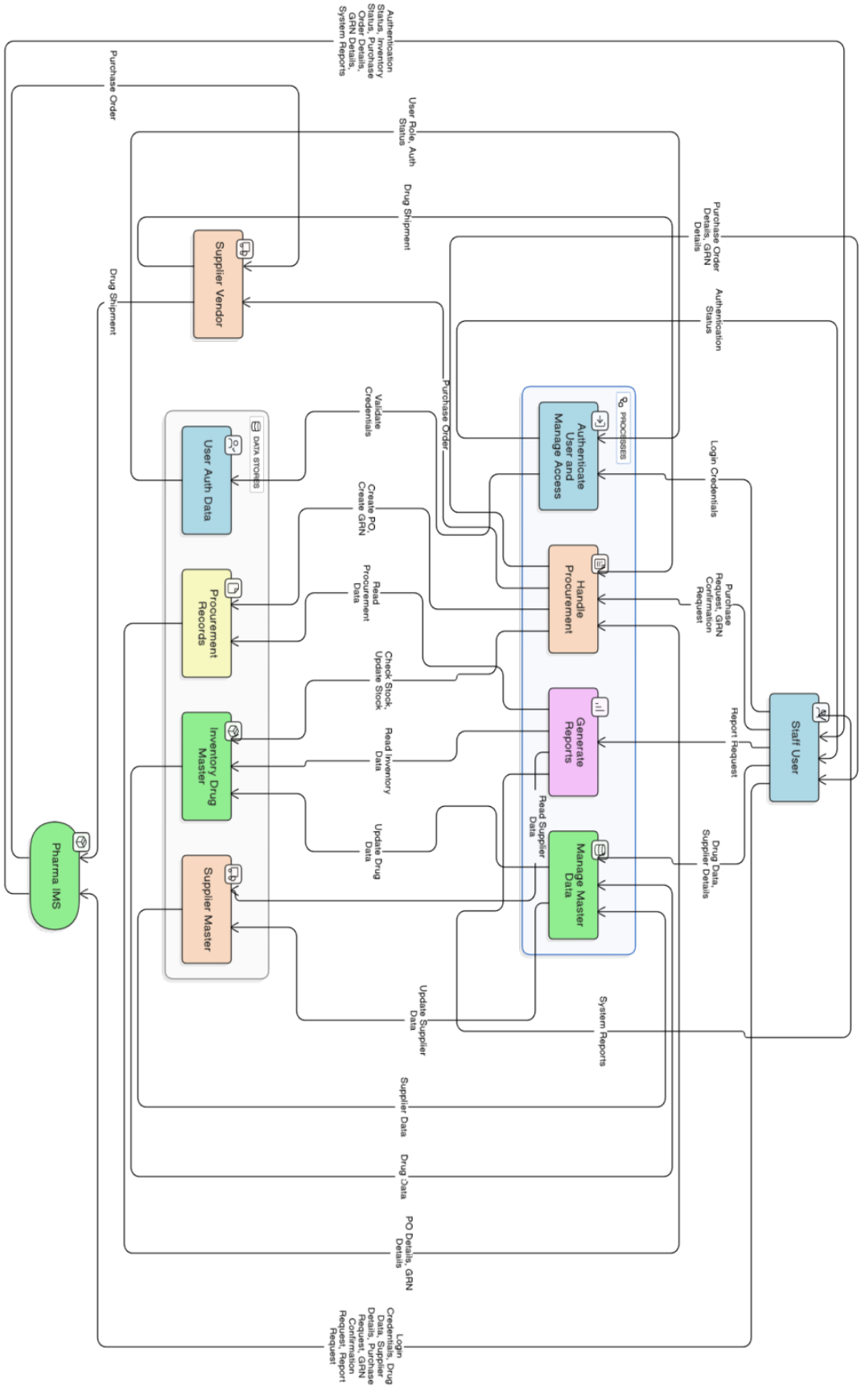
One-to-Many Relationships:

- Supplier_Master (1) → Drug_Master (N) (Preferred Supplier).
- Supplier_Master (1) → Purchase_Order (N).
- Purchase_Order (1) →PurchaseOrder_Item (N) → GRN_Item (N) → Goods_Received_Note (N).

Composite Key/Junction-like Relationship (Many-to-Many resolution):

- The Stock_Inventory table resolves the N:M relationship between Drug_Master and Location_Master (i.e., many drugs can be in many locations, and one location can hold many drugs) by using a composite unique key: (material_code, location_code, batch_number).

3.2.5Data flow diagram (DfD):



3.3. SYSTEM REQUIREMENTS

The Pharmaceutical Inventory Management System (pharma-ims) operates on a client-server architecture, requiring specific hardware and software components to function effectively.

1. Backend & Application Server

Operating System: Windows Server, Linux, or macOS.

Database Management System: MySQL (or a compatible MariaDB/Percona Server) for managing all master and transactional data (Drugs, Suppliers, Purchase Orders, GRNs, and Stock Inventory). The database name is explicitly pharma_ims.

Programming Language: Java for the entire backend application logic (Business and Data Access layers).

Database Connectivity: JDBC (Java Database Connectivity) for establishing a seamless connection between the Java application and the MySQL database.

Application Hosting/Runtime: A **Java Runtime Environment (JRE)** version 8 or later is required to run the Java-based application/backend services.

2. Frontend (Client Application)

Client Type: Desktop Application. The front-end user interface is implemented as a standalone executable client, not a web application, based on the presence of GUI-specific Java classes (...GUI.java, ...Panel.java, ...Dialog.java).

GUI Libraries: Java Swing/AWT (Standard Java GUI libraries) are used to render the interface panels and dialogs (e.g., LoginGUI, InventoryGUI, CreatePurchaseOrderDialog).

Client Runtime: A **Java Runtime Environment (JRE)** must be installed on the client machine to launch and operate the GUI application.

3. Development Tools

IDE: An integrated Development Environment (IDE) that supports Java and SQL development, such as **IntelliJ IDEA**, **Eclipse**, or **Visual Studio Code (with Java extensions)**.

Version Control System: Git for source code control and collaborative development.

Database Tool: A utility like MySQL Workbench or DBeaver for initial database creation, schema setup, and data insertion/testing (e.g., executing the db_setup.sql script).

3.4. DATABASE DESIGN

The database consists of two primary tables:

1. User_Master table is used to store user credentials for logging into the system.

2. Supplier_Master: This table stores comprehensive master data for all pharmaceutical suppliers/distributors.

3. Drug_Master: This master table maintains the core catalog of all individual drug products stocked.

4. Location_Master: This table defines all possible physical locations within the facility where stock can be stored.

5. Purchase_Order: This table serves as the header record for every commitment to purchase drugs from a supplier.

6. Goods_Received_Note: This table documents the administrative confirmation that an order (or part of an order) has been physically received.

7. Stock_Inventory: This is the core table representing the **current, physical, on-hand stock**.

Table: **User_Master**

Purpose:

The User_Master table is used to store user credentials for logging into the system. It includes each user's unique ID, their hashed password, and their **role** (e.g., 'Admin', 'Pharmacist', 'Staff') for granting appropriate access permissions.

Table Schema:

Table 3.1. User_Master Schema

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	auto_increment
username	varchar(100)	NO	UNI	NULL	
password_hash	varchar(255)	NO		NULL	
full_name	varchar(255)	YES		NULL	
role	varchar(50)	YES		NULL	
is_active	tinyint(1)	YES		1	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Constraints:

Primary Key: The user_id column is defined as the Primary Key to ensure uniqueness across the user records.

Unique Key: The username column is defined as **UNIQUE**, enforcing that no two users can share the same login name.

NOT NULL: The username and password_hash columns are defined with the **NOT NULL** constraint.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS User_Master (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(100) UNIQUE NOT NULL,
   password_hash VARCHAR(255) NOT NULL,
   full_name VARCHAR(255),
   role VARCHAR(50),
   is_active BOOLEAN DEFAULT TRUE,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Table: Supplier_Master

Purpose:

This table stores comprehensive master data for all pharmaceutical suppliers/distributors. It is crucial for procurement and includes contact details, payment terms, and regulatory identifiers like the gstin and drug_license_number.

Table Schema:

Table 3.2. Supplier_Master Schema

supplier_id	int	NO	PRI	NULL	auto_increment
supplier_name	varchar(255)	NO		NULL	
contact_person	varchar(255)	YES		NULL	
address	text	YES		NULL	
email	varchar(255)	YES		NULL	
phone_number	varchar(20)	YES		NULL	
gstin	varchar(50)	YES		NULL	
drug_license_number	varchar(100)	YES		NULL	
payment_terms	varchar(255)	YES		NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Constraints:

Primary Key: The supplier_id column is defined as the Primary Key.

NOT NULL: The supplier_name column cannot be empty.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Supplier_Master (
   supplier_id INT AUTO_INCREMENT PRIMARY KEY,
   supplier_name VARCHAR(255) NOT NULL,
   contact_person VARCHAR(255),
   address TEXT,
   email VARCHAR(255),
   phone_number VARCHAR(20),
   gstin VARCHAR(50),
   drug_license_number VARCHAR(100),
   payment_terms VARCHAR(255),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP);
```

Table : **Drug_Master**

Purpose:

This master table maintains the core catalog of all individual drug products stocked. It captures critical fields like brand_name, generic_name, strength, required storage_conditions, and the minimum reorder_level for inventory management.

Table Schema:

Table 3.2. Drug_Master Schema

Field	Type	Null	Key	Default	Extra
material_code	varchar(50)	NO	PRI	NULL	
brand_name	varchar(255)	NO		NULL	
generic_name	varchar(255)	YES		NULL	
manufacturer	varchar(255)	YES		NULL	
formulation	varchar(100)	YES		NULL	
strength	varchar(100)	YES		NULL	
schedule_category	varchar(50)	YES		NULL	
storage_conditions	text	YES		NULL	
reorder_level	int	YES		0	
is_active	tinyint(1)	YES		1	
preferred_supplie...	int	YES	MUL	NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Constraints:

Primary Key: The material_code serves as the unique identifier for each drug product.

Foreign Key: The optional preferred_supplier_id links to the Supplier_Master table. It uses **ON DELETE SET NULL** to ensure the drug record is kept if the supplier is deleted.

NOT NULL: The brand_name is mandatory.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Drug_Master (
   material_code VARCHAR(50) PRIMARY KEY,
   brand_name VARCHAR(255) NOT NULL,
   generic_name VARCHAR(255),
   manufacturer VARCHAR(255),
   formulation VARCHAR(100),
   strength VARCHAR(100),
   schedule_category VARCHAR(50),
```

```
    storage_conditions TEXT,
    reorder_level INT DEFAULT 0,
    is_active BOOLEAN DEFAULT TRUE,

    preferred_supplier_id INT,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    -- Define the Foreign Key constraint for preferred supplier
    FOREIGN KEY (preferred_supplier_id) REFERENCES Supplier_Master(supplier_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

Table : Location_Master

Purpose:

This table defines all possible physical locations within the facility where stock can be stored. This allows for accurate tracking of where every drug batch is physically located, aiding retrieval and inventory audits.

Table Schema:

Table 3.2. Location_Master Schema

Field	Type	Null	Key	Default	Extra
location_code	varchar(50)	NO	PRI	NULL	
location_name	varchar(255)	NO		NULL	
description	text	YES		NULL	
capacity	int	YES		0	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Constraints:

Primary Key: The location_code acts as the unique identifier for each storage area.

NOT NULL: The location_name is mandatory.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Location_Master (
   location_code VARCHAR(50) PRIMARY KEY,
   location_name VARCHAR(255) NOT NULL,
   description TEXT,
   capacity INT DEFAULT 0,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Table : Purchase_Order

Purpose:

This table serves as the header record for every commitment to purchase drugs from a supplier. It captures order-level information, such as the order_date, expected delivery date, and the overall status ('Pending', 'Received').

Table Schema:

Table 3.2. Purchase_Order Schema

Field	Type	Null	Key	Default	Extra
po_id	int	NO	PRI	NULL	auto_increment
supplier_id	int	NO	MUL	NULL	
order_date	date	NO		NULL	
expected_date	date	YES		NULL	
total_amount	decimal(10,2)	YES		0.00	
status	varchar(50)	YES		Pending	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Constraints:

Primary Key: The po_id is the unique identifier for the purchase order.
Foreign Key: The supplier_id links to Supplier_Master(supplier_id). Action: ON DELETE RESTRICT prevents deletion of a supplier if open orders still exist.
NOT NULL: The supplier_id and order_date fields are mandatory.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Purchase_Order (
    po_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_id INT NOT NULL,
    order_date DATE NOT NULL,
    expected_date DATE,
    total_amount DECIMAL(10, 2) DEFAULT 0.00,
    status VARCHAR(50) DEFAULT 'Pending', -- Pending, Shipped, Received
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (supplier_id) REFERENCES Supplier_Master(supplier_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Table: **Goods_Received_Note**

Purpose:

This table documents the administrative confirmation that an order (or part of an order) has been physically received. It links the receipt directly to the original Purchase_Order and records who received the goods and when.

Table Schema:

Table 3.1. Goods_Received_Note Schema

Field	Type	Null	Key	Default	Extra
grn_id	int	NO	PRI	NULL	auto_increment
po_id	int	NO	MUL	NULL	
received_date	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
received_by	varchar(255)	YES		NULL	
status	varchar(50)	YES		Verified	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Constraints:

Primary Key: The `grn_id` is the unique identifier for the goods receipt transaction.

Foreign Key: The `po_id` links to `Purchase_Order(po_id)`. **Action:** `ON DELETE RESTRICT` prevents the deletion of a PO if goods have already been received against it.

NOT NULL: The `po_id` field is mandatory.

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Goods_Received_Note (
   grn_id INT AUTO_INCREMENT PRIMARY KEY,
   po_id INT NOT NULL,
   received_date DATETIME DEFAULT CURRENT_TIMESTAMP,
   received_by VARCHAR(255),
   status VARCHAR(50) DEFAULT 'Verified',
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (po_id) REFERENCES Purchase_Order(po_id)
      ON DELETE RESTRICT
      ON UPDATE CASCADE
);
```

Table: **Stock_Inventory**

Purpose:

This is the core table representing the **current, physical, on-hand stock**. Each record tracks the quantity of a specific material_code (drug), identified by a unique batch_number, located at a specific location_code. This granularity is vital for managing expiry dates and product recalls.

Table Schema:

Table 3.1. Stock_Inventory Schema

Field	Type	Null	Key	Default	Extra
stock_id	int	NO	PRI	NULL	auto_increment
material_code	varchar(50)	NO	MUL	NULL	
location_code	varchar(50)	NO	MUL	NULL	
batch_number	varchar(100)	NO		NULL	
quantity	int	NO		0	
unit_cost	decimal(10,2)	YES		NULL	
mfg_date	date	YES		NULL	
exp_date	date	YES		NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

Constraints:

Primary Key: stock_id is the row identifier.

Unique Key: The composite key (material_code, location_code, batch_number) ensures

that a specific production batch of a drug is tracked only once per physical location, preventing redundant records and maintaining data integrity.

Foreign Key: material_code links to Drug_Master(material_code). Action: ON DELETE RESTRICT prevents deletion of a drug record if stock exists.

Foreign Key: location_code links to Location_Master(location_code). Action: ON DELETE RESTRICT prevents deletion of a location while it holds stock.

NOT NULL: material_code, location_code, batch_number, and quantity are mandatory fields.

SQL Definition:

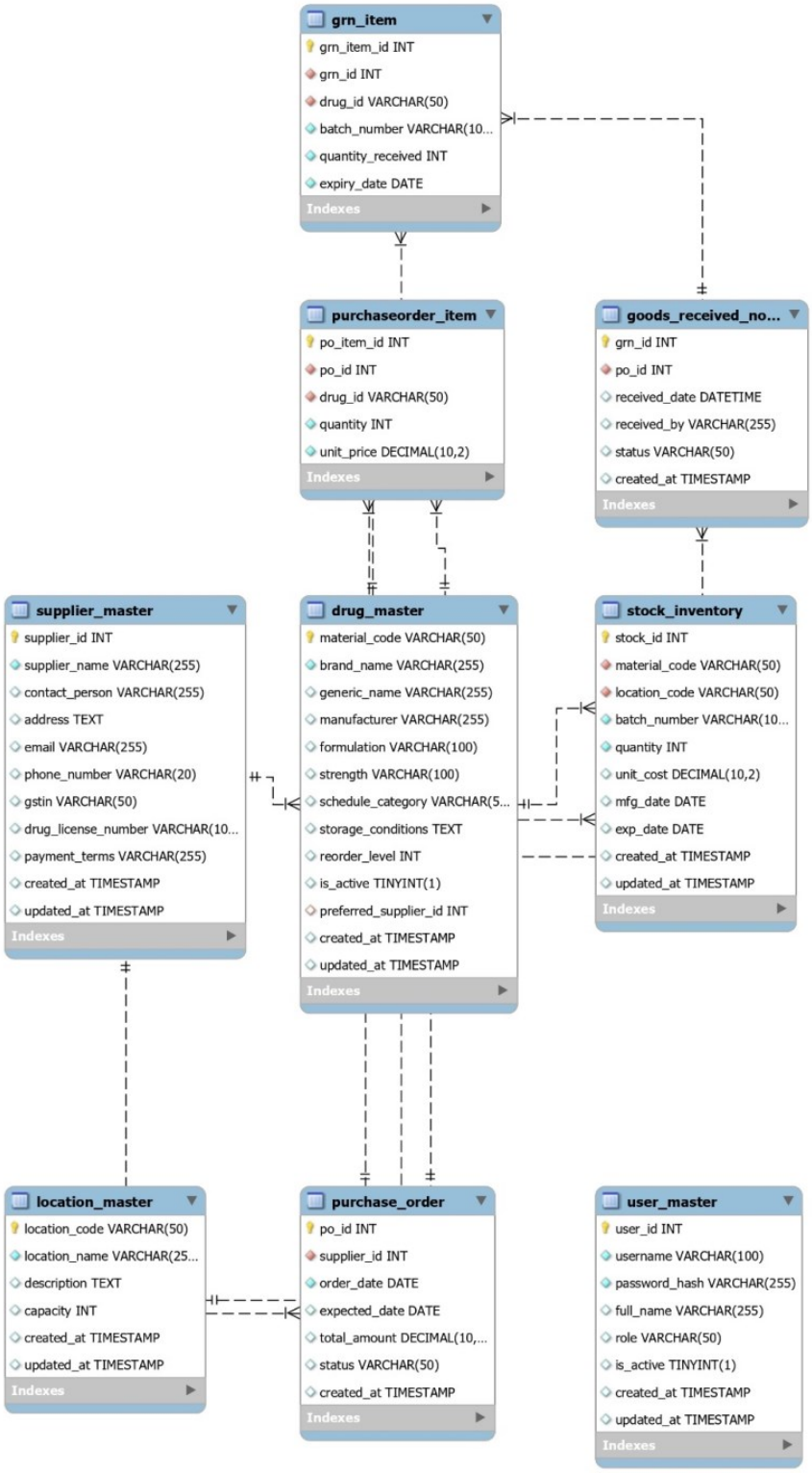
```
CREATE TABLE IF NOT EXISTS Stock_Inventory (
    stock_id INT AUTO_INCREMENT PRIMARY KEY,
    material_code VARCHAR(50) NOT NULL,
    location_code VARCHAR(50) NOT NULL,
    batch_number VARCHAR(100) NOT NULL,
    quantity INT NOT NULL DEFAULT 0,
    unit_cost DECIMAL(10, 2),
    mfg_date DATE,
    exp_date DATE,

    -- Ensures a specific batch of a drug is tracked only once per location
    UNIQUE KEY (material_code, location_code, batch_number),

    FOREIGN KEY (material_code) REFERENCES Drug_Master(material_code)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (location_code) REFERENCES Location_Master(location_code)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

3.4.1 Entity-Relationship (ER) Diagram:



CHAPTER 4

PROJECT DESCRIPTION

The Pharmaceutical Inventory Management System (pharma-ims) is a database-backed application designed to manage the entire lifecycle of pharmaceutical products, with a core focus on granular, batch-level inventory control. The system is built around three main functional modules: Master Data Management, Procurement Workflow, and Inventory Control.

4.1 Master Data Management

This module handles the static, reference data essential for daily operations.

Drug Master (Drug_Master): This is the core catalog, identifying each drug by a unique material_code. It tracks critical information necessary for compliance and purchasing, including the brand_name, generic_name, manufacturer, strength, and regulatory data like the schedule_category ('Schedule H', 'OTC'). It also stores specific storage_conditions and defines a reorder_level for replenishment planning. It maintains a link to a preferred_supplier_id.

Supplier Master (Supplier_Master): Stores comprehensive details for all drug vendors, including supplier_name, contact information, and mandatory compliance identifiers like gstin and drug_license_number.

Location Master (Location_Master): Defines and tracks all physical storage areas (e.g., "Cold Storage B2," "Warehouse Zone A1") using a unique location_code and storing its storage capacity.

User Master (User_Master): Manages user access, utilizing a username, password_hash, and assigning a hierarchical role ('Admin', 'Pharmacist', 'Staff') for permission control.

4.2 Procurement Workflow (PO & GRN)

This module automates the process from ordering stock to receiving it into the warehouse.

Purchase Order (PO):

- The purchase process begins with the Purchase_Order header, recording basic order details like supplier_id, order_date, expected_date, and overall status ('Pending', 'Shipped', 'Received').
- Details of the requested drugs are stored in the PurchaseOrder_Item table, linking back to the PO header via po_id and specifying the drug_id, quantity, and unit_price.

Goods Received Note (GRN):

- The Goods_Received_Note documents the physical receipt of goods against a PO. It links to the relevant po_id and records the received_date and the received_by user.
- The crucial component is the GRN_Item table, which captures lot-specific information for the inventory, recording the unique **batch_number** and the **expiry_date** of the received drugs.

4.3 Inventory Control

This module holds the definitive truth regarding on-hand stock and physical placement.

Stock Inventory (Stock_Inventory): This is the central transactional table for current inventory levels.

Key Tracking Dimensions: To ensure precision, stock is tracked at the most granular level, enforced by a **unique key** defined by the combination of material_code, location_code, and batch_number.

Inventory Update: The stock quantity is updated in this table based on the successful processing of a GRN_Item record. It permanently stores financial (like unit_cost) and quality control data (like mfg_date and exp_date).

This design prioritizes **lot/batch traceability** and **expiry date visibility**, which are non-negotiable requirements for a compliant pharmaceutical inventory system.

CHAPTER 5

OUTPUT SCREEN SHOTS

LOGIN/SIGN UP PAGE:

Serves as the entry point for users, allowing them to either log in with existing credentials or navigate to the sign-up page to create a new account.

Figure 5.1. Login/Sign Up

LOGIN PAGE:

Users enter their username and password to access the system. Invalid attempts prompt error messages for correction.

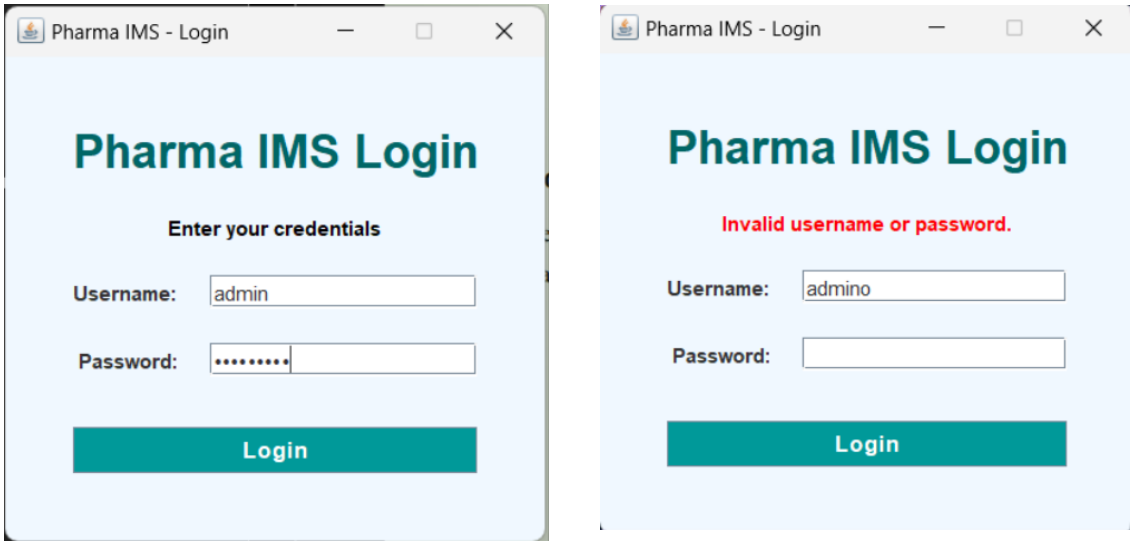
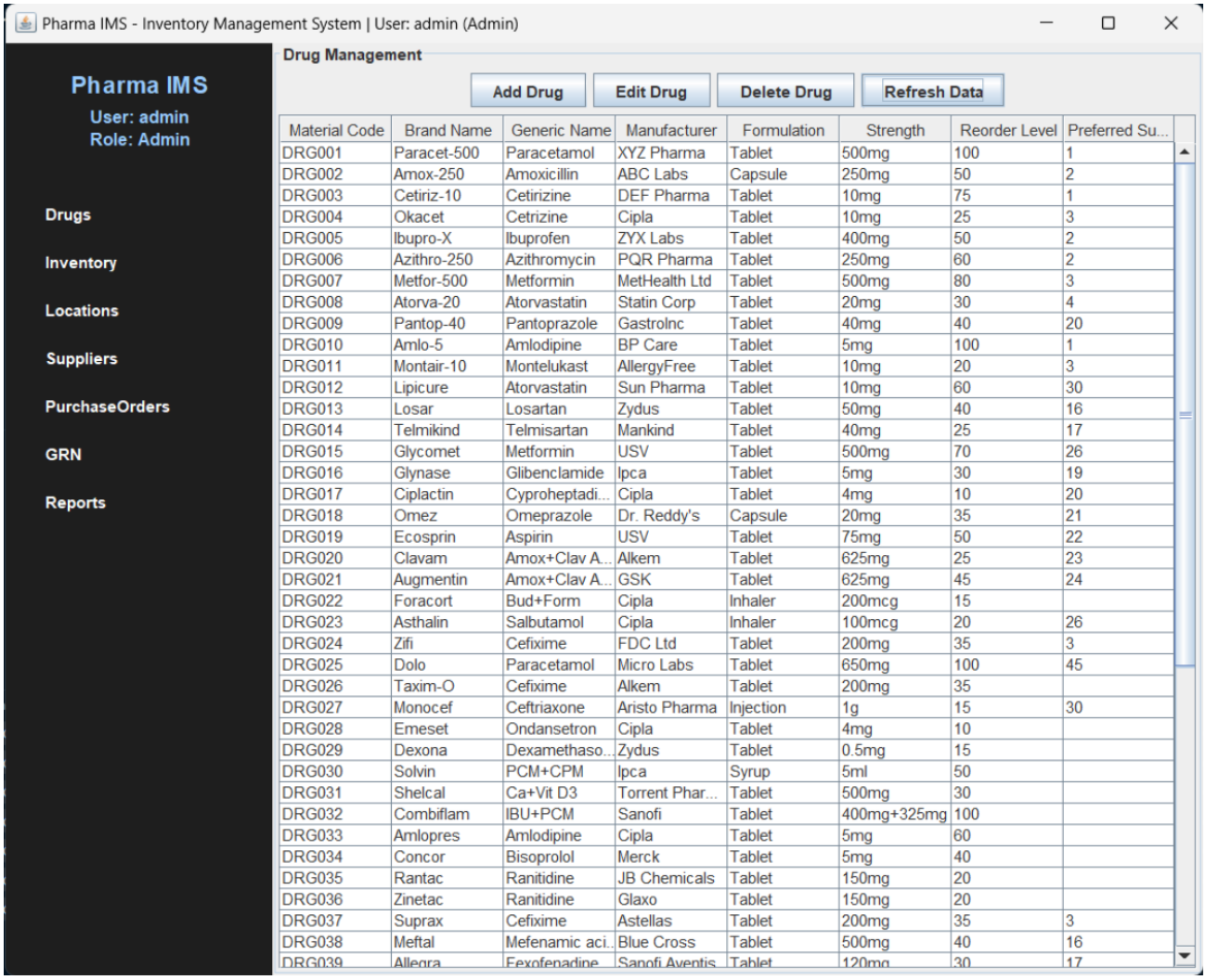


Figure 5.2. Login Figure

DRUG MASTER:

Displays the drugs details such as drug id, brand name, generic name, manufacturer, formulation, strength, reorder level and preferred supplier id, etc.



Material Code	Brand Name	Generic Name	Manufacturer	Formulation	Strength	Reorder Level	Preferred Su...
DRG001	Paracet-500	Paracetamol	XYZ Pharma	Tablet	500mg	100	1
DRG002	Amox-250	Amoxicillin	ABC Labs	Capsule	250mg	50	2
DRG003	Cetiriz-10	Cetirizine	DEF Pharma	Tablet	10mg	75	1
DRG004	Okacet	Cetrizine	Cipla	Tablet	10mg	25	3
DRG005	Ibupro-X	Ibuprofen	ZYX Labs	Tablet	400mg	50	2
DRG006	Azithro-250	Azithromycin	PQR Pharma	Tablet	250mg	60	2
DRG007	Metfor-500	Metformin	MetHealth Ltd	Tablet	500mg	80	3
DRG008	Atorva-20	Atorvastatin	Statin Corp	Tablet	20mg	30	4
DRG009	Pantop-40	Pantoprazole	GastroInc	Tablet	40mg	40	20
DRG010	Amlo-5	Amlodipine	BP Care	Tablet	5mg	100	1
DRG011	Montair-10	Montelukast	AllergyFree	Tablet	10mg	20	3
DRG012	Lipicure	Atorvastatin	Sun Pharma	Tablet	10mg	60	30
DRG013	Losar	Losartan	Zydus	Tablet	50mg	40	16
DRG014	Telmikind	Telmisartan	Mankind	Tablet	40mg	25	17
DRG015	Glycomet	Metformin	USV	Tablet	500mg	70	26
DRG016	Glynase	Glibenclamide	Ipca	Tablet	5mg	30	19
DRG017	Ciplactin	Cyproheptadi...	Cipla	Tablet	4mg	10	20
DRG018	Omez	Omeprazole	Dr. Reddy's	Capsule	20mg	35	21
DRG019	Ecosprin	Aspirin	USV	Tablet	75mg	50	22
DRG020	Clavam	Amox+Clav A...	Alkem	Tablet	625mg	25	23
DRG021	Augmentin	Amox+Clav A...	GSK	Tablet	625mg	45	24
DRG022	Foracort	Bud+Form	Cipla	Inhaler	200mcg	15	
DRG023	Asthalin	Salbutamol	Cipla	Inhaler	100mcg	20	26
DRG024	Zifi	Cefixime	FDC Ltd	Tablet	200mg	35	3
DRG025	Dolo	Paracetamol	Micro Labs	Tablet	650mg	100	45
DRG026	Taxim-O	Cefixime	Alkem	Tablet	200mg	35	
DRG027	Monocef	Ceftriaxone	Aristo Pharma	Injection	1g	15	30
DRG028	Emeset	Ondansetron	Cipla	Tablet	4mg	10	
DRG029	Dexona	Dexamethaso...	Zydus	Tablet	0.5mg	15	
DRG030	Solvin	PCM+CPM	Ipca	Syrup	5ml	50	
DRG031	Shelcal	Ca+Vit D3	Torrent Phar...	Tablet	500mg	30	
DRG032	Combiflam	IBU+PCM	Sanofi	Tablet	400mg+325mg	100	
DRG033	Amlopres	Amlodipine	Cipla	Tablet	5mg	60	
DRG034	Concor	Bisoprolol	Merck	Tablet	5mg	40	
DRG035	Rantac	Ranitidine	JB Chemicals	Tablet	150mg	20	
DRG036	Zinetac	Ranitidine	Glaxo	Tablet	150mg	20	
DRG037	Suprax	Cefixime	Astellas	Tablet	200mg	35	3
DRG038	Meftal	Mefenamic aci...	Blue Cross	Tablet	500mg	40	16
DRG039	Allegra	Fexofenadine	Sanofi Aventis	Tablet	120mg	30	17

Figure 5.3. Drug Panel on display

CRUD OPERATION IN DRUG MASTER:

Performs Create, Update, Delete operations on Drug_Master table.

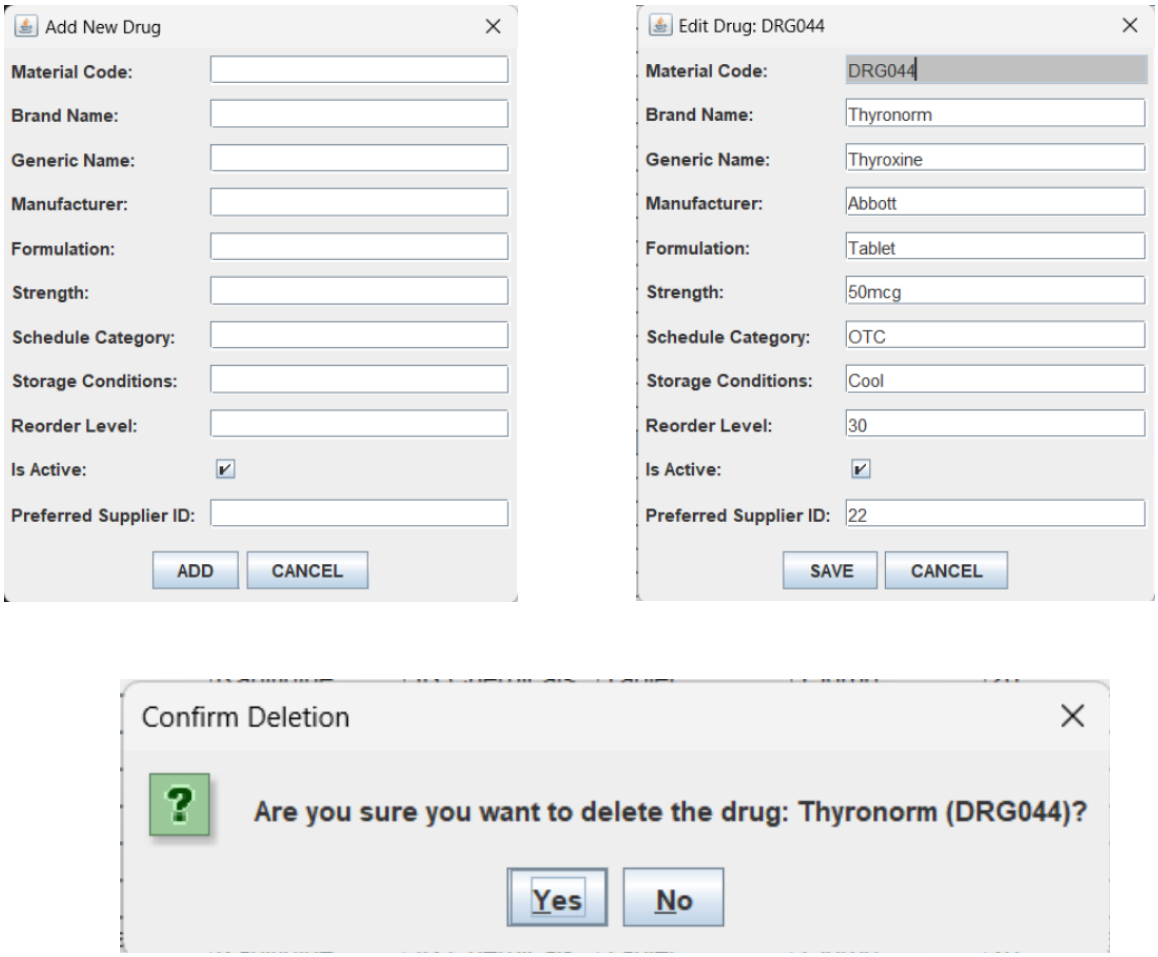
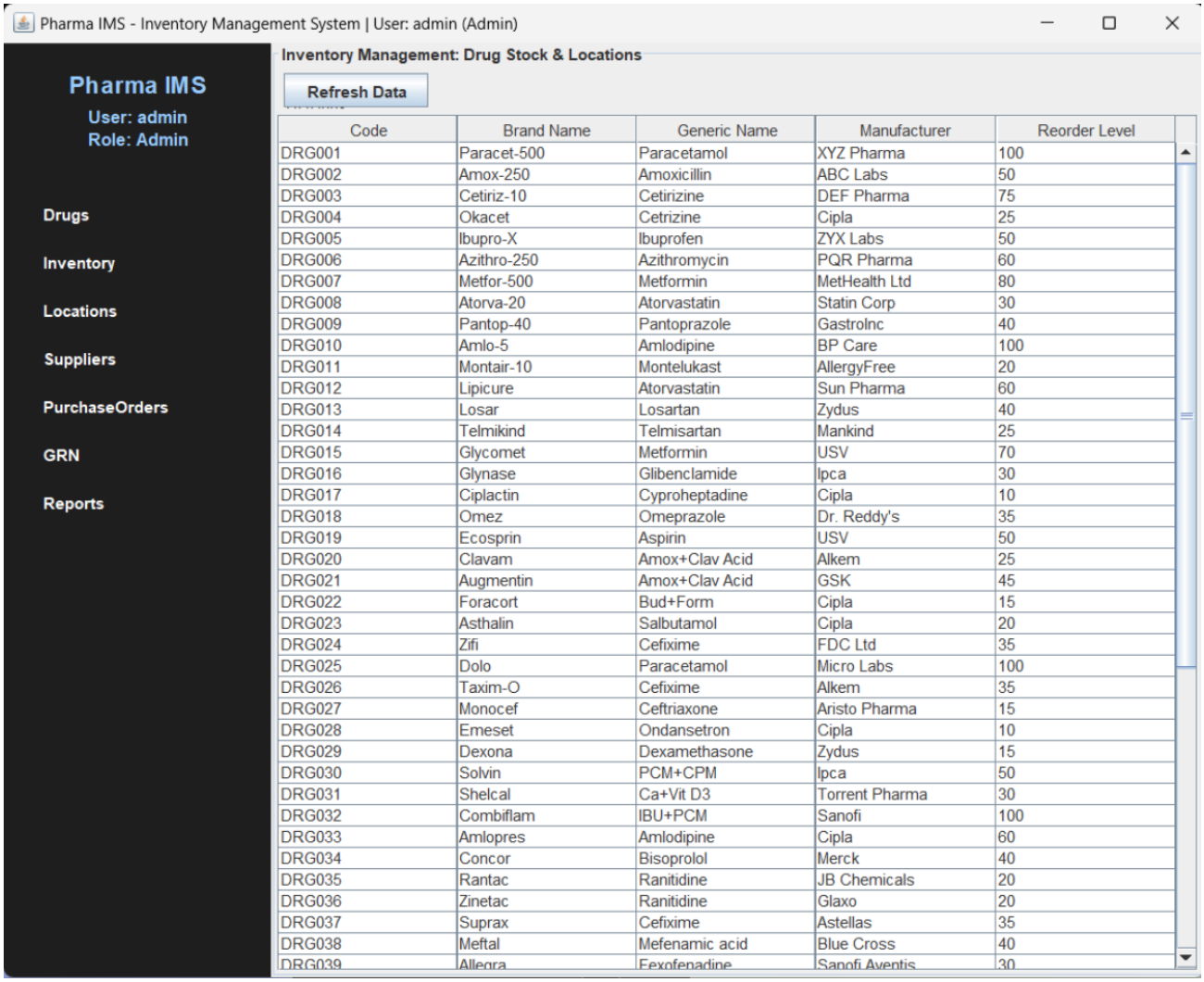


Figure 5.4. CRUD operations on Drug_Master table

INVENTORY PANEL:

Displays the drug id, brand name, generic name, manufacturer and reorder level.



Code	Brand Name	Generic Name	Manufacturer	Reorder Level
DRG001	Paracet-500	Paracetamol	XYZ Pharma	100
DRG002	Amox-250	Amoxicillin	ABC Labs	50
DRG003	Cetiriz-10	Cetirizine	DEF Pharma	75
DRG004	Okacet	Cetrizine	Cipla	25
DRG005	Ibupro-X	Ibuprofen	ZYX Labs	50
DRG006	Azithro-250	Azithromycin	PQR Pharma	60
DRG007	Metfor-500	Metformin	MetHealth Ltd	80
DRG008	Atorva-20	Atorvastatin	Statin Corp	30
DRG009	Pantop-40	Pantoprazole	GastroInc	40
DRG010	Amlo-5	Amlodipine	BP Care	100
DRG011	Montair-10	Montelukast	AllergyFree	20
DRG012	Lipicure	Atorvastatin	Sun Pharma	60
DRG013	Losar	Losartan	Zydus	40
DRG014	Telmikind	Telmisartan	Mankind	25
DRG015	Glycomet	Metformin	USV	70
DRG016	Glynase	Glibenclamide	Ipca	30
DRG017	Ciplactin	Cyproheptadine	Cipla	10
DRG018	Omez	Omeprazole	Dr. Reddy's	35
DRG019	Ecosprin	Aspirin	USV	50
DRG020	Clavam	Amox+Clav Acid	Alkem	25
DRG021	Augmentin	Amox+Clav Acid	GSK	45
DRG022	Foracort	Bud+Form	Cipla	15
DRG023	Asthalin	Salbutamol	Cipla	20
DRG024	Zifi	Cefixime	FDC Ltd	35
DRG025	Dolo	Paracetamol	Micro Labs	100
DRG026	Taxim-O	Cefixime	Alkem	35
DRG027	Monocef	Ceftriaxone	Aristo Pharma	15
DRG028	Emeset	Ondansetron	Cipla	10
DRG029	Dexona	Dexamethasone	Zydus	15
DRG030	Solvin	PCM+CPM	Ipca	50
DRG031	Shelcal	Ca+Vit D3	Torrent Pharma	30
DRG032	Combiflam	IBU+PCM	Sanofi	100
DRG033	Amlopres	Amlodipine	Cipla	60
DRG034	Concor	Bisoprolol	Merck	40
DRG035	Rantac	Ranitidine	JB Chemicals	20
DRG036	Zinetac	Ranitidine	Glaxo	20
DRG037	Suprax	Cefixime	Astellas	35
DRG038	Meftal	Mefenamic acid	Blue Cross	40
DRG039	Allegra	Fexofenadine	Sanofi Aventis	30

Figure 5.5. Inventory Panel Display

LOCATION PANEL:

Displays the Location_master table, this panel displays location code, location name, description and capacity.

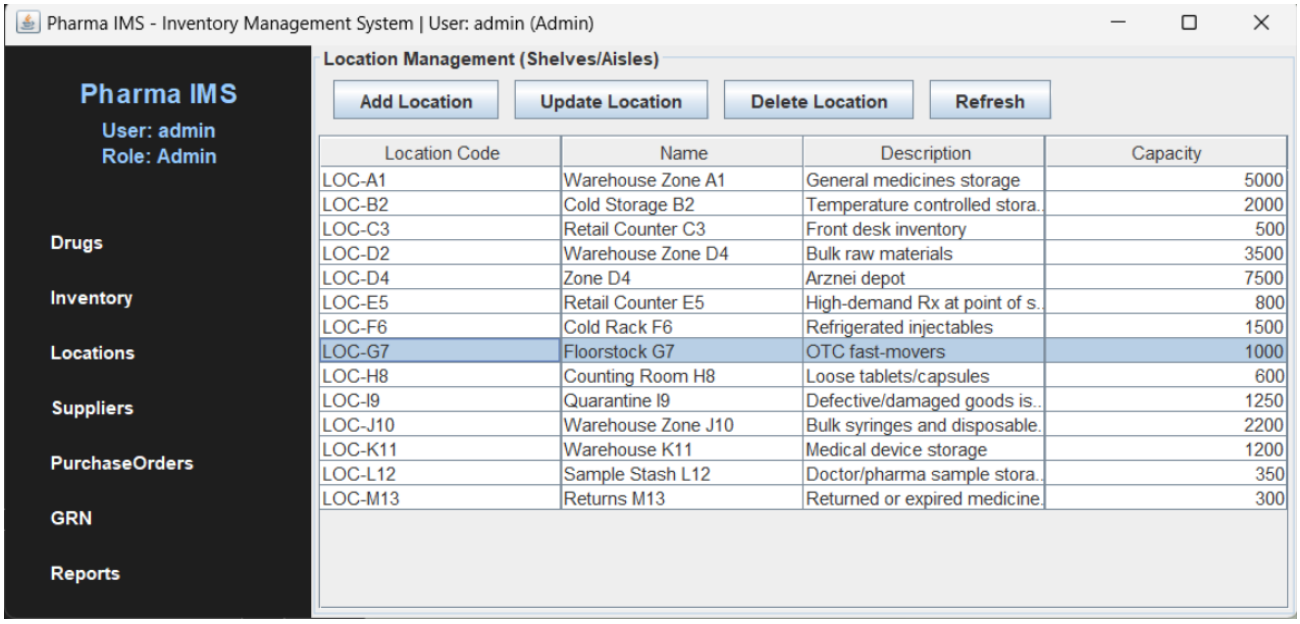
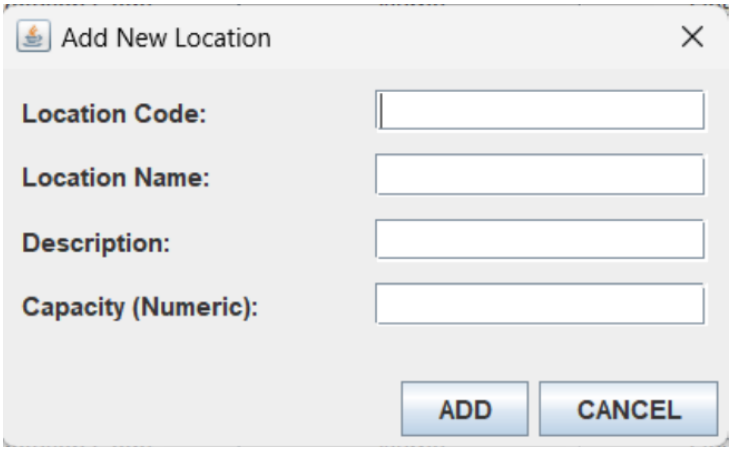
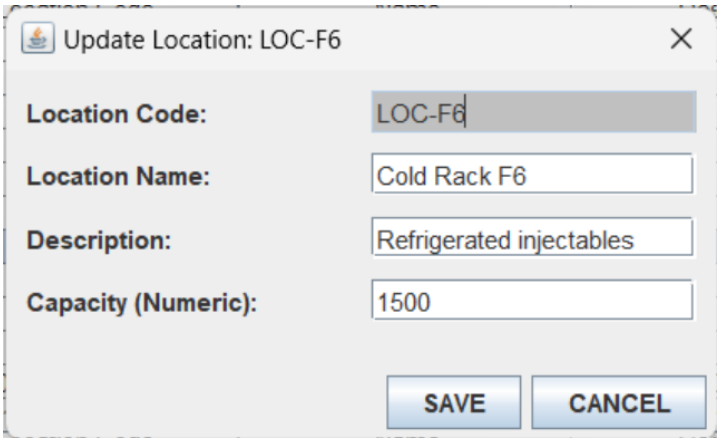


Figure 5.6. Location Panel Display

CRUD OPERATION IN LOCATION MANAGEMENT:

Performs Create, Update, Delete operations on Location_Master table.





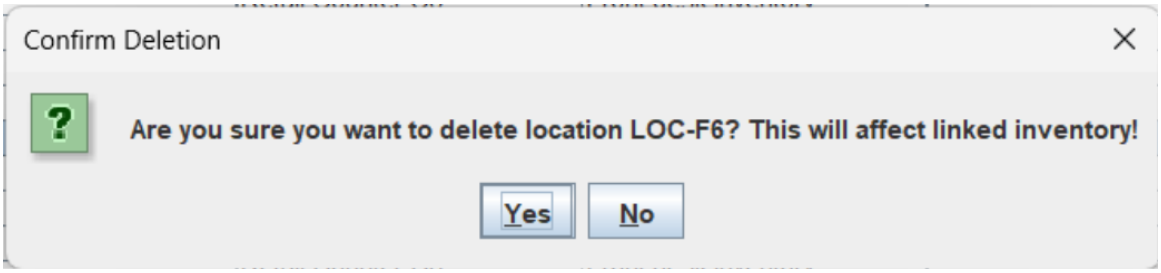


Figure 5.7. CRUD operations on Location_Master table

SUPPLIER MASTER:

Displays all the records from the Supplier_Master, supplier id, supplier name, contact person, email, phone, GSTIN, liscense id and payment terms.

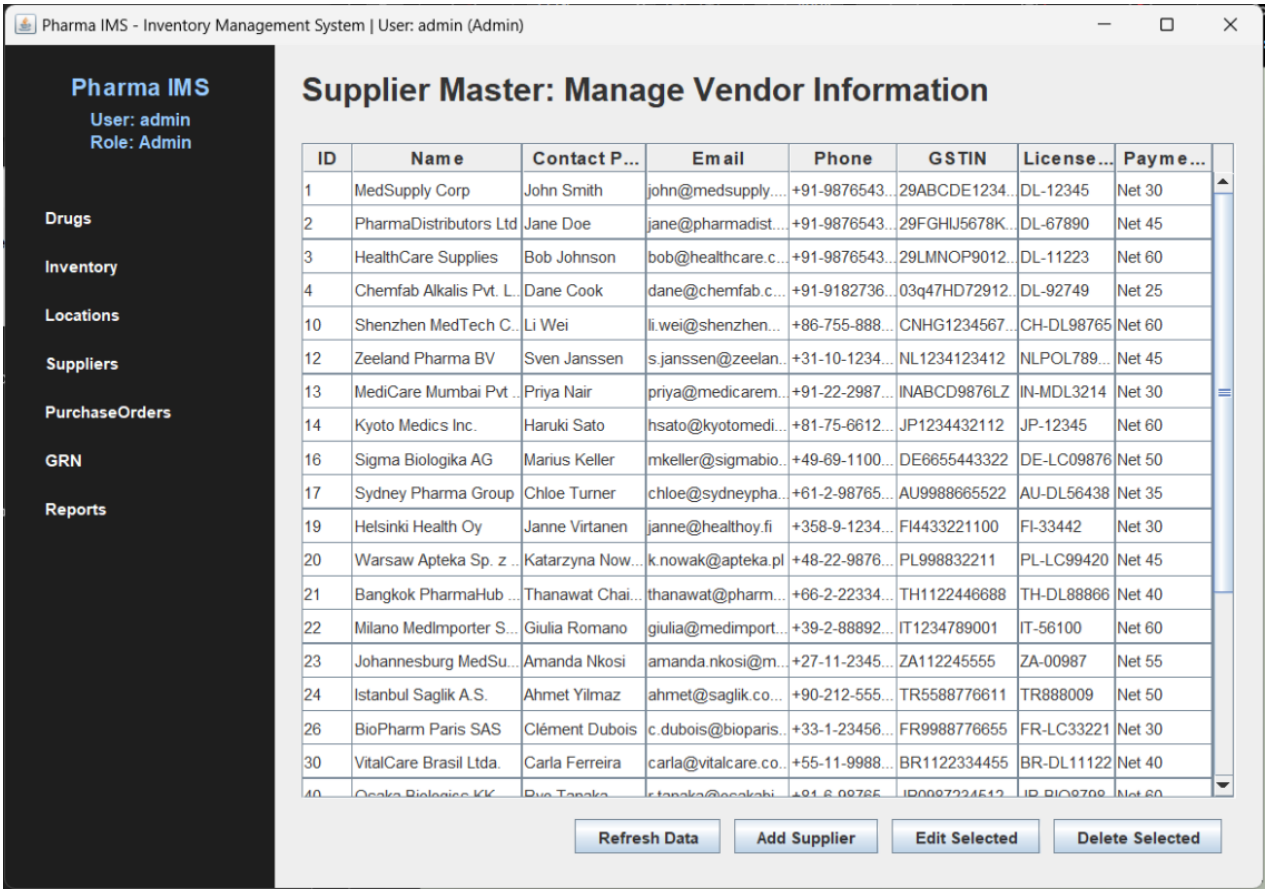


Figure 5.8. CRUD operations on Location_Master table

CRUD OPERATION IN SUPPLIER MASTER:

Performs Create, Update, Delete operations on Supplier_Master table.

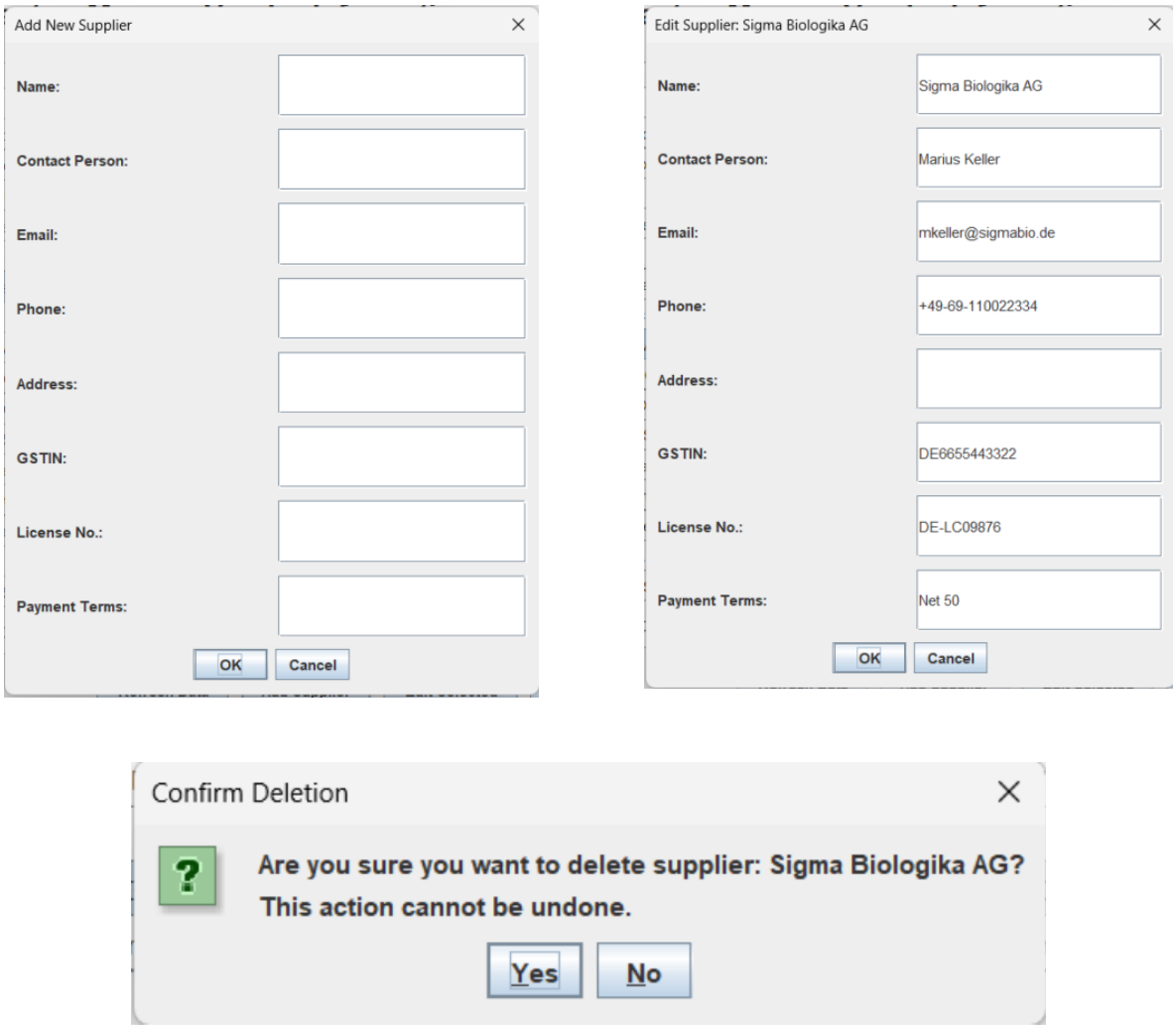


Figure 5.9. CRUD Operations on Supplier Master

PURCHASE ORDER:

Displays the purchase order details, namely, purchase order id, supplier name, order date, expected date, total cost and status.

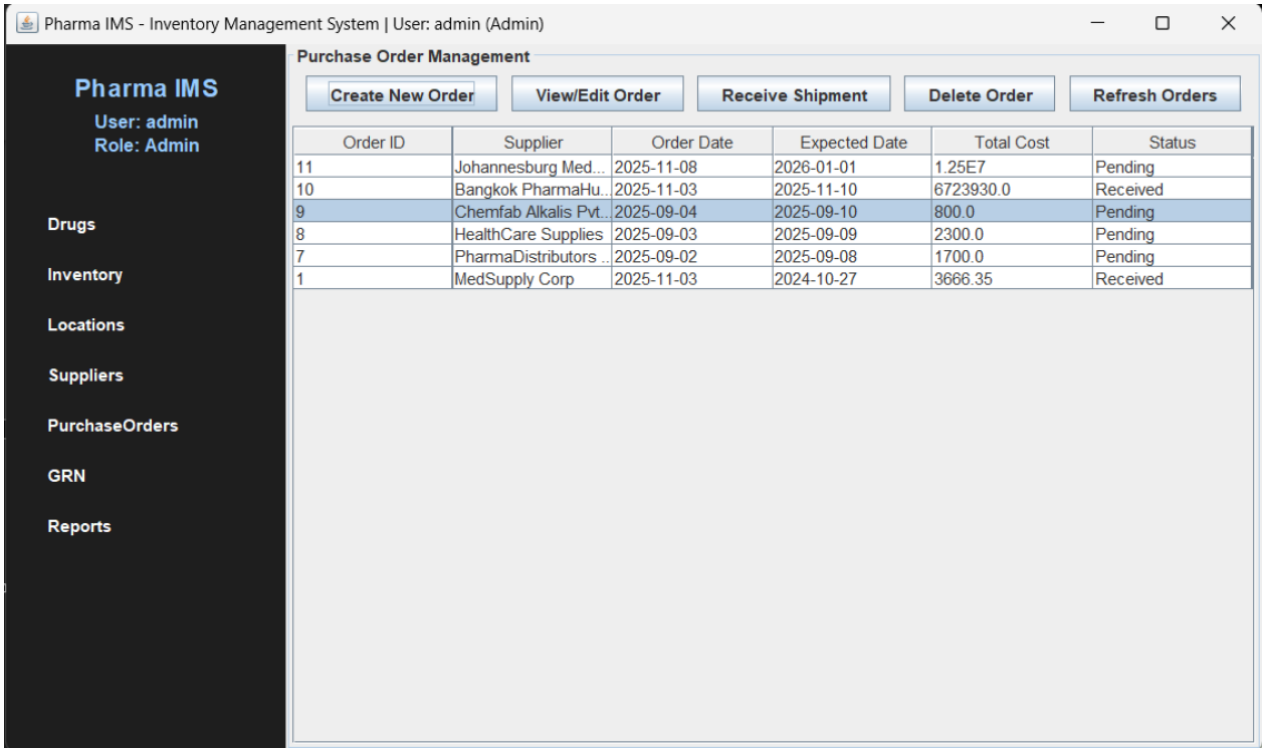


Figure 5.10. Purchase Order panel displaying purchase order details.

CRUD OPERATIONS ON PURCHASE ORDER:

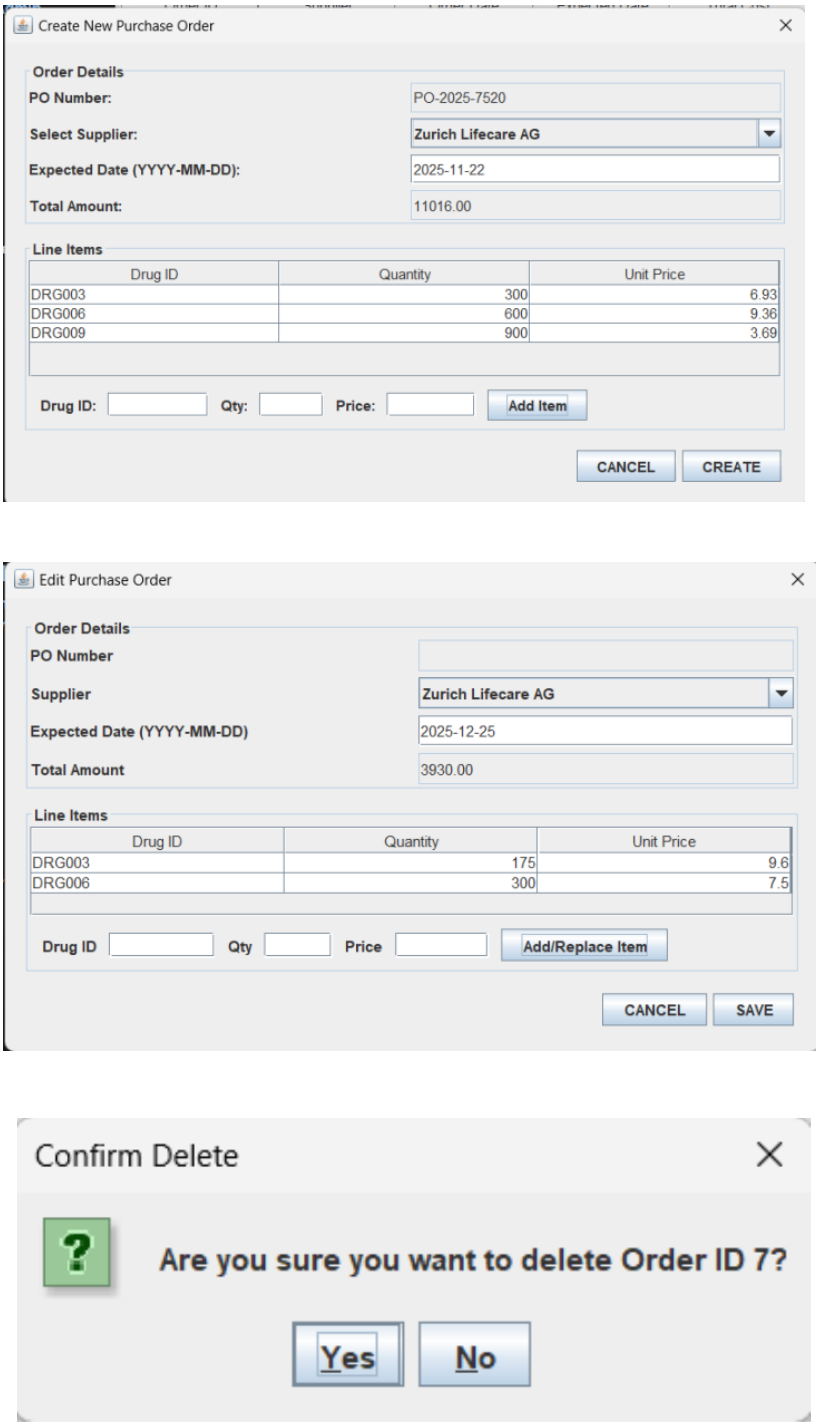


Figure 5.11. Performing CRUD operations on Purchase Order

CHAPTER 6

CONCLUSION AND FUTURE WORKS

The Pharmaceutical Inventory Management System successfully fulfils the objectives of providing a reliable, computerized, and batch-aware inventory solution. Leveraging MySQL's relational capabilities and Java's application framework, the project offers a structured way to manage critical pharmaceutical data, streamline procurement (PO/GRN), and enable granular stock control, thereby mitigating risks associated with drug expiry and manual errors. The design is modular and extensible, setting a solid foundation for future enhancements.

Potential future enhancements to the system include:

1. **Automated Alerts and Notifications:** Implement logic (e.g., scheduled tasks or triggers) to automatically generate alerts when the stock quantity falls below the specified reorder_level for a drug or when a batch's exp_date is nearing.

2. **Reporting Module Enhancement:** Develop advanced analytical reports (as noted in ReportsPanel.java) for drug consumption rate, supplier performance, and financial valuation of inventory.

3. **Advanced User Management:** Implement stronger security by replacing plaintext passwords with industry-standard hashing and salting techniques for the password_hash field in User_Master.

4. **Sales/Outward Transaction Module:** Develop a module to track drug dispensing or sales to accurately manage outgoing inventory and maintain a complete audit trail.

REFERENCES

[1] Smith, J. R., & Johnson, A. M. (Year). *Optimizing Pharmaceutical Inventory Management through Database Systems*. Journal of Healthcare Logistics https://www.researchgate.net/publication/384987682_Optimizing_pharmaceutical_inventory_management_A_global_framework_for_efficiency_and_cost_reduction

[2] World Health Organization (WHO). (2020). *Good Storage and Distribution Practices for Medical Products*. https://www.who.int/publications/i/item/978-92-4-000182-4

[3] Vodnala Supriya1, Dr. G. Krishna Kumari, Design and Implementation of an Inventory Management System (IMS) https://ijrpr.com/uploads/V6ISSUE8/IJRPR52286.pdf

[4] Silberschatz, A., Korth, H. F., & Sudarshan, S. S. (2020). *Database System Concepts*, 7th Edition, McGraw-Hill.

[5] Elmasri, Ramez. Fundamentals of database systems / Ramez Elmasri, Shamkant B. Navathe.—6th ed. p. cm

[6] Herbert Schildt and Dr. Danny Coward. *Java: The Complete Reference, Thirteenth Edition*

[7] MySQL Official Documentation. *MySQL 8.4 Reference Manual*. https://dev.mysql.com/doc/refman/8.4/en/

[8] Oracle (Java) Official Documentation. JDBC™ API Tutorial and Reference. https://docs.oracle.com/javase/tutorial/jdbc/

APPENDIX

DATABASE SCHEMA DEFINITION

```
-- 1. Create Database and Switch Context
CREATE DATABASE IF NOT EXISTS pharma_ims;
USE pharma_ims;

-- 2. Drop existing tables for a clean slate (Run this script multiple times)
DROP TABLE IF EXISTS GRN_Item;
DROP TABLE IF EXISTS Goods_Received_Note;
DROP TABLE IF EXISTS PurchaseOrder_Item;
DROP TABLE IF EXISTS Purchase_Order;
DROP TABLE IF EXISTS Stock_Inventory;
DROP TABLE IF EXISTS Drug_Master;
DROP TABLE IF EXISTS Supplier_Master;
DROP TABLE IF EXISTS Location_Master;
DROP TABLE IF EXISTS User_Master;

-- ------------------------------------------------------------------------
- MASTER TABLES (Primary Data)
-- ------------------------------------------------------------------------

-- 3. Supplier Master Table
CREATE TABLE IF NOT EXISTS Supplier_Master (
supplier_id INT AUTO_INCREMENT PRIMARY KEY,
supplier_name VARCHAR(255) NOT NULL,
contact_person VARCHAR(255),
address TEXT,
email VARCHAR(255),
phone_number VARCHAR(20),
gstin VARCHAR(50),
drug_license_number VARCHAR(100),
payment_terms VARCHAR(255),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- 4. Drug Master Table (Includes Foreign Key to Supplier)
CREATE TABLE IF NOT EXISTS Drug_Master (
material_code VARCHAR(50) PRIMARY KEY,
brand_name VARCHAR(255) NOT NULL,
generic_name VARCHAR(255),
manufacturer VARCHAR(255),
formulation VARCHAR(100),
strength VARCHAR(100),
schedule_category VARCHAR(50),
storage_conditions TEXT,
reorder_level INT DEFAULT 0,
is_active BOOLEAN DEFAULT TRUE,

preferred_supplier_id INT,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
```

```
-- Define the Foreign Key constraint for preferred supplier
FOREIGN KEY (preferred_supplier_id) REFERENCES Supplier_Master(supplier_id)
ON DELETE SET NULL
ON UPDATE CASCADE
);

-- 5. Location Master Table
CREATE TABLE IF NOT EXISTS Location_Master (
location_code VARCHAR(50) PRIMARY KEY,
location_name VARCHAR(255) NOT NULL,
description TEXT,
capacity INT DEFAULT 0,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- 6. User Master Table (Using plaintext passwords for simple AuthService.java match)
CREATE TABLE IF NOT EXISTS User_Master (
user_id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(100) UNIQUE NOT NULL,
password_hash VARCHAR(255) NOT NULL,
full_name VARCHAR(255),
role VARCHAR(50),
is_active BOOLEAN DEFAULT TRUE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- ----------------------------------------------------------------------
- TRANSACTIONAL & INVENTORY TABLES
-- ----------------------------------------------------------------------

-- 7. Purchase Order Header Table (Needed for PurchaseOrder.java functionality)
CREATE TABLE IF NOT EXISTS Purchase_Order (
po_id INT AUTO_INCREMENT PRIMARY KEY,
supplier_id INT NOT NULL,
order_date DATE NOT NULL,
expected_date DATE,
total_amount DECIMAL(10, 2) DEFAULT 0.00,
status VARCHAR(50) DEFAULT 'Pending', -- Pending, Shipped, Received
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (supplier_id) REFERENCES Supplier_Master(supplier_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

-- 8. Purchase Order Item/Line Table
CREATE TABLE IF NOT EXISTS PurchaseOrder_Item (
po_item_id INT AUTO_INCREMENT PRIMARY KEY,
po_id INT NOT NULL,
drug_id VARCHAR(50) NOT NULL, -- Links to Drug_Master.material_code
quantity INT NOT NULL,
unit_price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (po_id) REFERENCES Purchase_Order(po_id)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (drug_id) REFERENCES Drug_Master(material_code)
```

```
ON DELETE RESTRICT
ON UPDATE CASCADE
);

-- 9. Goods Received Note Header Table (Needed for GRN.java functionality)
CREATE TABLE IF NOT EXISTS Goods_Received_Note (
grn_id INT AUTO_INCREMENT PRIMARY KEY,
po_id INT NOT NULL,
received_date DATETIME DEFAULT CURRENT_TIMESTAMP,
received_by VARCHAR(255),
status VARCHAR(50) DEFAULT 'Verified',
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY (po_id) REFERENCES Purchase_Order(po_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

-- 10. GRN Item/Line Table (Contains batch details for inventory)
CREATE TABLE IF NOT EXISTS GRN_Item (
grn_item_id INT AUTO_INCREMENT PRIMARY KEY,
grn_id INT NOT NULL,
drug_id VARCHAR(50) NOT NULL, -- Links to Drug_Master.material_code
batch_number VARCHAR(100) NOT NULL,
quantity_received INT NOT NULL,
expiry_date DATE NOT NULL,

FOREIGN KEY (grn_id) REFERENCES Goods_Received_Note(grn_id)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (drug_id) REFERENCES Drug_Master(material_code)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

-- 11. Stock/Inventory Table (The core warehouse stock levels)
CREATE TABLE IF NOT EXISTS Stock_Inventory (
stock_id INT AUTO_INCREMENT PRIMARY KEY,
material_code VARCHAR(50) NOT NULL,
location_code VARCHAR(50) NOT NULL,
batch_number VARCHAR(100) NOT NULL,
quantity INT NOT NULL DEFAULT 0,
unit_cost DECIMAL(10, 2),
 mfg_date DATE,
 exp_date DATE,

UNIQUE KEY (material_code, location_code, batch_number),
FOREIGN KEY (material_code) REFERENCES Drug_Master(material_code)
ON DELETE RESTRICT
ON UPDATE CASCADE,
FOREIGN KEY (location_code) REFERENCES Location_Master(location_code)
ON DELETE RESTRICT
ON UPDATE CASCADE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- ----------------------------------------------------------------------
- SAMPLE DATA INSERTION
```

```
-- ---------------------------------------------------------------------

-- Sample Suppliers
INSERT INTO Supplier_Master (supplier_name, contact_person, email, phone_number, gstin,
drug_license_number, payment_terms) VALUES
('MedSupply Corp', 'John Smith', 'john@medsupply.com', '+91-9876543210', '29ABCDE1234F1Z5',
'DL-12345', 'Net 30'),
('PharmaDistributors Ltd', 'Jane Doe', 'jane@pharmadist.com', '+91-9876543211',
'29FGHIJ5678K2Z6', 'DL-67890', 'Net 45'),
('HealthCare Supplies', 'Bob Johnson', 'bob@healthcare.com', '+91-9876543212',
'29LMNOP9012Q3Z7', 'DL-11223', 'Net 60');

-- Sample Drugs (using supplier IDs from above: 1, 2)
INSERT INTO Drug_Master (material_code, brand_name, generic_name, manufacturer, formulation,
strength, schedule_category, storage_conditions, reorder_level, is_active, preferred_supplier_id)
VALUES
('DRG001', 'Paracet-500', 'Paracetamol', 'XYZ Pharma', 'Tablet', '500mg', 'OTC', 'Store below 30°C',
100, TRUE, 1),
('DRG002', 'Amox-250', 'Amoxicillin', 'ABC Labs', 'Capsule', '250mg', 'Schedule H', 'Store in cool
place', 50, TRUE, 2),
('DRG003', 'Cetiriz-10', 'Cetirizine', 'DEF Pharma', 'Tablet', '10mg', 'OTC', 'Store below 25°C', 75,
TRUE, 1);

-- Sample Locations
INSERT INTO Location_Master (location_code, location_name, description, capacity) VALUES
('LOC-A1', 'Warehouse Zone A1', 'General medicines storage', 5000),
('LOC-B2', 'Cold Storage B2', 'Temperature controlled storage', 2000),
('LOC-C3', 'Retail Counter C3', 'Front desk inventory', 500);

-- Sample Users (Passwords are plaintext: 'adminpass', 'pharmacistpass', 'staffpass')
INSERT INTO User_Master (username, password_hash, full_name, role, is_active) VALUES
('admin', 'adminpass', 'System Administrator', 'Admin', TRUE),
('pharmacist', 'pharmacistpass', 'Dr. Sarah Williams', 'Pharmacist', TRUE),
('staff', 'staffpass', 'Mike Brown', 'Staff', TRUE);

-- Sample Transactional Data (to populate PO and GRN tables)

-- 1. Create a Purchase Order (from MedSupply Corp - ID 1)
INSERT INTO Purchase_Order (supplier_id, order_date, expected_date, total_amount, status)
VALUES
(1, '2024-10-20', '2024-10-27', 1000.00, 'Pending');
SET @last_po_id = LAST_INSERT_ID(); -- Get the ID of the newly created PO

-- 2. Add Items to that Purchase Order
INSERT INTO PurchaseOrder_Item (po_id, drug_id, quantity, unit_price) VALUES
(@last_po_id, 'DRG001', 500, 1.50),
(@last_po_id, 'DRG003', 250, 1.00);

-- 3. Create a Goods Received Note for that PO (Simulating a full receipt)
INSERT INTO Goods_Received_Note (po_id, received_date, received_by, status) VALUES
(@last_po_id, NOW(), 'admin', 'Verified');
SET @last_grn_id = LAST_INSERT_ID(); -- Get the ID of the newly created GRN

-- 4. Add Items (Batches) to the GRN
INSERT INTO GRN_Item (grn_id, drug_id, batch_number, quantity_received, expiry_date)
VALUES
(@last_grn_id, 'DRG001', 'B-1001-A', 500, '2025-12-31'),
(@last_grn_id, 'DRG003', 'C-3001-Z', 250, '2026-06-30');

-- Final Stock/Inventory (Should reflect the GRN)
```

```
INSERT INTO Stock_Inventory (material_code, location_code, batch_number, quantity, unit_cost,
mfg_date, exp_date) VALUES
('DRG001', 'LOC-A1', 'B-1001-A', 500, 1.50, '2024-01-01', '2025-12-31'), -- From the new GRN
('DRG003', 'LOC-A1', 'C-3001-Z', 250, 1.00, '2024-01-01', '2026-06-30'), -- From the new GRN
('DRG002', 'LOC-B2', 'BATCH002B', 100, 2.75, '2024-03-15', '2026-03-15'); -- Existing sample
```

JAVA SERVICE FOR DRUG MASTER CRUD OPERATIONS (DrugService.JAVA)

```
package com.pharma_ims.service;

import com.pharma_ims.db.DatabaseService;
import com.pharma_ims.entity.Drug; // Assuming you have a Drug entity class
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DrugService {

   private static final Logger LOGGER = Logger.getLogger(DrugService.class.getName());

   // --- CREATE Operation ---
   public boolean addDrug(Drug drug) {
      Connection conn = null;
      PreparedStatement stmt = null;
      try {
         conn = DatabaseService.getConnection();
         String sql = "INSERT INTO Drug_Master (material_code, brand_name,
generic_name, manufacturer, " +
                      "formulation, strength, schedule_category, storage_conditions,
reorder_level, " +
                      "preferred_supplier_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
         stmt = conn.prepareStatement(sql);
         stmt.setString(1, drug.getMaterialCode());
         stmt.setString(2, drug.getBrandName());
         stmt.setString(3, drug.getGenericName());
         stmt.setString(4, drug.getManufacturer());
         stmt.setString(5, drug.getFormulation());
         stmt.setString(6, drug.getStrength());
         stmt.setString(7, drug.getScheduleCategory());
         stmt.setString(8, drug.getStorageConditions());
         stmt.setInt(9, drug.getReorderLevel());
         // Handle null for preferred_supplier_id if not set
         if (drug.getPreferredSupplierId() != null) {
            stmt.setInt(10, drug.getPreferredSupplierId());
         } else {
            stmt.setNull(10, java.sql.Types.INTEGER);
         }
```

```
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                LOGGER.info("Drug added successfully: " + drug.getBrandName());
                return true;
            }
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Error adding drug: " + drug.getBrandName(), e);
        } finally {
            DatabaseService.closeResources(null, stmt, conn);
        }
        return false;
    }

    // --- READ Operation (Retrieve all drugs) ---
    public List<Drug> getAllDrugs() {
        List<Drug> drugs = new ArrayList<>();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            conn = DatabaseService.getConnection();
            String sql = "SELECT material_code, brand_name, generic_name, manufacturer,
formulation, strength, " +
                         "schedule_category, storage_conditions, reorder_level, is_active,
preferred_supplier_id " +
                         "FROM Drug_Master WHERE is_active = TRUE"; // Assuming you might
filter active even if no soft delete
            stmt = conn.prepareStatement(sql);
            rs = stmt.executeQuery();

            while (rs.next()) {
                Drug drug = new Drug();
                drug.setMaterialCode(rs.getString("material_code"));
                drug.setBrandName(rs.getString("brand_name"));
                drug.setGenericName(rs.getString("generic_name"));
                drug.setManufacturer(rs.getString("manufacturer"));
                drug.setFormulation(rs.getString("formulation"));
                drug.setStrength(rs.getString("strength"));
                drug.setScheduleCategory(rs.getString("schedule_category"));
                drug.setStorageConditions(rs.getString("storage_conditions"));
                drug.setReorderLevel(rs.getInt("reorder_level"));
                drug.setActive(rs.getBoolean("is_active"));
                drug.setPreferredSupplierId(rs.getObject("preferred_supplier_id", Integer.class));
// Handle nullable int
                drugs.add(drug);
            }
            LOGGER.info("Retrieved " + drugs.size() + " drugs.");
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Error retrieving all drugs.", e);
        } finally {
            DatabaseService.closeResources(rs, stmt, conn);
        }
        return drugs;
    }
```

```
    // --- READ Operation (Retrieve drug by material code) ---
    public Drug getDrugByMaterialCode(String materialCode) {
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        Drug drug = null;
        try {
            conn = DatabaseService.getConnection();
            String sql = "SELECT material_code, brand_name, generic_name, manufacturer,
formulation, strength, " +
                         "schedule_category, storage_conditions, reorder_level, is_active,
preferred_supplier_id " +
                         "FROM Drug_Master WHERE material_code = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, materialCode);
            rs = stmt.executeQuery();

            if (rs.next()) {
                drug = new Drug();
                drug.setMaterialCode(rs.getString("material_code"));
                drug.setBrandName(rs.getString("brand_name"));
                drug.setGenericName(rs.getString("generic_name"));
                drug.setManufacturer(rs.getString("manufacturer"));
                drug.setFormulation(rs.getString("formulation"));
                drug.setStrength(rs.getString("strength"));
                drug.setScheduleCategory(rs.getString("schedule_category"));
                drug.setStorageConditions(rs.getString("storage_conditions"));
                drug.setReorderLevel(rs.getInt("reorder_level"));
                drug.setActive(rs.getBoolean("is_active"));
                drug.setPreferredSupplierId(rs.getObject("preferred_supplier_id", Integer.class));
                LOGGER.info("Drug found: " + drug.getBrandName());
            } else {
                LOGGER.warning("Drug not found with material code: " + materialCode);
            }
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Error retrieving drug by material code: " +
materialCode, e);
        } finally {
            DatabaseService.closeResources(rs, stmt, conn);
        }
        return drug;
    }


    // --- UPDATE Operation ---
    public boolean updateDrug(Drug drug) {
        Connection conn = null;
        PreparedStatement stmt = null;
        try {
            conn = DatabaseService.getConnection();
            String sql = "UPDATE Drug_Master SET brand_name = ?, generic_name = ?,
manufacturer = ?, " +
                         "formulation = ?, strength = ?, schedule_category = ?, storage_conditions =
```

```
?, " +
                    "reorder_level = ?, is_active = ?, preferred_supplier_id = ?, updated_at =
CURRENT_TIMESTAMP " +
                    "WHERE material_code = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, drug.getBrandName());
            stmt.setString(2, drug.getGenericName());
            stmt.setString(3, drug.getManufacturer());
            stmt.setString(4, drug.getFormulation());
            stmt.setString(5, drug.getStrength());
            stmt.setString(6, drug.getScheduleCategory());
            stmt.setString(7, drug.getStorageConditions());
            stmt.setInt(8, drug.getReorderLevel());
            stmt.setBoolean(9, drug.isActive());
            if (drug.getPreferredSupplierId() != null) {
                stmt.setInt(10, drug.getPreferredSupplierId());
            } else {
                stmt.setNull(10, java.sql.Types.INTEGER);
            }
            stmt.setString(11, drug.getMaterialCode());

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                LOGGER.info("Drug updated successfully: " + drug.getBrandName());
                return true;
            }
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Error updating drug: " + drug.getBrandName(), e);
        } finally {
            DatabaseService.closeResources(null, stmt, conn);
        }
        return false;
    }

    // --- DELETE Operation (Standard Hard Delete) ---
    public boolean deleteDrug(String materialCode) {
        Connection conn = null;
        PreparedStatement stmt = null;
        try {
            conn = DatabaseService.getConnection();
            String sql = "DELETE FROM Drug_Master WHERE material_code = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, materialCode);

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                LOGGER.info("Drug (material_code: " + materialCode + ") deleted
successfully.");
                return true;
            }
        } catch (SQLException e) {
            // This catch block is important! If there are foreign key constraints,
            // the database will prevent deletion if related records exist (e.g., in
PurchaseOrder_Item).
```

```
            LOGGER.log(Level.SEVERE, "Error deleting drug: " + materialCode + ". " +
                    "Deletion might be blocked by existing related records (e.g., in Purchase
Orders or GRNs).", e);
        } finally {
            DatabaseService.closeResources(null, stmt, conn);
        }
        return false;
    }
}
```

JAVA DATABASE SERVICE CLASS (DatabaseService.java)

```
package com.pharma_ims.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DatabaseService {

    private static final Logger LOGGER =
Logger.getLogger(DatabaseService.class.getName());

    // Database connection parameters
    private static final String DB_URL = "jdbc:mysql://localhost:3306/pharma_ims";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "SiriusBlack@369";

    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            LOGGER.info("MySQL JDBC Driver registered!");
        } catch (ClassNotFoundException e) {
            LOGGER.log(Level.SEVERE, "MySQL JDBC Driver not found!", e);
            System.exit(1); // Exit if driver is not found
        }
    }

    /**
     * Establishes a connection to the database.
     * @return A valid Connection object, or null if connection fails.
     */
    public static Connection getConnection() {
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
```

```
            LOGGER.info("Database connection established.");
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Failed to establish database connection.", e);
        }
        return connection;
    }

    /**
     * Closes a database connection, statement, and result set.
     * @param rs ResultSet to close (can be null)
     * @param stmt Statement to close (can be null)
     * @param conn Connection to close (can be null)
     */
    public static void closeResources(ResultSet rs, PreparedStatement stmt, Connection conn)
{
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
            LOGGER.info("Database resources closed.");
        } catch (SQLException e) {
            LOGGER.log(Level.SEVERE, "Error closing database resources.", e);
        }
    }
}
```