PHARMACY MANAGEMENT SYSTEM

Computer Science (083) Project

Developed By

SHARVAN TYAGI
12th E
29

Index

Certificate

This is to certify PHARMACY MANAGEMENT SYSTEM
Computer Science project is developed by **SHARVAN TYAGI** under my supervision in the session 2024-2025.

The work done by him is original.

_____ Computer Science Teacher

_____ EXTERNAL EXAMINER

Date: _____

Acknowledgement

I express our immense gratitude to our Computer Science teacher POOJA KHARE for her intellectual vigour and generously given support that has been invaluable in escalating our determination to reach the goal of writing this project successfully.

I can hardly find appropriate words to express our obligations and gratefulness to the Principal.

I also feel immense pleasure in recording deep sense of indebtedness, gratitude and sincere thanks to all fellow group mates for their help, company and hard work.

I are especially indebted to our parents for their sincere love, moral support and spontaneous encouragement throughout the entire period of this work.

Thank you!

Project Synopsis

Introduction and Need

- The **Pharmacy Management System** is a software application that simplifies the management of medicine inventory, sales, and billing in a pharmacy.

- It automates critical tasks such as managing medicine inventory, recording sales, and generating customer bills. By leveraging Python and MySQL, the system ensures data integrity, quick processing, and ease of use for both administrators and sales staff.

AIM

- The objective of this project is to let us apply programming knowledge into a real- world situation/problem and expose how programming skills help in developing a good software.

Idea of the Project

The Pharmacy Management System is developed to address the operational complexities faced by pharmacies in managing their day-to-day activities. The idea behind this project is to create an automated, database-driven system that helps pharmacy owners and staff efficiently manage medicines, sales, and billing, thereby ensuring smooth operations and better customer service.

The system is designed to:

1. Automate Inventory Management: By allowing pharmacy staff to add, update, view, and delete medicines from the inventory database. This eliminates manual stock tracking, ensuring accurate and up-to-date records.

2. Track Sales and Billing: The system enables the recording of sales transactions, keeping a real-time record of items sold, quantities, and total amounts. It then generates bills for customers, ensuring correct calculations and timely billing.

3. Improve Customer Experience: By providing fast, accurate, and automated services, the system ensures that customers get their medicines quickly, and their billing process is hassle-free. It also helps the pharmacy maintain a professional and organized operation.

4. Data-Driven Decisions: The system provides management with detailed reports of medicines, sales, and customer transactions, helping them make informed decisions regarding stock levels, pricing strategies, and promotions.

5. Role-Based Access: The system separates roles into Admin and Salesperson to restrict access and protect sensitive operations (like adding new medicines, updating inventory, and generating reports) to authorized users only.

Tabular Representation of Data :

TABLE : MEDICINE

Column	Data Type	Constraints	Description
MED_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each medicine
NAME	VARCHAR(255)	NOT NULL	Name of the medicine
PRICE	FLOAT	CHECK (PRICE > 0)	Price per unit
STOCK	INT	CHECK (STOCK >= 0)	Available stock quantity

TABLE : SALES

Column	Data Type	Constraints	Description
SALE_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each sale
MED_ID	INT	FOREIGN KEY (medicines)	Identifier of the sold medicine
QUANTITY	INT	CHECK (QUANTITY > 0)	Quantity sold
SALE_DATE TIMESTAMP		DEFAULT CURRENT_TIMESTAMP	Date and time of the sale

TABLE : BILLING

Column	Data Type	Constraints	Description
BILL_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each bill
NAME	VARCHAR(255)	NOT NULL	Customer's name
PHONE_NO	VARCHAR(15)	NOT NULL	Customer's phone number
TOTAL_AMOUNT	FLOAT	CHECK (TOTAL_AMOUNT >= 0)	Total bill amount
BILL_DATE	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Date and time of billing

Menu Options :

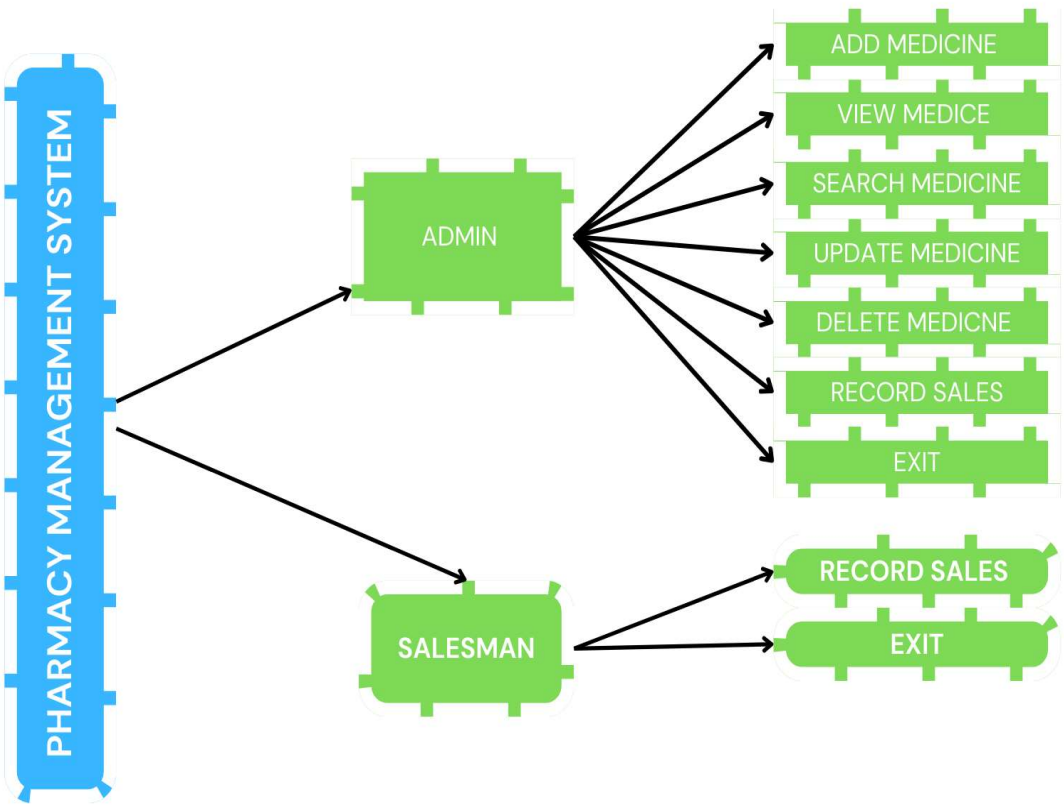
Role	Menu Option	Description
Admin	Add Medicine (AM)	Add a new medicine to the inventory.
	View Medicines (VM)	View the list of all available medicines in the inventory.
	Search Medicine (SM)	Search for medicines by MED_ID or NAME.
	Update Medicine (UM)	Update details (e.g., price, stock) of an existing medicine.
	Delete Medicine (DM)	Remove a medicine from the inventory.
	Record and Generate Bill (R&B)	Record a sale and generate a customer bill.
	Exit (X)	Exit the Admin interface.
Salesperson	Record and Generate Bill (R&B)	Record a sale and generate a customer bill.
	Exit (X)	Exit the Salesperson interface.

Admin Menu

Action	Key	Description
Add Medicine	**AM**	Add new medicines to the inventory system.
View Medicines	**VM**	View a list of all medicines available in stock.
Search Medicine	**SM**	Search for a medicine by ID or Name in the database.
Update Medicine	**UM**	Update details of an existing medicine (e.g., price, stock).
Delete Medicine	**DM**	Remove a medicine from the inventory system.
Record and Generate Bill	**R&B**	Generate a bill after a sale, record sale details.
Exit	**X**	Exit from the Admin menu and return to the login interface.

Salesperson Menu

Action	Key	Description
Record and Generate Bill	**R&B**	Record a sale and generate a bill for the customer.
Exit	**X**	Exit the Salesperson menu and return to the login interface.



Validation and Add on Features

- If a user enters an invalid input, the system will prompt them to reattempt, ensuring a seamless experience. The program is designed to be intuitive and user-centric, making it easy to navigate and interact with. Moreover, the code includes specific features designed to cater to unique requirements, which are clearly detailed and implemented.

SOURCE CODE

```
import mysql.connector as pymysql
from datetime import datetime

passwrd = None
db = None
C = None

def base_check():
    check = 0
    db = pymysql.connect(host="localhost", user="root",
password=passwrd)
    cursor = db.cursor()
    cursor.execute('SHOW DATABASES')
    result = cursor.fetchall()
    for r in result:
        for i in r:
            if i == 'pharmacy':
                cursor.execute('USE pharmacy')
                check = 1
    if check != 1:
        create_database()

def table_check():
    db = pymysql.connect(host="localhost", user="root",
password=passwrd)
    cursor = db.cursor()
    cursor.execute('SHOW DATABASES')
    result = cursor.fetchall()
    for r in result:
        for i in r:
            if i == 'pharmacy':
                cursor.execute('USE pharmacy')
                cursor.execute('SHOW TABLES')
                result = cursor.fetchall()
                if len(result) <= 2:
                    create_tables()
                else:
                    print('      Booting systems...')

def create_database():
```

```
    try:
        db = pymysql.connect(host="localhost", user="root",
password=passwrd)
        cursor = db.cursor()
        cursor.execute("CREATE DATABASE IF NOT EXISTS pharmacy")
        db.commit()
        db.close()
        print("Database 'pharmacy' created successfully.")
    except pymysql.Error as e:
        print(f"Error creating database: {str(e)}")

def create_tables():
    try:
        db = pymysql.connect(host="localhost", user="root",
password=passwrd, database="pharmacy")
        cursor = db.cursor()

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS medicines (
                MED_ID INT PRIMARY KEY,
                NAME VARCHAR(255),
                PRICE FLOAT,
                STOCK INT
            )
        """)

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS sales (
                SALE_ID INT AUTO_INCREMENT PRIMARY KEY,
                MED_ID INT,
                QUANTITY INT,
                SALE_DATE DATE,
                FOREIGN KEY (MED_ID) REFERENCES medicines(MED_ID)
            )
        """)

        cursor.execute("""
            CREATE TABLE IF NOT EXISTS billing (
                BILL_ID INT AUTO_INCREMENT PRIMARY KEY,
                NAME VARCHAR(255),
                PHONE_NO VARCHAR(15),
                TOTAL_AMOUNT FLOAT,
                BILL_DATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """)

        db.commit()
        db.close()
        print("Tables 'medicines', 'sales', and 'billing' created
successfully.")
    except pymysql.Error as e:
        print(f"Error creating tables: {str(e)}")

def QR():
```

```
    result = C.fetchall()
    for r in result:
        print(r)

def add_medicine():
    med_id = int(input("Enter Medicine ID: "))
    name = input("Enter Medicine Name: ")
    price = float(input("Enter Medicine Price: "))
    stock = int(input("Enter Medicine Stock: "))
    data = (med_id, name, price, stock)
    sql = "INSERT INTO medicines (MED_ID, NAME, PRICE, STOCK) VALUES
(%s, %s, %s, %s)"
    try:
        C.execute(sql, data)
        db.commit()
        print('Medicine added successfully...')
    except pymysql.Error as e:
        print(f"Error adding medicine: {str(e)}")

def view_medicines():
    C.execute("SELECT * FROM medicines")
    QR()

def search_medicine():
    search_by = input("Search by [MED_ID, NAME]: ")
    if search_by == 'NAME':
        name = input("Enter Medicine Name: ")
        sql = "SELECT * FROM medicines WHERE NAME = %s"
        C.execute(sql, (name,))
    elif search_by == 'MED_ID':
        med_id = int(input("Enter Medicine ID: "))
        sql = "SELECT * FROM medicines WHERE MED_ID = %s"
        C.execute(sql, (med_id,))
    else:
        print("Invalid search parameter.")
        return
    QR()

def update_medicine():
    med_id = int(input("Enter Medicine ID to update: "))
    field = input("Enter field to update [NAME, PRICE, STOCK]: ")
    new_value = input(f"Enter new value for {field}: ")
    if field in ['PRICE', 'STOCK']:
        new_value = float(new_value) if field == 'PRICE' else
int(new_value)
    sql = f"UPDATE medicines SET {field} = %s WHERE MED_ID = %s"
    try:
        C.execute(sql, (new_value, med_id))
        db.commit()
        print('Medicine updated successfully...')
    except pymysql.Error as e:
        print(f"Error updating medicine: {str(e)}")

def delete_medicine():
```

```
    med_id = int(input("Enter Medicine ID to delete: "))
    sql = "DELETE FROM medicines WHERE MED_ID = %s"
    try:
        C.execute(sql, (med_id,))
        db.commit()
        print('Medicine deleted successfully...')
    except pymysql.Error as e:
        print(f"Error deleting medicine: {str(e)}")

def record_and_generate_bill():
    name = input("Enter Customer Name: ")
    phone_no = input("Enter Customer Phone Number: ")

    total_amount = 0.0
    while True:
        med_id = int(input("Enter Medicine ID for sale (or 0 to
finish): "))
        if med_id == 0:
            break
        quantity = int(input("Enter Quantity: "))

        # Record Sale
        sale_date = datetime.now().date()
        sale_data = (med_id, quantity, sale_date)
        sql_sale = "INSERT INTO sales (MED_ID, QUANTITY, SALE_DATE)
VALUES (%s, %s, %s)"
        try:
            C.execute(sql_sale, sale_data)
        except pymysql.Error as e:
            print(f"Error recording sale: {str(e)}")
            db.rollback()
            continue

        # Calculate Total Amount
        sql_price = "SELECT PRICE FROM medicines WHERE MED_ID = %s"
        C.execute(sql_price, (med_id,))
        result = C.fetchone()
        if result:
            price = result[0]
            total_amount += price * quantity

    # Generate Bill
    bill_data = (name, phone_no, total_amount)
    sql_bill = "INSERT INTO billing (NAME, PHONE_NO, TOTAL_AMOUNT)
VALUES (%s, %s, %s)"
    try:
        C.execute(sql_bill, bill_data)
        db.commit()
        print(f'Bill generated successfully. Total amount:
{total_amount}')
    except pymysql.Error as e:
        print(f"Error generating bill: {str(e)}")

def main():
```

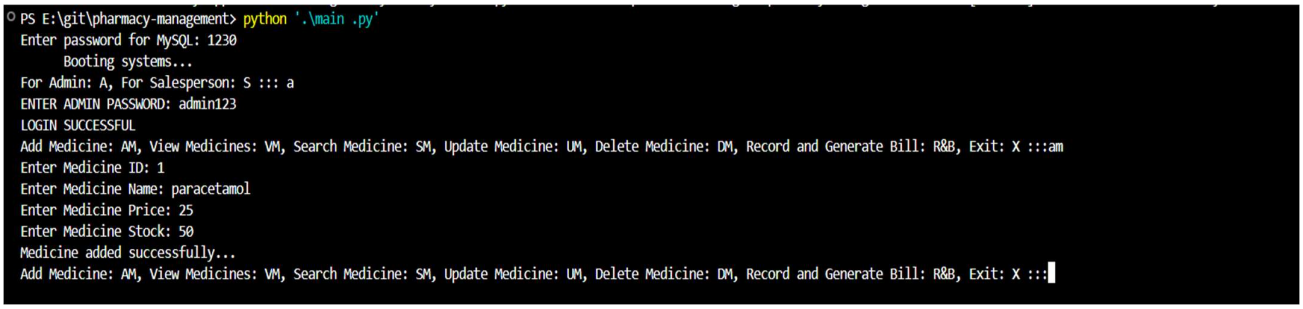
```
    global passwrd
    passwrd = input("Enter password for MySQL: ")

    base_check()

    table_check()

    global db, C
    db = pymysql.connect(host="localhost", user="root",
password=passwrd, database="pharmacy")
    C = db.cursor()
    while True:
        log = input("For Admin: A, For Salesperson: S, EXIT: X ::: ")
        if log.upper() == "A":
            p = input("ENTER ADMIN PASSWORD: ")
            if p == 'admin123':
                print("LOGIN SUCCESSFUL")
                while True:
                    menu = input('''Add Medicine: AM, View Medicines:
VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM,
Record and Generate Bill: R&B, Exit: X :::''')
                    if menu.upper() == 'AM':
                        add_medicine()
                    elif menu.upper() == 'VM':
                        view_medicines()
                    elif menu.upper() == 'SM':
                        search_medicine()
                    elif menu.upper() == 'UM':
                        update_medicine()
                    elif menu.upper() == 'DM':
                        delete_medicine()
                    elif menu.upper() == 'R&B':
                        record_and_generate_bill()
                    elif menu.upper() == 'X':
                        break
                    else:
                        print("Wrong Input")

        elif log.upper() == "S":
            print("Salesperson Interface")
            while True:
                menu = input('''Record and Generate Bill: R&B, Exit: X
:::''')
                if menu.upper() == 'R'or  menu.upper()=="B" :
                    record_and_generate_bill()
                elif menu.upper() == 'X':
                    break
                else:
                    print("Wrong Input")
        elif log.upper()=="X":
           break
```

```
if __name__ == "__main__":
    main()
```

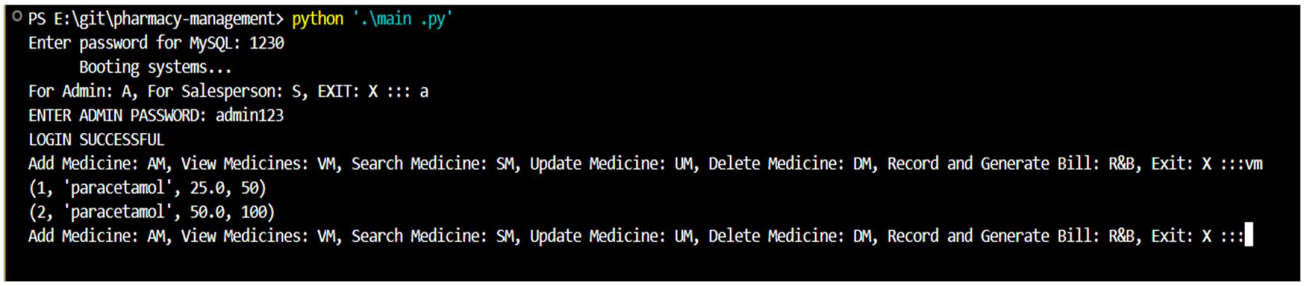
OUTPUT

➢ Admin Controls

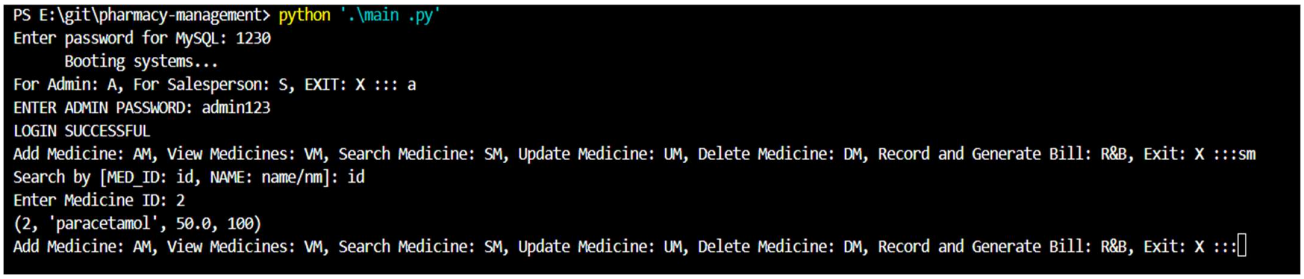
- **ADD MEDICINE**



- **VIEW MEDICINE**



- **SEARCH MEDICINE**



- **UPDATE MEDICINE**

```
PS E:\git\pharmacy-management> python '.\main .py'
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, For Salesperson: S, EXIT: X ::: a
ENTER ADMIN PASSWORD: admin123
LOGIN SUCCESSFUL
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::um
Enter Medicine ID to update: 1
Enter field to update [NAME, PRICE, STOCK]: name
Enter new value for name: paracip
Medicine updated successfully...
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::
```

- **DELETE MEDICINE**

```
PS E:\git\pharmacy-management> python '.\main .py'
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, For Salesperson: S, EXIT: X ::: a
ENTER ADMIN PASSWORD: admin123
LOGIN SUCCESSFUL
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::dm
Enter Medicine ID to delete: 2
Medicine deleted successfully...
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::vm
(1, 'paracip', 25.0, 50)
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::
```

- **RECORD SALES**

```
PS E:\git\pharmacy-management> python '.\main .py'
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, For Salesperson: S, EXIT: X ::: a
ENTER ADMIN PASSWORD: admin123
LOGIN SUCCESSFUL
Add Medicine: AM, View Medicines: VM, Search Medicine: SM, Update Medicine: UM, Delete Medicine: DM, Record and Generate Bill: R, Exit: X :::r
Enter Customer Name: SHRAVAN
Enter Customer Phone Number: 1234567890
Enter Medicine ID for sale (or 0 to finish): 1
Enter Quantity: 2
Enter Medicine ID for sale (or 0 to finish): 0
Bill generated successfully. Total amount: 50.0
```

➢ User Controls

- **RECORD SALES**

```
PS E:\git\pharmacy-management> python '.\main .py'
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, For Salesperson: S, EXIT: X ::: s
Salesperson Interface
Record and Generate Bill: R, Exit: X :::r
Enter Customer Name: shubham
Enter Customer Phone Number: 2112114578
Enter Medicine ID for sale (or 0 to finish): 1
Enter Quantity: 1
Enter Medicine ID for sale (or 0 to finish): 0
Bill generated successfully. Total amount: 25.0
Record and Generate Bill: R, Exit: X :::
```

- **EXIT**

```
PS E:\git\pharmacy-management> python '.\main .py'
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, For Salesperson: S, EXIT: X ::: s
Salesperson Interface
Record and Generate Bill: R, Exit: X :::r
Enter Customer Name: shubham
Enter Customer Phone Number: 2112114578
Enter Medicine ID for sale (or 0 to finish): 1
Enter Quantity: 1
Enter Medicine ID for sale (or 0 to finish): 0
Bill generated successfully. Total amount: 25.0
Record and Generate Bill: R, Exit: X :::x
For Admin: A, For Salesperson: S, EXIT: X ::: x


THANK YOU FOR USING OUR SOFTWAREEEEEE
```

Hardware Requirement PC/Laptop/MacBook with Intel core/i3/i5/i7 or any equivalent With at least 2 GB RAM 10 MB free space on Hard Disk LCD/LED Operating System & Compiler MS Windows/Ubuntu/MacOS Python IDLE 3.x OR colab.research.google.com (gmail account) and MySQL 8.x

References

1.Classnotes

2.www.w3schools.com

3.www.geekforgeeks.com

4.Friends