DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY, CHENNAI – 25.

OBJECT - ORIENTED ANALYSIS AND DESIGN – CS6110

MINI PROJECT

PHARMACY MANAGEMENT SYSTEM

TEAM NO **- 18:**

JOTHI SRI.S – 2022103049

PUNITHA.K – 2022103561

B.E. Computer Science and Engineering - Batch – P

College of Engineering Guindy, Anna University, Chennai – 25.

Table of Contents

1 ABSTRACT

The Pharmacy Management System is an advanced solution designed to streamline and automate the various operations of a pharmacy, including inventory management, sales processing, customer management, supplier coordination, and report generation. This project leverages Object-Oriented Analysis and Design (OOAD) principles to ensure a structured and modular approach, facilitating efficient handling of the pharmacy's day-to day activities. Our project is to develop a comprehensive system that effectively manages the critical functions of a pharmacy like, Inventory Management (To monitor medicine stock levels, add, update, and delete medicines in the inventory, track expiration dates, and automate the reordering process), Sales Processing (To handle the entire sales process, from billing and invoicing to tracking daily sales and applying necessary discounts and taxes), Customer Management (To store and manage detailed customer information, including contact details and prescription history, allowing for personalized service and better customer relationship management), Supplier Management (To manage supplier details and order history, ensuring efficient coordination and timely procurement of medicines) and Reporting (To generate a variety of reports, including daily sales summaries, inventory status updates, and financial statements, providing valuable insights for business management). The object-oriented approach identifies key objects within the system and defines their attributes and interactions. Key classes include Medicine (Includes attributes such as name, batch number, expiry date, price, and quantity are defined to manage stock effectively), Customer (This class manages customer details, including their prescription history), Pharmacist (Attributes like name, employee ID, and role are essential for managing pharmacist activities within the system), Supplier (Supplier information, including contact details and the list of medicines supplied, is managed within this class), Sales (This class handles sales transactions, tracking details like sale ID, date, total amount, and payment method), Order (This class manages orders placed with suppliers, including order ID, date, and the list of medicines ordered), Prescription (The system will track prescriptions with attributes such as prescription ID, date, and list of prescribed medicines) and Invoice (Invoicing details, including invoice ID, sale ID, date, list of medicines, total amount, taxes, and discounts, are managed within this class). By implementing this project using OOAD principles, the Pharmacy Management System will be a robust, maintainable, and scalable solution capable of supporting the efficient management of a pharmacy’s operations.

2 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

2.1 Introduction

2.1.1 Purpose:

The Online Pharmacy Management System (OPMS) is designed to provide a seamless platform for customers to purchase medicines online and for administrators to manage inventory, orders, and deliveries efficiently. It aims to improve customer convenience and streamline pharmacy operations through automation and secure online transactions.

2.1.2 Scope

The OPMS will include features such as:

- ❖ Customer registration and login.
- ❖ Browsing and searching for medicines.
- ❖ Adding medicines to a cart and online payment.
- ❖ Inventory and order management for administrators.
- ❖ Real-time stock updates and delivery tracking.

2.1.3 Overview

This document provides a detailed description of the system's functional and non-functional requirements, interface design, performance expectations, and constraints.

2.2 General Description

2.2.1 Product Perspective

The PMS is a web-based system integrating e-commerce functionalities with pharmacy management features. It eliminates the need for manual stock management and order processing, offering a scalable solution for pharmacies.

2.2.2 Product Functions

- ❖ Customer Features:
  - Search for medicines by category or name.
  - Add medicines to the cart and make secure payments.
- ❖ Admin Features:
  - Add, update, and delete medicines.
  - Manage orders, payments, and customer queries.
  - Generate inventory and sales reports.

2.3 Functional Requirements

2.3.1 User Authentication:

a. Admin, Pharmacist, and Customer login with unique credentials.
b. Password validation and error messaging for incorrect credentials.

2.3.2 Customer Features:

a. Search Medicine: Allow customers to search for medicines by name, category, or description.
b. View Medicine Details: Display detailed information, including availability and location in the pharmacy.
c. Add to Cart: Add medicines to a virtual cart for purchase.
d. View Cart: Display a summary of selected medicines, quantities, and prices.
e. Order Placement: Process orders and save them to the database only after successful payment.
f. View Order History: Allow customers to view past purchases with order details.
g. Profile Management: Enable customers to view and update their profile information.

2.3.3 Pharmacist Features:

a. View Inventory: Display the current stock of medicines.
b. Add Medicine: Allow adding new medicines to the database with all relevant details.
c. Update Medicine: Modify existing medicine details, including price, quantity, or description.
d. Delete Medicine: Remove medicines no longer in stock or discontinued from the inventory.

2.3.4 Admin Features:

a. Manage user accounts (create, update, and delete admin, pharmacist, or customer accounts).
b. View sales reports and statistics for better decision-making.

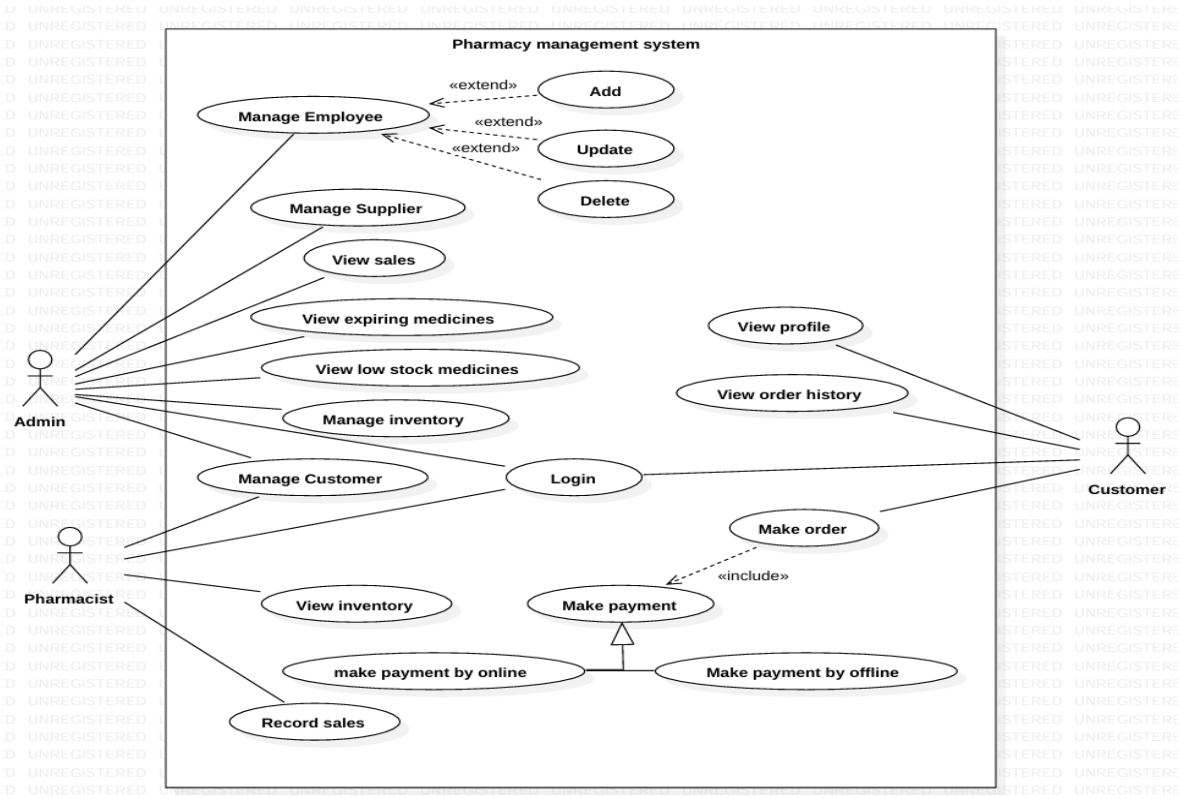
2.3.5 Database Management:

a. Store and manage all relevant data, including user credentials, medicine details, sales records, and order history.

3 UML DIAGRAMS

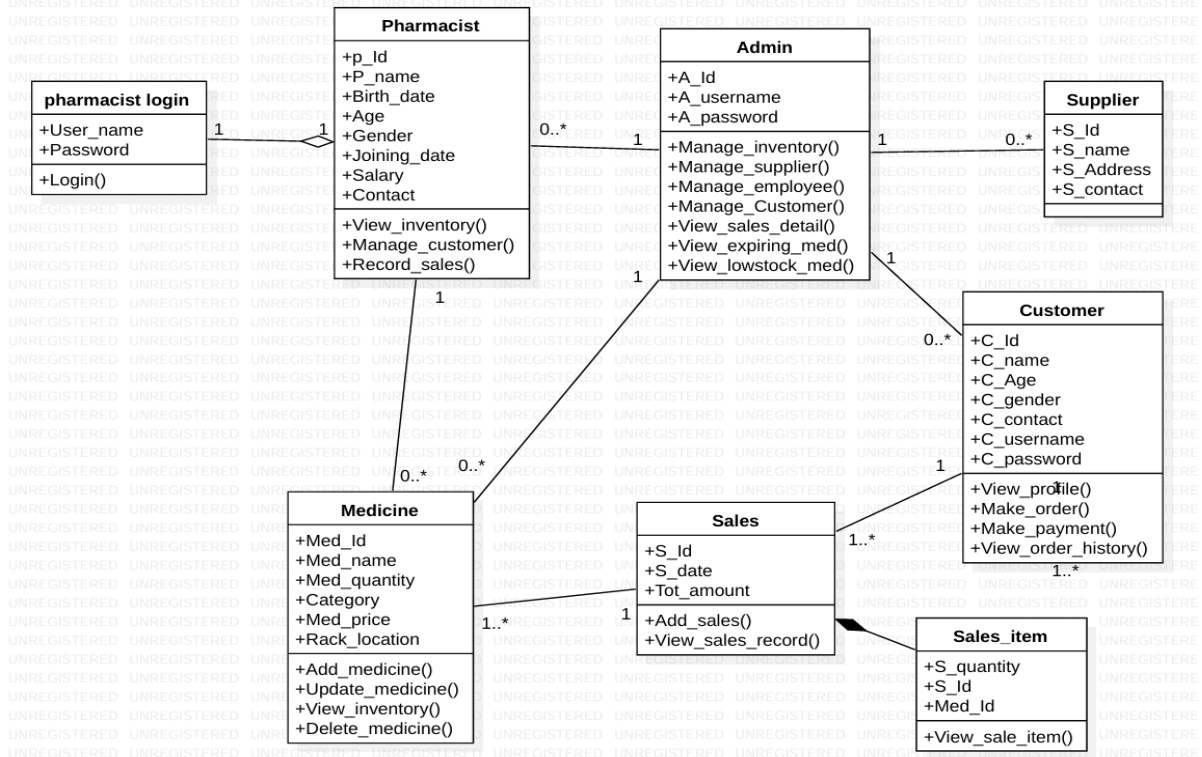
3.1 Use case:

A use case diagram is a visual representation of the interactions between actors (users or systems) and a system, showing the system's functionalities as use cases. It highlights the relationships between actors and use cases. Use case diagrams are used in UML to model system requirements.



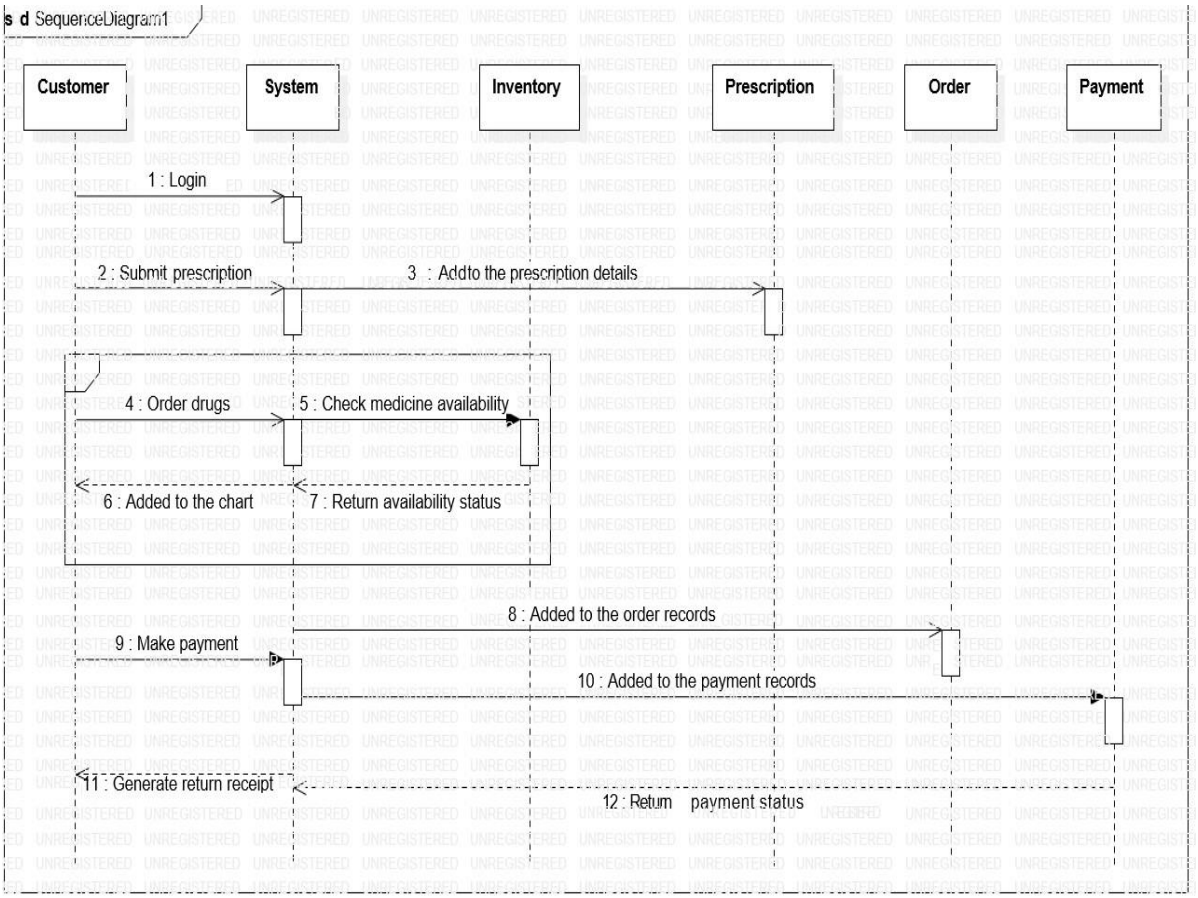
3.2 Class diagram:

A class diagram is a visual representation of the structure of a system, showing its classes, attributes, methods, and the relationships between them. It is a key component of UML, used to model the static view of a system. Class diagrams help in understanding and designing object-oriented systems.



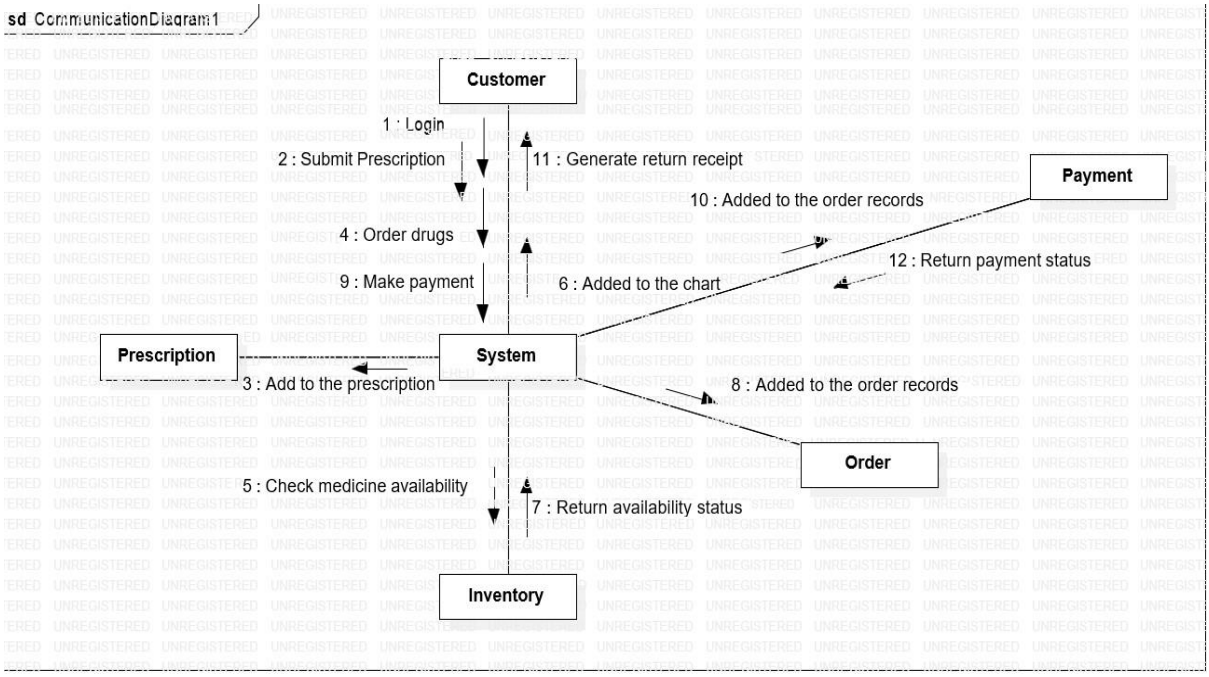
3.3 Sequence Diagram:

A sequence diagram is a UML diagram that illustrates the interaction between objects in a specific sequence to achieve a use case or process. It shows the flow of messages, events, or calls between actors and system components over time. Sequence diagrams are used to model dynamic behavior in a system.



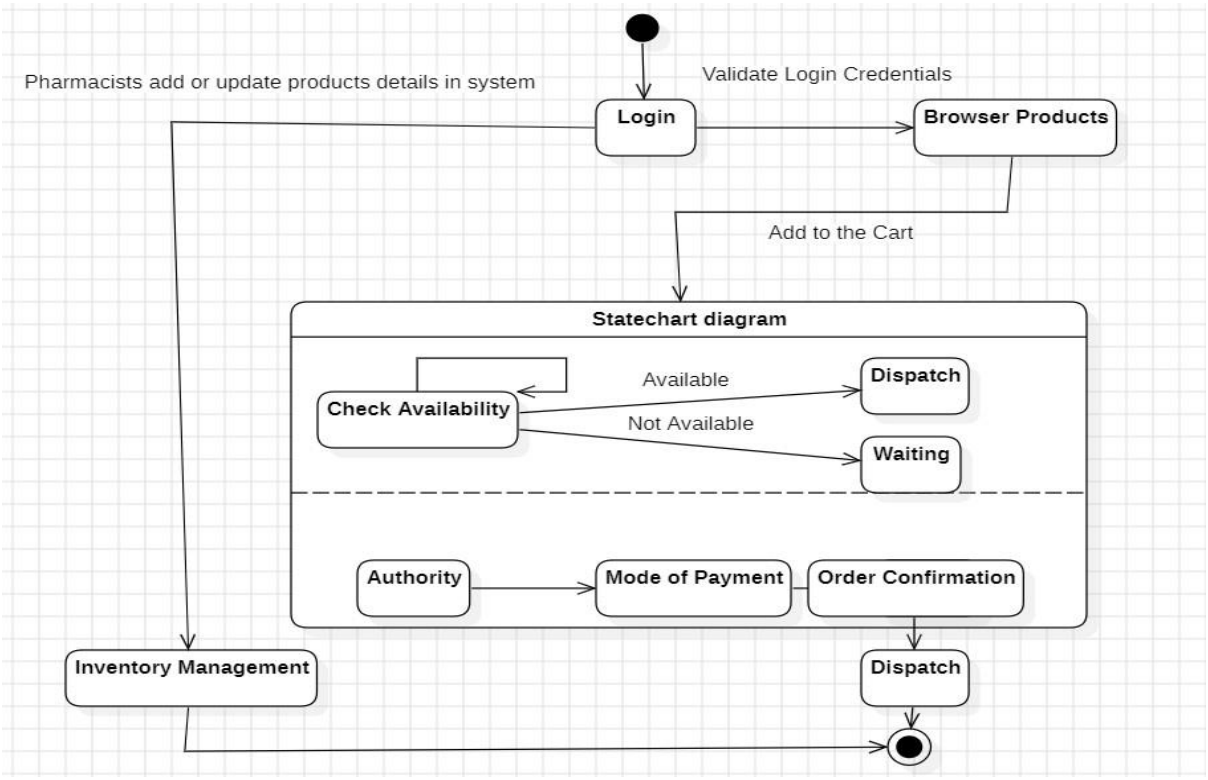
3.4 Collaboration diagram:

A collaboration diagram is a UML diagram that visualizes the interactions between objects and their relationships to achieve a specific task. It focuses on object roles, links, and message exchange. Collaboration diagrams emphasize the structural organization of objects during interactions.



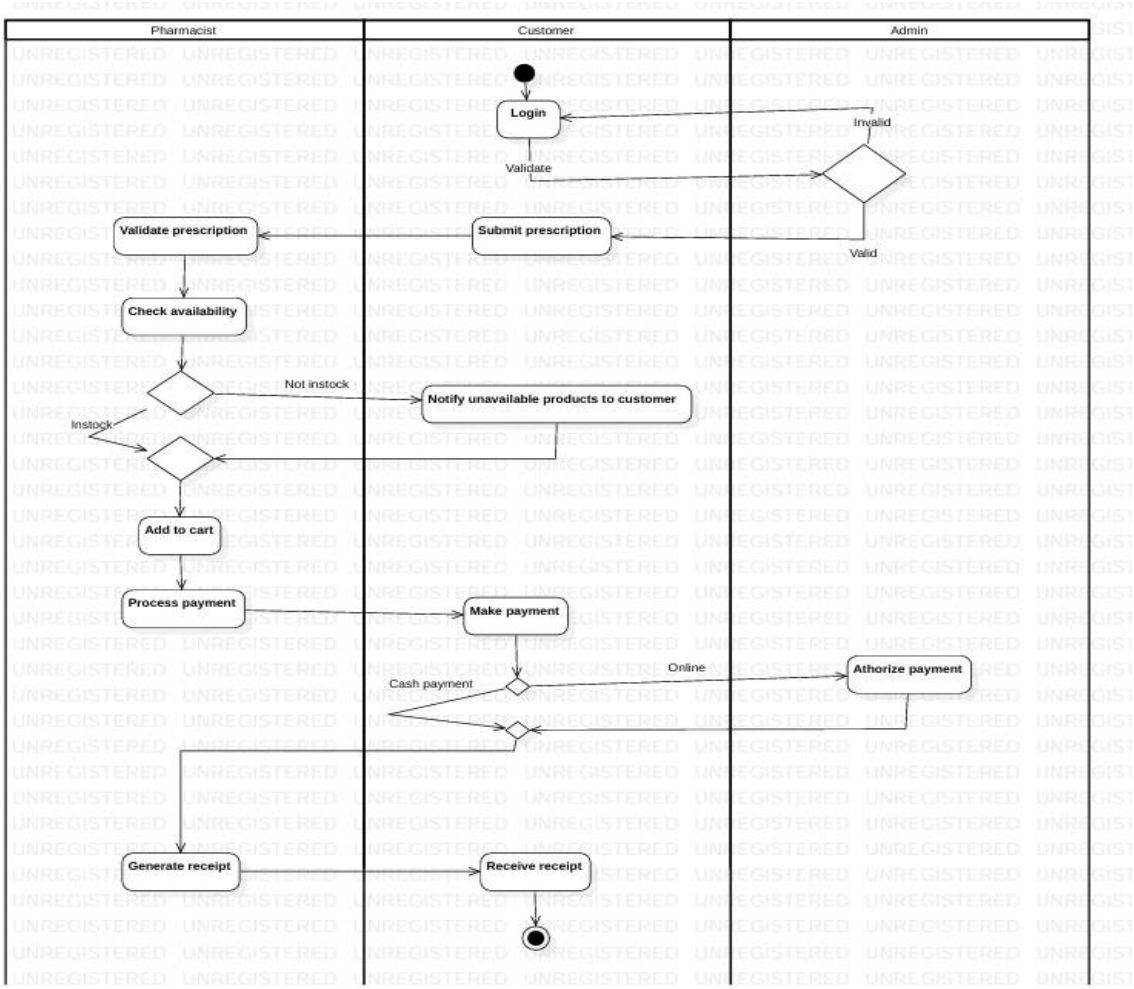
3.5 Statechart Diagram:

A statechart diagram is a UML diagram that represents the states of an object and the transitions between them in response to events. It models the dynamic behavior of a system, focusing on the object's lifecycle. Statechart diagrams are used to describe how an object changes state over time.



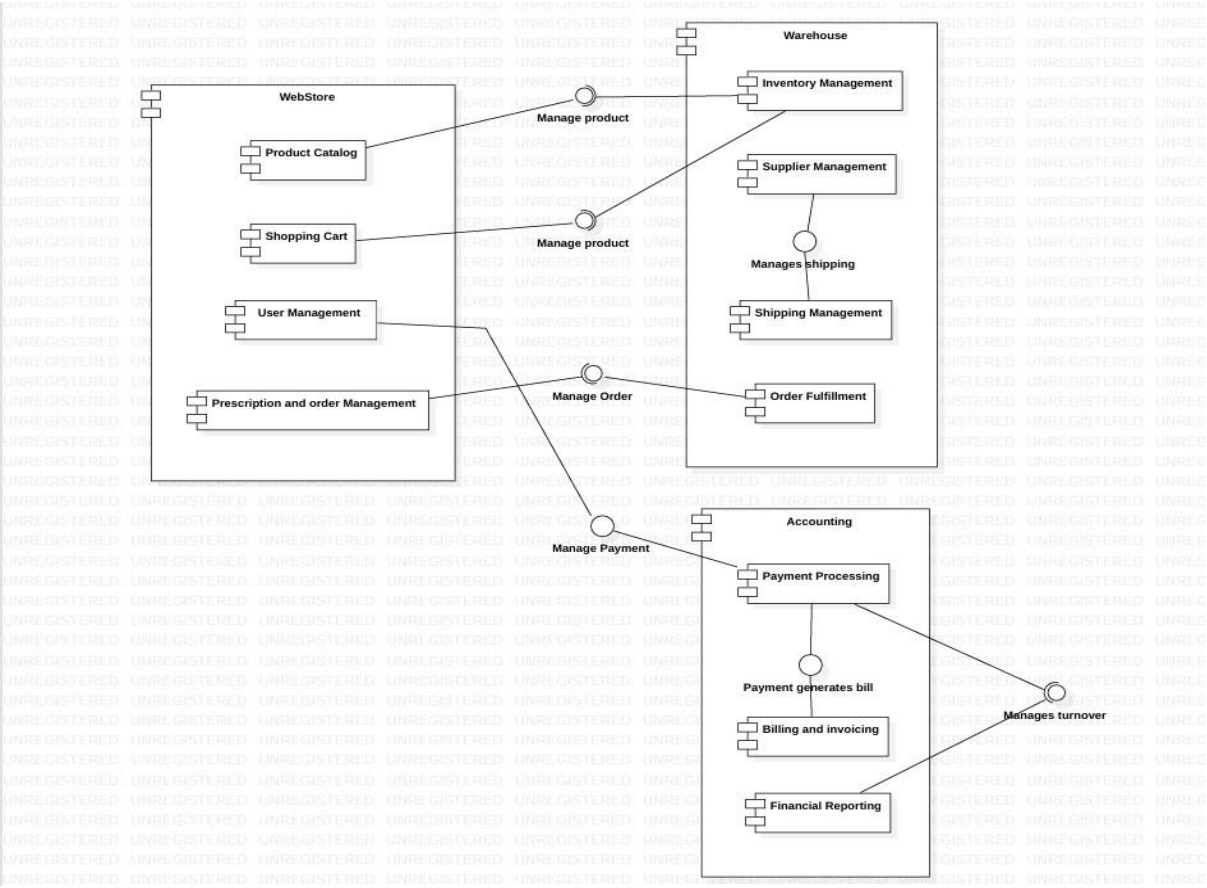
3.6 Activity Diagram:

An activity diagram is a UML diagram that depicts the flow of activities or tasks in a process or system. It represents the sequential and concurrent workflows, including decision points and loops. Activity diagrams are used to model the dynamic aspects of a system or business process.



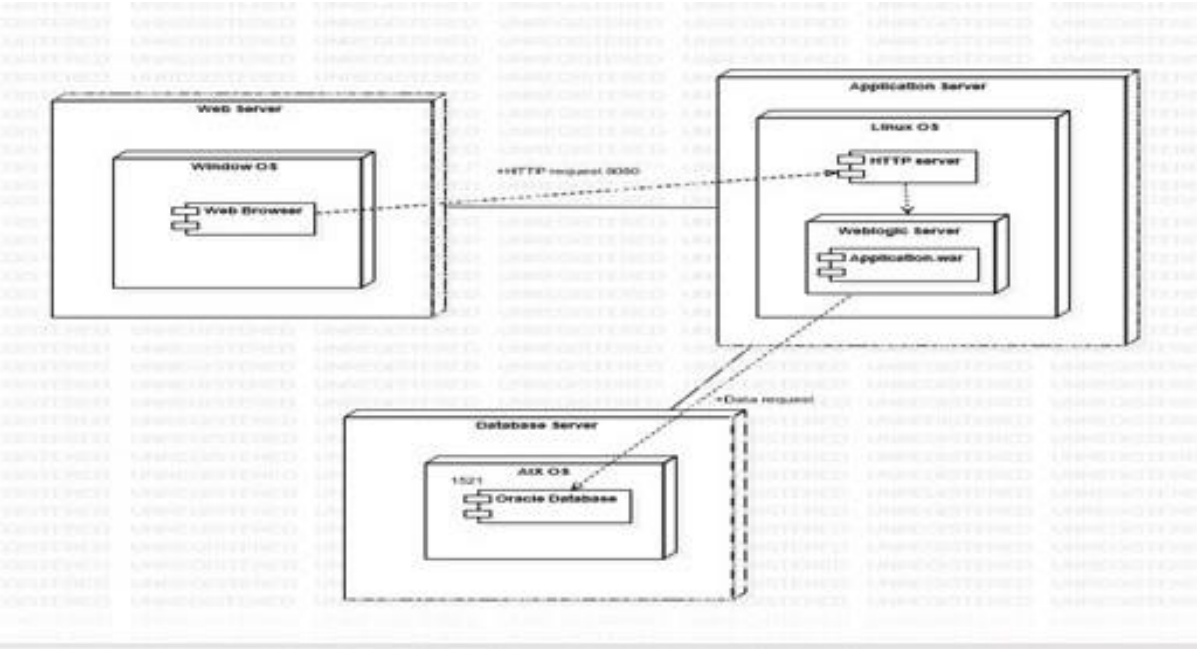
3.7 Component Diagram:

A component diagram is a UML diagram that shows the physical components of a system and their relationships. It illustrates how software components, such as classes, interfaces, and modules, interact to form the system. Component diagrams are used to model the structural organization and dependencies of a system.



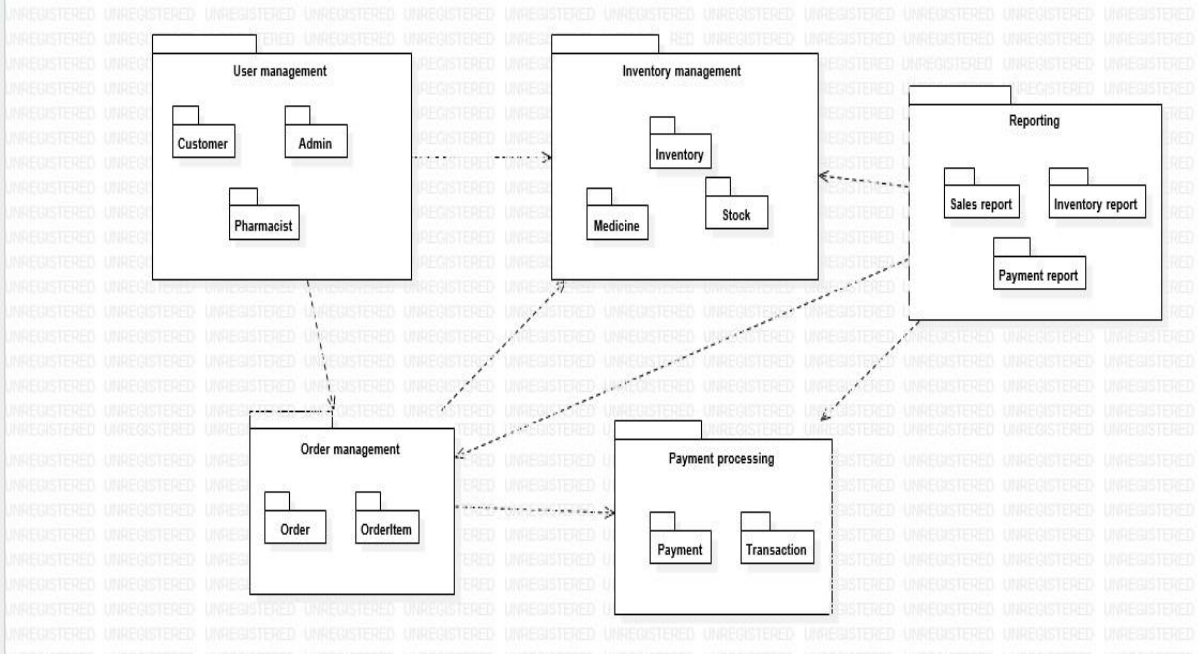
3.8 Deployment Diagram:

A deployment diagram is a UML diagram that shows the physical deployment of software components on hardware nodes. It illustrates the relationships between software artifacts and the hardware infrastructure they run on. Deployment diagrams are used to model the system's hardware and software architecture.



3.9 Package Diagram:

A package diagram is a UML diagram that organizes and groups related classes, components, or other elements into packages. It shows the dependencies and relationships between different packages within a system. Package diagrams are used to model the high-level structure and modularity of a system.



4 CODE

```
#include <mysql.h>
#include <iostream>
#include <string>
#include <iomanip>
#include <sstream>
#include <ctime>
#ifdef _WIN32
#include <conio.h>
#else
#include <termios.h>
#include <unistd.h>
#endif
using namespace std;
string getPassword() {
    string password;
#ifdef _WIN32
    char ch;
```

```
    while ((ch = _getch()) != '\r')
{
        if (ch == '\b') {  // Handle
backspace
            if (!password.empty()) {
                cout << "\b \b";
                password =
password.substr(0, password.size() -
1);
            }
        } else {
            cout << '*';
            password += ch;
        }
    }
#else

    struct termios oldt, newt;
```

```
    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~ECHO;
    tcsetattr(STDIN_FILENO, TCSANOW,
&newt);

    getline(cin, password);

    tcsetattr(STDIN_FILENO, TCSANOW,
&oldt);
#endif
    cout << endl;
    return password;
}
class User {
protected:
    string username;
    string password;
public:
    User(string uname, string pass)
: username(uname), password(pass) {}
    virtual bool login(MYSQL* conn)
= 0;
    virtual void
displayActions(MYSQL* conn) = 0;
     void viewInventory(MYSQL* conn)
{
        cout << "Displaying
Inventory...\n";
        string query = "SELECT
MED_ID, MED_NAME, MED_QTY, CATEGORY,
MED_PRICE, LOCATION_RACK FROM meds";
        if (mysql_query(conn,
query.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            if (res) {
                MYSQL_ROW row;
                cout << left <<
setw(10) << "ID" << setw(20) <<
"Name"
                     << setw(10) <<
"Quantity" << setw(15) << "Category"
                     << setw(10) <<
"Price" << setw(15) << "Location\n";
                cout << string(80,
'-') << endl;
```

```
                while ((row =
mysql_fetch_row(res))) {
                    cout << left <<
setw(10) << row[0] << setw(20) <<
row[1]
                         << setw(10)
<< row[2] << setw(15) << row[3]
                         << setw(10)
<< row[4] << setw(15) << row[5] <<
endl;
                }
                mysql_free_result(re
s);
            } else {
                cerr << "Failed to
fetch inventory data: " <<
mysql_error(conn) << endl;
            }
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        }
    }
    void showCustomersMenu(MYSQL*
conn) {
        int customerChoice;
        do {
            cout << "\n---Customers
Management ---\n";
            cout << "1. View
Customers\n";
            cout << "2. Add
Customers\n";
            cout << "3. Update
Customers\n";
            cout << "4. Delete
Customers\n";
            cout << "5. Back to
Admin Dashboard\n";
            cout << "Enter your
choice: ";
            cin >> customerChoice;
            switch (customerChoice)
{
                case 1:
viewCustomer(conn); break;
```

```
                case 2:
addCustomer(conn); break;
                case 3:
updateCustomer(conn); break;
                case 4:
deleteCustomer(conn); break;
                case 5: return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (customerChoice !=
5);
    }
    void viewCustomer(MYSQL* conn) {
    cout << "Displaying
Customers...\n";
    string query = "SELECT C_ID,
C_FNAME, C_LNAME, C_AGE, C_SEX,
C_PHNO, C_MAIL FROM customer";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        if (res) {
            MYSQL_ROW row;
            cout << left << setw(10)
<< "ID" << setw(15) << "First Name"
                 << setw(15) <<
"Last Name" << setw(5) << "Age"
                 << setw(10) <<
"Sex" << setw(15) << "Phone"
                 << setw(30) <<
"Email\n";
            cout << string(90, '-')
<< endl;

            while ((row =
mysql_fetch_row(res))) {
                cout << left <<
setw(10) << row[0] << setw(15) <<
row[1]
                     << setw(15) <<
row[2] << setw(5) << row[3]
                     << setw(10) <<
row[4] << setw(15) << row[5]
                     << setw(30) <<
row[6] << endl;
            }
            mysql_free_result(res);
        } else {
            cerr << "Failed to fetch
customer data: " <<
mysql_error(conn) << endl;
        }
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}
void addCustomer(MYSQL* conn) {
    string custId, firstName,
lastName, age, sex, phone, email,
username, password;
    cout << "Enter Customer ID: ";
    cin >> custId;
    string checkQuery = "SELECT
COUNT(*) FROM customer WHERE C_ID =
'" + custId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) > 0)
{
            cerr << "Error: Customer
ID already exists. Cannot add
duplicate customer ID.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter First Name: ";
    cin >> firstName;
    cout << "Enter Last Name: ";
```

```
    cin >> lastName;
    cout << "Enter Age: ";
    cin >> age;
    cout << "Enter Sex: ";
    cin >> sex;
    cout << "Enter Phone Number: ";
    cin >> phone;
    cout << "Enter Email: ";
    cin >> email;
    cout << "Enter Username: ";
    cin >> username;
    cout << "Enter Password: ";
    cin >> password;
    string insertQuery = "INSERT
INTO customer (C_ID, C_FNAME,
C_LNAME, C_AGE, C_SEX, C_PHNO,
C_MAIL, C_USERNAME, C_PASSWORD)
VALUES ('"
                         + custId +
"', '" + firstName + "', '" +
lastName + "', '" + age + "', '" +
sex + "', '" + phone + "', '" +
email + "', '" + username + "', '" +
password + "')";
    if (mysql_query(conn,
insertQuery.c_str()) == 0) {
        cout << "Customer added
successfully!\n";
    } else {
        cerr << "Failed to add
customer: " << mysql_error(conn) <<
endl;
    }
}
void updateCustomer(MYSQL* conn) {
    string custId, firstName,
lastName, age, sex, phone, email;
    cout << "Enter Customer ID to
Update: ";
    cin >> custId;
    string checkQuery = "SELECT
COUNT(*) FROM customer WHERE C_ID =
'" + custId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Customer
ID does not exist. Cannot update a
non-existing customer.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter New First Name:
";
    cin >> firstName;
    cout << "Enter New Last Name: ";
    cin >> lastName;
    cout << "Enter New Age: ";
    cin >> age;
    cout << "Enter New Sex: ";
    cin >> sex;
    cout << "Enter New Phone Number:
";
    cin >> phone;
    cout << "Enter New Email: ";
    cin >> email;
    string updateQuery = "UPDATE
customer SET C_FNAME = '" +
firstName + "', C_LNAME = '" +
lastName + "', C_AGE = '" + age +
"', C_SEX = '" + sex + "', C_PHNO =
'" + phone + "', C_MAIL = '" + email
+ "' WHERE C_ID = '" + custId + "'";
    if (mysql_query(conn,
updateQuery.c_str()) == 0) {
        cout << "Customer updated
successfully!\n";
    } else {
        cerr << "Failed to update
customer: " << mysql_error(conn) <<
endl;
    }
}
```

```
void deleteCustomer(MYSQL* conn) {
    string custId;
    cout << "Enter Customer ID to
Delete: ";
    cin >> custId;
    string checkQuery = "SELECT
COUNT(*) FROM customer WHERE C_ID =
'" + custId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Customer
ID does not exist. Cannot delete a
non-existing customer.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    string deleteQuery = "DELETE
FROM customer WHERE C_ID = '" +
custId + "'";
    if (mysql_query(conn,
deleteQuery.c_str()) == 0) {
        cout << "Customer deleted
successfully!\n";
    } else {
        cerr << "Failed to delete
customer: " << mysql_error(conn) <<
endl;
    }
}
    void logout() {
        cout << "Logging out...\n";
    }
};
class Admin : public User {
public:
```

```
    Admin(string uname, string pass)
: User(uname, pass) {}
    bool login(MYSQL* conn) override
{
        string query = "SELECT
COUNT(*) FROM admin WHERE A_USERNAME
= '" + username + "' AND A_PASSWORD
= '" + password + "'";
        if (mysql_query(conn,
query.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            MYSQL_ROW row =
mysql_fetch_row(res);

            bool loginSuccess = row
&& atoi(row[0]) > 0;
            mysql_free_result(res);
            return loginSuccess;
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
            return false;
        }
    }
    void displayActions(MYSQL* conn)
override {
        int choice;
        do {
            cout << "\n--- Admin
Dashboard ---\n";
            cout << "1.
Inventory\n";
            cout << "2.
Suppliers\n";
            cout << "3.
Employees\n";
            cout << "4.
Customers\n";
            cout << "5. View Sales
Invoice Details\n";
            cout << "6. Medicines -
Soon to Expire\n";
            cout << "7. Medicines -
Low Stock\n";
            cout << "8. Logout\n";
```

```
            cout << "Enter your
choice: ";
            cin >> choice;
            switch (choice) {
                case 1:
                    showInventoryMen
u(conn);
                    break;
                case 2:
                    showSuppliersMen
u(conn);
                    break;
                case 3:
showEmployeesMenu(conn); break;
                case 4:
showCustomersMenu(conn); break;
                case 5:
showSalesInvoice(conn); break;
                case 6:
medSoonToExpire(conn); break;
                case 7:
medLowStock(conn); break;
                case 8:
                    logout();
                    return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (choice != 8);
    }
private:
    void showInventoryMenu(MYSQL*
conn) {
        int inventoryChoice;
        do {
            cout << "\n--- Inventory
Management ---\n";
            cout << "1. View
Inventory\n";
            cout << "2. Add
Medicines\n";
            cout << "3. Update
Inventory\n";
            cout << "4. Delete
Medicines\n";
            cout << "5. Back to
Admin Dashboard\n";
            cout << "Enter your
choice: ";
            cin >> inventoryChoice;
            switch (inventoryChoice)
{
                case 1:
viewInventory(conn); break;
                case 2:
addMedicine(conn); break;
                case 3:
updateInventory(conn); break;
                case 4:
deleteMedicine(conn); break;
                case 5: return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (inventoryChoice !=
5);
    }
    void showSuppliersMenu(MYSQL*
conn) {
        int supplierChoice;
        do {
            cout << "\n--- Supplier
Management ---\n";
            cout << "1. View
Suppliers\n";
            cout << "2. Add
Supplier\n";
            cout << "3. Update
Supplier\n";
            cout << "4. Delete
Supplier\n";
            cout << "5. Back to
Admin Dashboard\n";
            cout << "Enter your
choice: ";
            cin >> supplierChoice;
            switch (supplierChoice)
{
                case 1:
viewSuppliers(conn); break;
```

```
                case 2:
addSupplier(conn); break;
                case 3:
updateSupplier(conn); break;
                case 4:
deleteSupplier(conn); break;
                case 5: return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (supplierChoice !=
5);
    }
    void showEmployeesMenu(MYSQL*
conn) {
        int employeeChoice;
        do {
            cout << "\n---Employee
Management ---\n";
            cout << "1. View
Employees\n";
            cout << "2. Add
Employees\n";
            cout << "3. Update
Employees\n";
            cout << "4. Delete
Employees\n";
            cout << "5. Back to
Admin Dashboard\n";
            cout << "Enter your
choice: ";
            cin >> employeeChoice;
            switch (employeeChoice)
{
                case 1:
viewEmployee(conn); break;
                case 2:
addEmployee(conn); break;
                case 3:
updateEmployee(conn); break;
                case 4:
deleteEmployee(conn); break;
                case 5: return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (employeeChoice !=
5);
    }
    void addMedicine(MYSQL* conn) {
        string medId, medName,
category, locationRack;
        int medQty;
        double medPrice;

        while (true) {
            cout << "Enter Medicine
ID: ";
            cin >> medId;
            string checkQuery =
"SELECT COUNT(*) FROM meds WHERE
MED_ID = '" + medId + "'";
            if (mysql_query(conn,
checkQuery.c_str()) == 0) {
                MYSQL_RES* res =
mysql_store_result(conn);
                MYSQL_ROW row =
mysql_fetch_row(res);
                if (atoi(row[0]) >
0) {
                    cout <<
"Medicine ID already exists. Please
enter a unique ID.\n";
                    mysql_free_resul
t(res);
                    continue;
                }
                mysql_free_result(re
s);
                break;
            } else {
                cerr << "Query
Execution Error: " <<
mysql_error(conn) << endl;
                return;
            }
        }
        cout << "Enter Medicine
Name: ";
        cin.ignore();
        getline(cin, medName);
        cout << "Enter Category: ";
```

```
        getline(cin, category);
        cout << "Enter Quantity: ";
        cin >> medQty;
        cout << "Enter Price: ";
        cin >> medPrice;
        cout << "Enter Location
Rack: ";
        cin.ignore();
        getline(cin,
locationRack);
        ostringstream qtyStream,
priceStream;
        qtyStream << medQty;
        priceStream << medPrice;
        string insertQuery = "INSERT
INTO meds (MED_ID, MED_NAME,
MED_QTY, CATEGORY, MED_PRICE,
LOCATION_RACK) VALUES ('" +
                             medId +
"', '" + medName + "', " +
qtyStream.str() + ", '" + category +
"', " + priceStream.str() + ", '" +
locationRack + "')";
        if (mysql_query(conn,
insertQuery.c_str()) == 0) {
            cout << "Medicine added
successfully!\n";
        } else {
            cerr << "Failed to add
medicine: " << mysql_error(conn) <<
endl;
        }
    }
    void updateInventory(MYSQL*
conn) {
        string medId, newMedName,
newCategory, newLocationRack;
        int newMedQty;
        double newMedPrice;
        cout << "Enter Medicine ID
to update: ";
        cin >> medId;
        string fetchQuery = "SELECT
MED_NAME, MED_QTY, CATEGORY,
MED_PRICE, LOCATION_RACK FROM meds
WHERE MED_ID = '" + medId + "'";
        if (mysql_query(conn,
fetchQuery.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            MYSQL_ROW row =
mysql_fetch_row(res);
            if (row) {
                cout << "Current
details of " << medId << ":\n";
                cout << "Name: " <<
row[0] << "\n";
                cout << "Quantity: "
<< row[1] << "\n";
                cout << "Category: "
<< row[2] << "\n";
                cout << "Price: " <<
row[3] << "\n";
                cout << "Location
Rack: " << row[4] << "\n";
                mysql_free_result(re
s);
                cout << "Enter new
Medicine Name: ";
                cin.ignore();
                getline(cin,
newMedName);
                if
(newMedName.empty()) newMedName =
row[0];
                cout << "Enter new
Quantity: ";
                cin >> newMedQty;
                if (newMedQty == 0)
newMedQty = atoi(row[1]);
                cout << "Enter new
Category: ";
                cin.ignore();
                getline(cin,
newCategory);
                if
(newCategory.empty()) newCategory =
row[2];
                cout << "Enter new
Price: ";
                cin >> newMedPrice;
                if (newMedPrice ==
0) newMedPrice = atof(row[3]);
                cout << "Enter new
Location Rack: ";
                cin.ignore();
```

```
                getline(cin,
newLocationRack);
                if
(newLocationRack.empty())
newLocationRack = row[4];
                ostringstream
qtyStream, priceStream;
                qtyStream <<
newMedQty;
                priceStream <<
newMedPrice;
                string updateQuery =
"UPDATE meds SET MED_NAME = '" +
newMedName + "', MED_QTY = " +
qtyStream.str() +

 ", CATEGORY = '" + newCategory +
"', MED_PRICE = " +
priceStream.str() +

 ", LOCATION_RACK = '" +
newLocationRack + "' WHERE MED_ID =
'" + medId + "'";
                if
(mysql_query(conn,
updateQuery.c_str()) == 0) {
                    cout <<
"Medicine updated successfully!\n";
                } else {
                    cerr << "Failed
to update medicine: " <<
mysql_error(conn) << endl;
                }
            } else {
                cerr << "Medicine
not found with ID: " << medId <<
endl;
                mysql_free_result(re
s);
            }
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        }
    }
    void deleteMedicine(MYSQL* conn)
{
        string medId;
        while (true) {
            cout << "Enter Medicine
ID to delete: ";
            cin >> medId;
            string checkQuery =
"SELECT COUNT(*) FROM meds WHERE
MED_ID = '" + medId + "'";
            if (mysql_query(conn,
checkQuery.c_str()) == 0) {
                MYSQL_RES* res =
mysql_store_result(conn);
                MYSQL_ROW row =
mysql_fetch_row(res);
                if (row &&
atoi(row[0]) > 0) {
                    string
deleteQuery = "DELETE FROM meds
WHERE MED_ID = '" + medId + "'";
                    if
(mysql_query(conn,
deleteQuery.c_str()) == 0) {
                        cout <<
"Medicine with ID " << medId << "
has been deleted successfully!\n";
                        mysql_free_r
esult(res);
                        break;
                    } else {
                        cerr <<
"Error deleting medicine: " <<
mysql_error(conn) << endl;
                        mysql_free_r
esult(res);
                        break;
                    }
                } else {
                    cout <<
"Medicine with ID " << medId << "
does not exist. Please enter a valid
ID.\n";
                    mysql_free_resul
t(res);
                }
            } else {
                cerr << "Query
Execution Error: " <<
mysql_error(conn) << endl;
```

```
                break;
            }
        }
    }
    void viewSuppliers(MYSQL* conn)
{
        cout << "Displaying
Suppliers...\n";
        string query = "SELECT
SUP_ID, SUP_NAME, SUP_ADD, SUP_PHNO,
SUP_MAIL FROM suppliers";
        if (mysql_query(conn,
query.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            if (res) {
                MYSQL_ROW row;
                cout << left <<
setw(10) << "ID" << setw(20) <<
"Name"
                     << setw(30) <<
"Address" << setw(15) << "Phone"
                     << setw(30) <<
"Email\n";
                cout << string(80,
'-') << endl;

                while ((row =
mysql_fetch_row(res))) {
                    cout << left <<
setw(10) << row[0] << setw(20) <<
row[1]
                         << setw(30)
<< row[2] << setw(15) << row[3]
                         << setw(30)
<< row[4] << endl;
                }
                mysql_free_result(re
s);
            } else {
                cerr << "Failed to
fetch supplier data: " <<
mysql_error(conn) << endl;
            }
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        }
    }
void addSupplier(MYSQL* conn) {
    string supId, supName,
supAddress, supPhone, supEmail;

    cout << "Enter Supplier ID: ";
    cin >> supId;
    string checkQuery = "SELECT
COUNT(*) FROM suppliers WHERE SUP_ID
= '" + supId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) > 0)
{
            cerr << "Error: Supplier
ID already exists. Cannot add
duplicate supplier ID.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter Supplier Name: ";
    cin.ignore();
    getline(cin, supName);
    cout << "Enter Supplier Address:
";
    getline(cin, supAddress);
    cout << "Enter Supplier Phone:
";
    getline(cin, supPhone);
    cout << "Enter Supplier Email:
";
    getline(cin, supEmail);

    string insertQuery = "INSERT
INTO suppliers (SUP_ID, SUP_NAME,
SUP_ADD, SUP_PHNO, SUP_MAIL) VALUES
```

```
('" + supId + "', '" + supName + "',
'" + supAddress + "', '" + supPhone
+ "', '" + supEmail + "')";
    if (mysql_query(conn,
insertQuery.c_str()) == 0) {
        cout << "Supplier added
successfully!\n";
    } else {
        cerr << "Failed to add
supplier: " << mysql_error(conn) <<
endl;
    }
}
void updateSupplier(MYSQL* conn) {
    string supId, supName,
supAddress, supPhone, supEmail;
    cout << "Enter Supplier ID to
Update: ";
    cin >> supId;
    string checkQuery = "SELECT
COUNT(*) FROM suppliers WHERE SUP_ID
= '" + supId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Supplier
ID does not exist. Cannot update a
non-existing supplier.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter New Supplier
Name: ";
    cin.ignore();
    getline(cin, supName);
    cout << "Enter New Supplier
Address: ";
    getline(cin, supAddress);
    cout << "Enter New Supplier
Phone: ";
    getline(cin, supPhone);
    cout << "Enter New Supplier
Email: ";
    getline(cin, supEmail);
    string updateQuery = "UPDATE
suppliers SET SUP_NAME = '" +
supName + "', SUP_ADD = '" +
supAddress + "', SUP_PHNO = '" +
supPhone + "', SUP_MAIL = '" +
supEmail + "' WHERE SUP_ID = '" +
supId + "'";
    if (mysql_query(conn,
updateQuery.c_str()) == 0) {
        cout << "Supplier updated
successfully!\n";
    } else {
        cerr << "Failed to update
supplier: " << mysql_error(conn) <<
endl;
    }
}
void deleteSupplier(MYSQL* conn) {
    string supId;
    cout << "Enter Supplier ID to
Delete: ";
    cin >> supId;
    string checkQuery = "SELECT
COUNT(*) FROM suppliers WHERE SUP_ID
= '" + supId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Supplier
ID does not exist. Cannot delete a
non-existing supplier.\n";
            mysql_free_result(res);
            return;
        }
```

```
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    string deleteQuery = "DELETE
FROM suppliers WHERE SUP_ID = '" +
supId + "'";
    if (mysql_query(conn,
deleteQuery.c_str()) == 0) {
        cout << "Supplier deleted
successfully!\n";
    } else {
        cerr << "Failed to delete
supplier: " << mysql_error(conn) <<
endl;
    }
}
void viewEmployee(MYSQL* conn) {
    cout << "Displaying
Employees...\n";
    string query = "SELECT E_ID,
E_FNAME, E_LNAME, BDATE, E_AGE,
E_SEX, E_TYPE, E_JDATE, E_SAL,
E_PHNO, E_MAIL FROM employee";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        if (res) {
            MYSQL_ROW row;
            cout << left << setw(10)
<< "ID" << setw(20) << "First Name"
<< setw(20) << "Last Name"
                 << setw(12) <<
"Birth Date" << setw(5) << "Age" <<
setw(10) << "Sex" << setw(15) <<
"Job Type"
                 << setw(15) <<
"Join Date" << setw(10) << "Salary"
<< setw(15) << "Phone" << setw(30)
<< "Email\n";
            cout << string(150, '-')
<< endl;
            while ((row =
mysql_fetch_row(res))) {
                cout << left <<
setw(10) << row[0] << setw(20) <<
row[1] << setw(20) << row[2]
                     << setw(12) <<
row[3] << setw(5) << row[4] <<
setw(10) << row[5] << setw(15) <<
row[6]
                     << setw(15) <<
row[7] << setw(10) << row[8] <<
setw(15) << row[9] << setw(30) <<
row[10] << endl;
            }
            mysql_free_result(res);
        } else {
            cerr << "Failed to fetch
employee data: " <<
mysql_error(conn) << endl;
        }
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}
void addEmployee(MYSQL* conn) {
    string empId, firstName,
lastName, birthDate, age, sex,
jobType, joinDate, salary, phone,
email, address;
    cout << "Enter Employee ID: ";
    cin >> empId;
    string checkQuery = "SELECT
COUNT(*) FROM employee WHERE E_ID =
'" + empId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) > 0)
{
            cerr << "Error: Employee
ID already exists. Cannot add
duplicate employee ID.\n";
            mysql_free_result(res);
```

```
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter First Name: ";
    cin.ignore();
    getline(cin, firstName);
    cout << "Enter Last Name: ";
    getline(cin, lastName);
    cout << "Enter Birth Date (YYYY-
MM-DD): ";
    getline(cin, birthDate);
    cout << "Enter Age: ";
    getline(cin, age);
    cout << "Enter Sex: ";
    getline(cin, sex);
    cout << "Enter Job Type: ";
    getline(cin, jobType);
    cout << "Enter Join Date (YYYY-
MM-DD): ";
    getline(cin, joinDate);
    cout << "Enter Salary: ";
    getline(cin, salary);
    cout << "Enter Phone Number: ";
    getline(cin, phone);
    cout << "Enter Email: ";
    getline(cin, email);
    cout << "Enter Address: ";
    getline(cin, address);
    string insertQuery = "INSERT
INTO employee (E_ID, E_FNAME,
E_LNAME, BDATE, E_AGE, E_SEX,
E_TYPE, E_JDATE, E_SAL, E_PHNO,
E_MAIL, E_ADD) "
                         "VALUES ('"
+ empId + "', '" + firstName + "',
'" + lastName + "', '" + birthDate +
"', '" + age + "', '"
                         + sex + "',
'" + jobType + "', '" + joinDate +
"', '" + salary + "', '" + phone +
"', '" + email + "', '" + address +
"')";
    if (mysql_query(conn,
insertQuery.c_str()) == 0) {
        cout << "Employee added
successfully!\n";
    } else {
        cerr << "Failed to add
employee: " << mysql_error(conn) <<
endl;
    }
}
void updateEmployee(MYSQL* conn) {
    string empId, firstName,
lastName, birthDate, age, sex,
jobType, joinDate, salary, phone,
email, address;
    cout << "Enter Employee ID to
Update: ";
    cin >> empId;
    string checkQuery = "SELECT
COUNT(*) FROM employee WHERE E_ID =
'" + empId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Employee
ID does not exist. Cannot update
non-existing employee.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter New First Name:
";
    cin.ignore();
    getline(cin, firstName);
    cout << "Enter New Last Name: ";
    getline(cin, lastName);
```

```
    cout << "Enter New Birth Date
(YYYY-MM-DD): ";
    getline(cin, birthDate);
    cout << "Enter New Age: ";
    getline(cin, age);
    cout << "Enter New Sex: ";
    getline(cin, sex);
    cout << "Enter New Job Type: ";
    getline(cin, jobType);
    cout << "Enter New Join Date
(YYYY-MM-DD): ";
    getline(cin, joinDate);
    cout << "Enter New Salary: ";
    getline(cin, salary);
    cout << "Enter New Phone Number:
";
    getline(cin, phone);
    cout << "Enter New Email: ";
    getline(cin, email);
    cout << "Enter New Address: ";
    getline(cin, address);
    string updateQuery = "UPDATE
employee SET E_FNAME = '" +
firstName + "', E_LNAME = '" +
lastName + "', BDATE = '" +
birthDate + "', E_AGE = '"
                         + age + "',
E_SEX = '" + sex + "', E_TYPE = '" +
jobType + "', E_JDATE = '" +
joinDate + "', E_SAL = '"
                         + salary +
"', E_PHNO = '" + phone + "', E_MAIL
= '" + email + "', E_ADD = '" +
address + "' WHERE E_ID = '"
                         + empId +
"'";

    if (mysql_query(conn,
updateQuery.c_str()) == 0) {
        cout << "Employee updated
successfully!\n";
    } else {
        cerr << "Failed to update
employee: " << mysql_error(conn) <<
endl;
    }
}
void deleteEmployee(MYSQL* conn) {
    string empId;
    cout << "Enter Employee ID to
Delete: ";
    cin >> empId;
    string checkQuery = "SELECT
COUNT(*) FROM employee WHERE E_ID =
'" + empId + "'";
    if (mysql_query(conn,
checkQuery.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row && atoi(row[0]) ==
0) {
            cerr << "Error: Employee
ID does not exist. Cannot delete
non-existing employee.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    string deleteQuery = "DELETE
FROM employee WHERE E_ID = '" +
empId + "'";

    if (mysql_query(conn,
deleteQuery.c_str()) == 0) {
        cout << "Employee deleted
successfully!\n";
    } else {
        cerr << "Failed to delete
employee: " << mysql_error(conn) <<
endl;
    }
}

void showSalesInvoice(MYSQL* conn) {
    std::cout << "Displaying Sales
Invoices...\n";
```

```
    std::string query = "SELECT
SALE_ID, C_ID, S_DATE, S_TIME,
TOTAL_AMT, E_ID FROM sales";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        if (res) {
            MYSQL_ROW row;
            std::cout << std::left
<< std::setw(10) << "Sale ID "
                      <<
std::setw(10) << "Customer_ID "
                      <<
std::setw(15) << "Sale_Date "
                      <<
std::setw(10) << "Sale_Time "
                      <<
std::setw(12) << "Total_Amount "
                      <<
std::setw(10) << "Employee_ID\n";
            std::cout <<
std::string(70, '-') << std::endl;
            while ((row =
mysql_fetch_row(res))) {
                std::cout <<
std::left << std::setw(10) << row[0]
                          <<
std::setw(10) << row[1]
                          <<
std::setw(15) << row[2]
                          <<
std::setw(10) << row[3]
                          <<
std::setw(12) << row[4]
                          <<
std::setw(10) << row[5] <<
std::endl;
            }
            mysql_free_result(res);
        } else {
            std::cerr << "Failed to
fetch sales invoice data: " <<
mysql_error(conn) << std::endl;
        }
    } else {
        std::cerr << "Query
Execution Error: " <<
mysql_error(conn) << std::endl;
    }
}
void medSoonToExpire(MYSQL* conn) {
    std::cout << "Displaying
Medicines Soon to Expire (within 6
months)...\n";
    std::string query = "SELECT
MED_ID, EXP_DATE FROM purchase WHERE
EXP_DATE <= DATE_ADD(CURDATE(),
INTERVAL 6 MONTH)";

    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        if (res) {
            MYSQL_ROW row;
            std::cout << std::left
<< std::setw(10) << "Med ID"
                      <<
std::setw(20) << "Expiry Date\n";
            std::cout <<
std::string(30, '-') << std::endl;

            while ((row =
mysql_fetch_row(res))) {
                std::cout <<
std::left << std::setw(10) << row[0]
                          <<
std::setw(20) << row[1] <<
std::endl;
            }
            mysql_free_result(res);
        } else {
            std::cerr << "Failed to
fetch expiry data: " <<
mysql_error(conn) << std::endl;
        }
    } else {
        std::cerr << "Query
Execution Error: " <<
mysql_error(conn) << std::endl;
    }
}
void medLowStock(MYSQL* conn) {
```

```
    std::cout << "Displaying
Medicines with Low Stock (quantity <
50)...\n";
    std::string query = "SELECT
MED_ID, MED_NAME, MED_QTY FROM meds
WHERE MED_QTY < 50";

    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        if (res) {
            MYSQL_ROW row;
            std::cout << std::left
<< std::setw(10) << "Med ID"
                      <<
std::setw(20) << "Medicine Name"
                      <<
std::setw(10) << "Quantity\n";
            std::cout <<
std::string(40, '-') << std::endl;

            while ((row =
mysql_fetch_row(res))) {
                std::cout <<
std::left << std::setw(10) << row[0]
                          <<
std::setw(20) << row[1]
                          <<
std::setw(10) << row[2] <<
std::endl;
            }
            mysql_free_result(res);
        } else {
            std::cerr << "Failed to
fetch low stock data: " <<
mysql_error(conn) << std::endl;
        }
    } else {
        std::cerr << "Query
Execution Error: " <<
mysql_error(conn) << std::endl;
    }
}

};
class Pharmacist : public User {
private:
    int employeeId;
public:
    Pharmacist(string uname, string
pass) : User(uname, pass),
employeeId(0) {}
    bool login(MYSQL* conn) override
{
        string query = "SELECT E_ID
FROM emplogin WHERE E_USERNAME = '"
+ username + "' AND E_PASS = '" +
password + "'";
        if (mysql_query(conn,
query.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            MYSQL_ROW row =
mysql_fetch_row(res);
            if (row) {
                employeeId =
atoi(row[0]);
                mysql_free_result(re
s);
                return true;
            }
            mysql_free_result(res);
            return false;
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
            return false;
        }
    }
    int getEmployeeId() const {
        return employeeId;
    }
    void displayActions(MYSQL* conn)
override{
        int choice;
        do {
            cout << "\n---
Pharmacist Dashboard ---\n";
            cout << "1. View
Inventory\n";
            cout << "2. Add
Sales\n";
            cout << "3. Customer
Details\n";
```

```
            cout << "4. Logout\n";
            cout << "Enter your
choice: ";
            cin >> choice;
            switch (choice) {
                case 1:
                    viewInventory(co
nn);
                    break;
                case 2:
                    addSales(conn);
                    break;
                case 3:
                    showCustomersMen
u(conn);
                    break;
                case 4:
                    logout();
                    return;
                default:
                    cout << "Invalid
choice. Please try again.\n";
                    break;
            }
        } while (choice != 4);
    }
void addSales(MYSQL* conn) {
    cout << "Enter Customer ID: ";
    int customerId;
    cin >> customerId;
    ostringstream oss;
    oss << customerId;
    string customerIdStr =
oss.str();
    string query = "SELECT * FROM
customer WHERE C_ID = '" +
customerIdStr + "'";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (!row) {
            cout << "Customer not
found.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter Medicine Name: ";
    string medName;
    cin >> ws;
    getline(cin, medName);
    query = "SELECT MED_ID,
MED_NAME, MED_QTY, MED_PRICE,
CATEGORY FROM meds WHERE MED_NAME =
'" + medName + "'";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (!row) {
            cout << "Medicine not
found.\n";
            mysql_free_result(res);
            return;
        }
        string medId = row[0];
        string medName = row[1];
        int availableQty =
atoi(row[2]);
        double price = atof(row[3]);
        string category =
row[4];
        cout << "Medicine
Details:\n";
        cout << "Name: " << medName
<< "\n";
        cout << "Category: " <<
category << "\n";
        cout << "Available Quantity:
" << availableQty << "\n";
        cout << "Price per Unit: "
<< fixed << setprecision(2) << price
<< "\n";
        mysql_free_result(res);
```

```
        cout << "Enter quantity
required: ";
        int qty;
        cin >> qty;
        if (qty > availableQty) {
            cout << "Not enough
stock available.\n";
            return;
        }
        double totalPrice = qty *
price;
        cout << "Total Price: " <<
fixed << setprecision(2) <<
totalPrice << "\n";
        time_t now = time(0);
        struct tm* currentTime =
localtime(&now);
        char date[20], time[10];
        strftime(date, sizeof(date),
"%Y-%m-%d", currentTime);
        strftime(time, sizeof(time),
"%H:%M:%S", currentTime);
        ostringstream
totalPriceStream;
        totalPriceStream <<
totalPrice;
        string totalPriceStr =
totalPriceStream.str();
        ostringstream
employeeIdStream;
        employeeIdStream <<
getEmployeeId();
        string employeeIdStr =
employeeIdStream.str();
        query = "INSERT INTO sales
(C_ID, S_DATE, S_TIME, TOTAL_AMT,
E_ID) VALUES ('" + customerIdStr +
"', '" + string(date) + "', '" +
string(time) + "', '" +
totalPriceStr + "', '" +
employeeIdStr + "')";
        if (mysql_query(conn,
query.c_str()) == 0) {
            cout << "Sale recorded
successfully.\n";
        } else {
            cerr << "Error recording
sale: " << mysql_error(conn) <<
endl;
            return;
        }
        int saleId =
mysql_insert_id(conn);
        oss.str(""); oss <<
saleId;
        string saleIdStr =
oss.str();
        oss.str(""); oss << qty;
        string qtyStr =
oss.str();
        oss.str(""); oss <<
totalPrice;
        string totalPriceItemStr =
oss.str();
        query = "INSERT INTO
sales_items (SALE_ID, MED_ID,
SALE_QTY, TOT_PRICE) VALUES ('" +
saleIdStr + "', '" + medId + "', '"
+ qtyStr + "', '" +
totalPriceItemStr + "')";
        if (mysql_query(conn,
query.c_str()) == 0) {
            cout << "Sale item added
successfully.\n";
        } else {
            cerr << "Error adding
sale item: " << mysql_error(conn) <<
endl;
            return;
        }
        query = "UPDATE meds SET
MED_QTY = MED_QTY - " + qtyStr + "
WHERE MED_ID = '" + medId + "'";
        if (mysql_query(conn,
query.c_str()) == 0) {
            cout << "Medicine stock
updated.\n";
        } else {
            cerr << "Error updating
medicine stock: " <<
mysql_error(conn) << endl;
            return;
        }
    } else {
```

```
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}
};
class Customer : public User {
private:
    int customerId;
public:
    Customer(string uname, string
pass) : User(uname, pass),
customerId(-1) {}
    int getCustomerId() {
        return customerId;
    }
    bool login(MYSQL* conn) override
{
        string query = "SELECT C_ID
FROM customer WHERE C_USERNAME = '"
+ username + "' AND C_PASSWORD = '"
+ password + "'";
        if (mysql_query(conn,
query.c_str()) == 0) {
            MYSQL_RES* res =
mysql_store_result(conn);
            MYSQL_ROW row =
mysql_fetch_row(res);
            if (row) {
                customerId =
atoi(row[0]);
                mysql_free_result(re
s);
                return true;
            } else {
                mysql_free_result(re
s);
                return false;
            }
        } else {
            cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
            return false;
        }
    }
    void displayActions(MYSQL* conn)
override {
```

```
    int choice;
    do {
        cout << "\n--- Customer
Dashboard ---\n";
        cout << "1. Profile\n";
        cout << "2. Make Order\n";
        cout << "3. Order
Histories\n";
        cout << "4. Logout\n";
        cout << "Enter your choice:
";
        cin >> choice;

        switch (choice) {
            case 1:
                profile(conn);
                break;
            case 2:
                makeOrder(conn);
                break;
            case 3:
                orderHistory(conn);
                break;
            case 4:
                logout();
                return;
            default:
                cout << "Invalid
choice. Please try again.\n";
                break;
        }
    } while (choice != 4);
}
void profile(MYSQL* conn) {
    string query = "SELECT C_FNAME,
C_LNAME, C_AGE, C_SEX, C_PHNO,
C_MAIL FROM customer WHERE
C_USERNAME = '" + username + "'";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (row) {
            cout << "\n--- Customer
Profile ---\n";
```

```
            cout << "First Name: "
<< row[0] << endl;
            cout << "Last Name: " <<
row[1] << endl;
            cout << "Age: " <<
row[2] << endl;
            cout << "Sex: " <<
row[3] << endl;
            cout << "Phone Number: "
<< row[4] << endl;
            cout << "Email: " <<
(row[5] ? row[5] : "N/A") << endl;
        } else {
            cout << "No customer
details found.\n";
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}
void makeOrder(MYSQL* conn) {
    ostringstream customerIdStream;
    customerIdStream <<
getCustomerId();
    string customerIdStr =
customerIdStream.str();
    string query = "SELECT * FROM
customer WHERE C_ID = '" +
customerIdStr + "'";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (!row) {
            cout << "Customer not
found.\n";
            mysql_free_result(res);
            return;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
        return;
    }
    cout << "Enter Medicine Name: ";
    string medName;
    cin >> ws;
    getline(cin, medName);
    query = "SELECT MED_ID,
MED_NAME, MED_QTY, MED_PRICE,
CATEGORY FROM meds WHERE MED_NAME =
'" + medName + "'";
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row =
mysql_fetch_row(res);
        if (!row) {
            cout << "Medicine not
found.\n";
            mysql_free_result(res);
            return;
        }
        string medId = row[0];
        string medName = row[1];
        int availableQty =
atoi(row[2]);
        double price = atof(row[3]);
        string category = row[4];
        cout << "Medicine
Details:\n";
        cout << "Name: " << medName
<< "\n";
        cout << "Category: " <<
category << "\n";
        cout << "Available Quantity:
" << availableQty << "\n";
        cout << "Price per Unit: "
<< fixed << setprecision(2) << price
<< "\n";
        mysql_free_result(res);
        cout << "Enter quantity
required: ";
        int qty;
        cin >> qty;
        if (qty > availableQty) {
```

```
            cout << "Not enough
stock available.\n";
            return;
        }
        double totalPrice = qty *
price;
        cout << "Total Price: " <<
fixed << setprecision(2) <<
totalPrice << "\n";

        // Show options to pay or
cancel
        cout << "Choose an
option:\n1. Pay\n2. Cancel\n";
        int choice;
        cin >> choice;

        if (choice == 2) {
            cout << "Order
cancelled.\n";
            return;
        } else if (choice == 1) {
            // Process payment
            time_t now = time(0);
            struct tm* currentTime =
localtime(&now);
            char date[20], time[10];
            strftime(date,
sizeof(date), "%Y-%m-%d",
currentTime);
            strftime(time,
sizeof(time), "%H:%M:%S",
currentTime);

            ostringstream
totalPriceStream;
            totalPriceStream <<
totalPrice;
            string totalPriceStr =
totalPriceStream.str();
            string employeeIdStr =
"1";
            query = "INSERT INTO
sales (C_ID, S_DATE, S_TIME,
TOTAL_AMT, E_ID) VALUES ('" +
customerIdStr + "', '" +
string(date) + "', '" + string(time)
+ "', '" + totalPriceStr + "', '" +
employeeIdStr + "')";
            if (mysql_query(conn,
query.c_str()) == 0) {
                cout << "Sale
recorded successfully.\n";
            } else {
                cerr << "Error
recording sale: " <<
mysql_error(conn) << endl;
                return;
            }
            int saleId =
mysql_insert_id(conn);
            ostringstream
saleIdStream;
            saleIdStream << saleId;
            string saleIdStr =
saleIdStream.str();
            ostringstream qtyStream;
            qtyStream << qty;
            string qtyStr =
qtyStream.str();
            ostringstream
totalPriceItemStream;
            totalPriceItemStream <<
totalPrice;
            string totalPriceItemStr
= totalPriceItemStream.str();
            query = "INSERT INTO
sales_items (SALE_ID, MED_ID,
SALE_QTY, TOT_PRICE) VALUES ('" +
saleIdStr + "', '" + medId + "', '"
+ qtyStr + "', '" +
totalPriceItemStr + "')";
            if (mysql_query(conn,
query.c_str()) == 0) {
                cout << "Sale item
added successfully.\n";
            } else {
                cerr << "Error
adding sale item: " <<
mysql_error(conn) << endl;
                return;
            }
            query = "UPDATE meds SET
MED_QTY = MED_QTY - " + qtyStr + "
WHERE MED_ID = '" + medId + "'";
```

```
            if (mysql_query(conn,
query.c_str()) == 0) {
                cout << "Medicine
stock updated.\n";
            } else {
                cerr << "Error
updating medicine stock: " <<
mysql_error(conn) << endl;
                return;
            }
            cout << "Payment
successful. Thank you for your
purchase!\n";
        } else {
            cout << "Invalid choice.
Order cancelled.\n";
            return;
        }
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}

void orderHistory(MYSQL* conn) {
    if (customerId == -1) {
        cout << "Customer is not
logged in.\n";
        return;
    }
    ostringstream customerIdStream;
    customerIdStream << customerId;
    string customerIdStr =
customerIdStream.str();
    string query = "SELECT
si.MED_ID, m.MED_NAME, si.SALE_QTY,
si.TOT_PRICE, s.S_DATE "
                   "FROM sales s "
                   "JOIN sales_items
si ON s.SALE_ID = si.SALE_ID "
                   "JOIN meds m ON
si.MED_ID = m.MED_ID "
                   "WHERE s.C_ID =
'" + customerIdStr + "' "
                   "ORDER BY
s.S_DATE DESC";
```

```
    if (mysql_query(conn,
query.c_str()) == 0) {
        MYSQL_RES* res =
mysql_store_result(conn);
        MYSQL_ROW row;
        cout << "\n--- Order History
---\n";
        cout << left << setw(10) <<
"Med ID"
             << left << setw(30) <<
"Medicine Name"
             << left << setw(10) <<
"Quantity"
             << left << setw(15) <<
"Total Price"
             << left << setw(15) <<
"Date" << endl;
        while ((row =
mysql_fetch_row(res))) {
            cout << left << setw(10)
<< row[0]
                 << left << setw(30)
<< row[1]
                 << left << setw(10)
<< row[2]
                 << left << setw(15)
<< fixed << setprecision(2) <<
atof(row[3])
                 << left << setw(15)
<< row[4]
                 << endl;
        }
        mysql_free_result(res);
    } else {
        cerr << "Query Execution
Error: " << mysql_error(conn) <<
endl;
    }
}
};
int main() {
    MYSQL* conn;
    string server = "localhost",
user = "root", password = "pms123",
database = "pharmacy";
    conn = mysql_init(0);
    if (conn) {
```

```
        cout << "MySQL Library
initialized.\n";
    } else {
        cerr << "MySQL
Initialization failed.\n";
        return 1;
    }
    conn = mysql_real_connect(conn,
server.c_str(), user.c_str(),
password.c_str(), database.c_str(),
3306, NULL, 0);
    if (conn) {
        cout << "Connection
established.\n";
    } else {
        cerr << "Connection failed:
" << mysql_error(conn) << endl;
        return 1;
    }
    int roleChoice;
    string username, pass;
    User* userPtr = NULL;
    while (true) {
        cout << "Select login
role:\n1. Admin\n2. Pharmacist\n3.
Customer\nEnter choice: ";
        cin >> roleChoice;
        cout << "Enter username: ";
        cin >> username;
        cout << "Enter password: ";
        pass = getPassword();
        delete userPtr;
        if (roleChoice == 1) {
            userPtr = new
Admin(username, pass);
        } else if (roleChoice == 2)
{
            userPtr = new
Pharmacist(username, pass);
        } else if (roleChoice == 3)
{
            userPtr = new
Customer(username, pass);
        } else {
            cout << "Invalid role
choice. Please try again.\n";
            continue;
        }
        if (userPtr->login(conn)) {
            cout << "Login
successful!\n";
            userPtr-
>displayActions(conn);
            delete userPtr;
            break;
        } else {
            cout << "Invalid
username or password. Please try
again.\n";
            delete userPtr;
            userPtr = NULL;
        }
    }
    mysql_close(conn);
    return 0;
}
```

5 OUTPUT SCREENSHOT

```
MySQL Library initialized.
Connection established.
Select login role:
1. Admin
2. Pharmacist
3. Customer
Enter choice: 1
Enter username: admin
Enter password: ********
Login successful!

--- Admin Dashboard ---
1. Inventory
2. Suppliers
3. Employees
4. Customers
5. View Sales Invoice Details
6. Medicines - Soon to Expire
7. Medicines - Low Stock
8. Logout
Enter your choice: 1
```

```
MySQL Library initialized.
Connection established.
Select login role:
1. Admin
2. Pharmacist
3. Customer
Enter choice: 3
Enter username: safiam
Enter password: ********
Login successful!

--- Customer Dashboard ---
1. Profile
2. Make Order
3. Order Histories
4. Logout
Enter your choice:
```

```
MySQL Library initialized.
Connection established.
Select login role:
1. Admin
2. Pharmacist
3. Customer
Enter choice: 2
Enter username: amaya
Enter password: *****
Login successful!

--- Pharmacist Dashboard --
1. View Inventory
2. Add Sales
3. Customer Details
4. Logout
Enter your choice: |
```

```
--- Pharmacist Dashboard ---
1. View Inventory
2. Add Sales
3. Customer Details
4. Logout
Enter your choice: 2
Enter Customer ID: 987107
Enter Medicine Name: cyclopam
Medicine Details:
Name: Cyclopam
Category: Tablet
Available Quantity: 120
Price per Unit: 6.00
Enter quantity required: 10
Total Price: 60.00
Sale recorded successfully.
Sale item added successfully.
Medicine stock updated.
```

```
--- Inventory Management ---
1. View Inventory
2. Add Medicines
3. Update Inventory
4. Delete Medicines
5. Back to Admin Dashboard
Enter your choice: 1
Displaying Inventory...
```

ID	Name	Quantity	Category	Price	Location
123001	Dolo 650 MG	625	Tablet	1.00	rack 5
123002	Panadol Cold & Flu	90	Tablet	2.50	rack 6
123003	Livogen	15	Capsule	5.00	rack 3
123004	Gelusil	440	Tablet	1.25	rack 4
123005	Cyclopam	120	Tablet	6.00	rack 2
123006	Benadryl 200 ML	35	Syrup	50.00	rack 10
123007	Lopamide	15	Capsule	5.00	rack 7
123008	Vitamic C	90	Tablet	3.00	rack 8
123009	Omeprazole	35	Capsule	4.00	rack 3
123010	Concur 5 MG	600	Tablet	3.50	rack 9
123011	Augmentin 250 ML	115	Syrup	80.00	rack 7
123012	punitha	210	tablet	3.00	4

```
--- Inventory Management ---
1. View Inventory
2. Add Medicines
3. Update Inventory
4. Delete Medicines
5. Back to Admin Dashboard
Enter your choice: 2
Enter Medicine ID: 123013
Enter Medicine Name: Nicogen
Enter Category: Capsule
Enter Quantity: 200
Enter Price: 5
Enter Location Rack: 7
Medicine added successfully!
```

```
--- Inventory Management ---
1. View Inventory
2. Add Medicines
3. Update Inventory
4. Delete Medicines
5. Back to Admin Dashboard
Enter your choice: 3
Enter Medicine ID to update: 123012
Current details of 123012:
Name: Penicillin
Quantity: 210
Category: tablet
Price: 3.00
Location Rack: 4
Enter new Medicine Name: Penicillin
Enter new Quantity: 210
Enter new Category: tablet
Enter new Price: 4
Enter new Location Rack: 5
Medicine updated successfully!
```

```
--- Supplier Management ---
1. View Suppliers
2. Add Supplier
3. Update Supplier
4. Delete Supplier
5. Back to Admin Dashboard
Enter your choice: 1
Displaying Suppliers...
```

ID	Name	Address	Phone	Email
123	XYZ Pharmaceuticals	Chennai, Tamil Nadu	8745632145	xyz@xyzpharma.com
136	ABC PharmaSupply	Trichy	7894561235	abc@pharmsupp.com
145	Daily Pharma Ltd	Hyderabad	7854699321	daily@dpharma.com
156	MedAll	Chennai	9874585236	mainid@medall.com
162	MedHead Pharmaceuticals	Trichy	7894561335	abc@pharmsupp.com

```
--- Supplier Management ---
1. View Suppliers
2. Add Supplier
3. Update Supplier
4. Delete Supplier
5. Back to Admin Dashboard
Enter your choice: 2
Enter Supplier ID: 163
Enter Supplier Name: Suraj
Enter Supplier Address: ABD nagar chennai
Enter Supplier Phone: 1236782346
Enter Supplier Email: suraj@gmail.com
Supplier added successfully!
```

```
--- Supplier Management ---
1. View Suppliers
2. Add Supplier
3. Update Supplier
4. Delete Supplier
5. Back to Admin Dashboard
Enter your choice: 3
Enter Supplier ID to Update: 163
Enter New Supplier Name: Suraj Kumar
Enter New Supplier Address: ABD nagar chennai
Enter New Supplier Phone: 1236782346
Enter New Supplier Email: surajkumar@gmail.com
Supplier updated successfully!
```

```
---Employee Management ---
1. View Employees
2. Add Employees
3. Update Employees
4. Delete Employees
5. Back to Admin Dashboard
Enter your choice: 1
Displaying Employees...
ID       First Name         Last Name          Birth Date  Age  Sex      Job Type      Join Date     Salary    Phone         Email
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------
1        Admin              -                  1989-05-24  30   Female   Admin         2009-06-24    95000.00  9874563219    admin@pharmacia.com
4567001  Varshini           Elangovan          1995-10-05  25   Female   Pharmacist    2017-11-12    25000.00  9967845123    evarsh@hotmail.com
4567002  Anita              Shree              2000-10-03  20   Female   Pharmacist    2012-10-06    45000.00  8546123566    anita@gmail.com
4567003  Harish             Raja               1998-02-01  22   Male     Pharmacist    2019-07-06    21000.00  7854123694    harishraja@live.com
4567005  Amaya              Singh              1992-01-02  28   Female   Pharmacist    2017-05-16    32000.00  7894532165    amaya@gmail.com
4567006  Shoaib             Ahmed              1999-12-11  20   Male     Pharmacist    2018-09-05    28000.00  7896541234    shoaib@hotmail.com
4567009  Shayla             Hussain            1980-02-28  40   Female   Manager       2010-05-06    80000.00  7854123695    shaylah@gmail.com
4567010  Daniel             James              1993-04-05  27   Male     Pharmacist    2016-01-05    30000.00  7896541235    daniels@gmail.com
```

```
---Employee Management ---
1. View Employees
2. Add Employees
3. Update Employees
4. Delete Employees
5. Back to Admin Dashboard
Enter your choice: 2
Enter Employee ID: 4563220
Enter First Name: Jothi
Enter Last Name: Sri
Enter Birth Date (YYYY-MM-DD): 2002-08-18
Enter Age: 20
Enter Sex: Female
Enter Job Type: Pharmacist
Enter Join Date (YYYY-MM-DD): 2023-08-19
Enter Salary: 50000
Enter Phone Number: 9994676715
Enter Email: jothisri@gmail.com
Enter Address: No.34, ghandhi nagar, chennai
Employee added successfully!
```

```
---Employee Management ---
1. View Employees
2. Add Employees
3. Update Employees
4. Delete Employees
5. Back to Admin Dashboard
Enter your choice: 3
Enter Employee ID to Update: 4563220
Enter New First Name: Jothisri
Enter New Last Name: sankar
Enter New Birth Date (YYYY-MM-DD): 2005-08-18
Enter New Age: 20
Enter New Sex: Female
Enter New Job Type: Pharmacist
Enter New Join Date (YYYY-MM-DD): 2023-08-19
Enter New Salary: 50000
Enter New Phone Number: 9994676715
Enter New Email: jothisrisankar@gmail.com
Enter New Address: No.46, ghandhi nagar, chennai
Employee updated successfully!
```

```
---Employee Management ---
1. View Employees
2. Add Employees
3. Update Employees
4. Delete Employees
5. Back to Admin Dashboard
Enter your choice: 4
Enter Employee ID to Delete: 4563220
Employee deleted successfully!
```

```
---Customers Management ---
1. View Customers
2. Add Customers
3. Update Customers
4. Delete Customers
5. Back to Admin Dashboard
Enter your choice: 4
Enter Customer ID to Delete: 987108
Customer deleted successfully!
```

```
---Customers Management ---
1. View Customers
2. Add Customers
3. Update Customers
4. Delete Customers
5. Back to Admin Dashboard
Enter your choice: 1
Displaying Customers...
```

ID	First Name	Last Name	Age	Sex	Phone	Email
987101	Safia	Malik	22	Female	9632587415	safia@gmail.com
987102	Varun	Ilango	24	Male	9987565423	varun@gmail.com
987103	Suja	Suresh	45	Female	7896541236	suja@hotmail.com
987104	Agatha	Elizabeth	30	Female	7845129635	agatha@gmail.com
987105	Zayed	Shah	40	Male	6789541235	zshah@hotmail.com
987106	Vijay	Kumar	60	Male	8996574123	vijayk@yahoo.com
987107	Meera	Das	35	Female	7845963259	meera@gmail.com

```
---Customers Management ---
1. View Customers
2. Add Customers
3. Update Customers
4. Delete Customers
5. Back to Admin Dashboard
Enter your choice: 2
Enter Customer ID: 987108
Enter First Name: Punitha
Enter Last Name: Sai
Enter Age: 25
Enter Sex: Female
Enter Phone Number: 3456712345
Enter Email: puntha@gmail.com
Enter Username: punitha
Enter Password: puni123
Customer added successfully!
```

```
---Customers Management ---
1. View Customers
2. Add Customers
3. Update Customers
4. Delete Customers
5. Back to Admin Dashboard
Enter your choice: 3
Enter Customer ID to Update: 987108
Enter New First Name: Punitha
Enter New Last Name: Sai
Enter New Age: 25
Enter New Sex: Female
Enter New Phone Number: 2345634567
Enter New Email: punitha@gmail.com
Customer updated successfully!
```

```
--- Admin Dashboard ---
1. Inventory
2. Suppliers
3. Employees
4. Customers
5. View Sales Invoice Details
6. Medicines - Soon to Expire
7. Medicines - Low Stock
8. Logout
Enter your choice: 5
Displaying Sales Invoices...
```

Sale ID	Customer_ID	Sale_Date	Sale_Time	Total_Amount	Employee_ID
1	987101	2020-04-15	13:23:03	180.00	4567009
2	987106	2020-04-21	20:19:31	585.00	1
3	987103	2020-04-15	11:23:53	120.00	4567010
4	987104	2020-04-14	18:20:00	955.00	4567006
5	987103	2020-04-21	15:24:43	45.00	1
6	987102	2020-03-11	10:24:43	140.00	4567001
7	987105	2020-04-24	00:25:54	350.00	1
8	987104	2020-04-24	00:47:47	35.00	4567001
12	987103	2020-04-24	19:33:16	60.00	1
13	987104	2020-04-24	21:15:56	62.50	4567001
15	987107	2020-12-04	18:39:46	420.00	1
16	987106	2020-12-04	18:52:21	30.00	1
17	987103	2020-12-04	19:35:56	57.50	1
18	987105	2020-12-04	19:36:56	160.00	4567001
20	987103	2020-12-04	22:53:18	150.00	4567001
21	987107	2024-11-12	10:51:34	10.00	1
22	987107	2024-11-12	11:29:40	5.00	4567005
23	987101	2024-11-12	12:23:39	10.00	1
24	987101	2024-11-12	12:27:35	30.00	1

```
--- Admin Dashboard ---
1. Inventory
2. Suppliers
3. Employees
4. Customers
5. View Sales Invoice Details
6. Medicines - Soon to Expire
7. Medicines - Low Stock
8. Logout
Enter your choice: 6
Displaying Medicines Soon to Expire (within 6 months)...
Med ID     Expiry Date
      ------------------------------
123010     2021-05-10
123002     2020-12-05
123006     2020-07-01
123004     2023-05-06
123005     2021-04-01
123010     2020-05-02
123001     2022-03-06
```

```
--- Admin Dashboard ---
1. Inventory
2. Suppliers
3. Employees
4. Customers
5. View Sales Invoice Details
6. Medicines - Soon to Expire
7. Medicines - Low Stock
8. Logout
Enter your choice: 7
Displaying Medicines with Low Stock (quantity < 50)...
Med ID     Medicine Name          Quantity
  ------------------------------------------
123003     Livogen                15
123006     Benadryl 200 ML        35
123007     Lopamide               15
123009     Omeprazole             35
```

```
--- Customer Dashboard ---
1. Profile
2. Make Order
3. Order Histories
4. Logout
Enter your choice: 1

--- Customer Profile ---
First Name: Safia
Last Name: Malik
Age: 22
Sex: Female
Phone Number: 9632587415
Email: safia@gmail.com
```

```
--- Customer Dashboard ---
1. Profile
2. Make Order
3. Order Histories
4. Logout
Enter your choice: 3

--- Order History ---
Med ID     Medicine Name                Quantity  Total Price     Date
123003     Livogen                      1         5.00            2024-11-14
123003     Livogen                      2         10.00           2024-11-12
123012     Penicillin                   10        30.00           2024-11-12
123001     Dolo 650 MG                  20        20.00           2020-04-15
123011     Augmentin 250 ML             2         160.00          2020-04-15
```

6 CONCLUSION

The Pharmacy Management System is a comprehensive solution for automating pharmacy operations, built using Object-Oriented Analysis and Design (OOAD) principles. It efficiently manages inventory, sales, customers, suppliers, and reporting. Key features include stock monitoring, automated reordering, and expiration tracking. The system facilitates seamless sales processing with accurate billing, discounts, and taxes. It supports personalized customer service through detailed prescription history management. Supplier coordination ensures timely medicine procurement. The modular design enhances maintainability and scalability. Comprehensive reporting provides actionable business insights. This system ensures efficient, error-free operations and better overall management for pharmacies.