JSPM'S Jayawantrao Sawant College Of Engineering, Hadapsar

DEPARTMENT OF COMPUTER ENGINEERING
Academic Year - 2024-2025

REPORT ON

"Pharmacy Management System"

Submitted By -

3232	Kshitij Kiran Jagtap
3244	Srushti Santosh More

Under the guidance of

Prof. Nitin Zinzurke

(Department of Computer Engineering)

Acknowledgement

I would like to express my sincere gratitude to everyone who supported me throughout the process of developing this Pharmacy Management System.
First and foremost, I would like to thank my project supervisor, **Prof. Nitin Zinzurke Sir**, for their invaluable guidance, support, and encouragement. Their insights and feedback were instrumental in shaping the direction of this project and enhancing my understanding of the subject matter.

I would also like to extend my appreciation to my group members, who provided assistance and motivation during challenging times. Their constructive discussions and collaborative spirit greatly enriched my learning experience.
A special thanks to my family for their unwavering support and belief in my abilities. Their encouragement provided me with the strength to overcome obstacles and stay focused on my goals.

Lastly, I would like to acknowledge the resources and documentation provided by specific platforms, such as MySQL documentation, Python documentation, etc., which were crucial in the development and implementation of this project.
This project would not have been possible without the contributions of these individuals and resources. Thank you all for your support.

Abstract

The **Pharmacy Management System** is a database-driven application aimed at streamlining the core operations of a pharmacy. By leveraging Python for the frontend and MySQL for the backend, the system allows users to manage and monitor essential pharmacy functions, including medicine availability, alternative medicine suggestions, and billing.

A key feature of the system is the **automated discount management**, where discount offers are applied through triggers in the database, ensuring real-time updates to pricing. The system also supports checking alternative medicines when the requested ones are unavailable, enhancing customer satisfaction.

The system's **billing module** calculates the total based on the selected medicines and quantities, automatically deducting stock after each purchase. This ensures that inventory is updated dynamically, giving accurate insights into stock levels and medicine availability.

While the project meets basic pharmacy management requirements, it is limited in scalability, advanced analytics, and external integration. Future improvements could address these limitations, enhancing its functionality for real-world pharmacy operations.

Introduction

In today's fast-paced environment, pharmacies require efficient systems to manage essential operations such as inventory, billing, and customer service. Manual processes are prone to errors and inefficiencies, leading to delayed service and inaccurate stock management. To address these challenges, the **Pharmacy Management System** is designed to automate and streamline the core functions of a pharmacy, improving the overall efficiency of operations.

This project utilizes **Python** for the frontend and **MySQL** for the backend. Python provides a versatile and easy-to-use platform for building the user interface, while MySQL ensures reliable data management and real-time updates. Together, these technologies offer a seamless solution for tracking medicines, managing stock, and handling customer requests with greater accuracy and speed.

One of the standout features of the system is the **automated discount calculation** through MySQL triggers. The system dynamically applies discounts on medicines that are part of promotional offers, ensuring accurate pricing during the billing process. The system also suggests **alternative medicines** when the requested items are out of stock, providing customers with suitable substitutes when needed.

The system further enhances pharmacy operations by updating stock levels after each transaction, ensuring real-time inventory accuracy. This helps prevent stockouts and ensures that customers are informed of medicine availability. The project also includes the functionality to manage supplier information, update pricing, and maintain accurate records of medicine quantities.

While the **Pharmacy Management System** addresses key operational needs, it has some limitations, including its focus on single-location pharmacies and the lack of advanced features like multi-store management or integration with healthcare systems. However, the project serves as a solid foundation that can be expanded with additional features in the future, making it a valuable tool for pharmacy management.

Advantages & Disadvantages

Advantages

1. **Improved Efficiency:** The system automates inventory management, billing, and discount application, reducing manual errors and speeding up pharmacy operations.

2. **Real-Time Stock Updates:** The system ensures up-to-date information on medicine availability and stock levels, enabling efficient inventory tracking and preventing stockouts.

3. **Discount and Offer Management**: Automated discount calculations through triggers make it easy to handle special pricing offers without manual input, ensuring consistency.

4. **Alternative Medicine Suggestions:** When a medicine is unavailable, the system provides alternative suggestions, improving customer satisfaction by offering immediate solutions.

5. **Accurate Billing:** The billing system includes price adjustments based on quantity and discounts, ensuring precise financial transactions and minimizing errors.

Disadvantages

1. **High Initial Setup Costs:** Implementing a pharmacy management system, especially one with advanced features, can be costly in terms of software, hardware, and training. This can be a financial burden for smaller pharmacies**.**

2. **Technical Issues and Downtime:** The system is prone to technical problems, such as software bugs, hardware failures, or server downtime. If the system crashes or experiences issues, pharmacy operations could be severely disrupted.

3. **Data Security Risks:** Since pharmacy management systems handle sensitive data (e.g., patient information, prescription details), they are vulnerable to cyberattacks and data breaches, which could compromise customer privacy.

4. **Dependence on IT Support:** Pharmacies may need constant access to technical support for troubleshooting and system maintenance. Any IT-related issue can affect the pharmacy's ability to operate smoothly.

5. **Complexity for Non-Tech-Savvy Staff**: Staff who are not familiar with technology may find it difficult to learn and adapt to the system, leading to inefficiencies or errors in managing inventory, prescriptions, or billing.

Limitations of this project

1. **Basic UI:** The UI created with Tkinter may lack modernity and scalability. This could affect user experience and system adaptability for future needs.

2. **Online Orders:** The system does not support online pharmacy orders or delivery services. This limits its functionality compared to contemporary pharmacy systems.

3. **Advanced Analytics:** Lacks features for advanced analytics like sales trends and customer insights. This restricts the ability to make data-driven business decisions.

4. **Single Location:** Operates only for single locations without multi-store inventory synchronization. This can hinder expansion and operational efficiency.

5. **Manual Data Entry:** Requires manual input of medicines and suppliers, increasing setup time and errors. This may lead to inaccuracies in inventory management.

6. **Limited Inventory Management:** Does not provide features for automatic restocking or expiry alerts. This could result in stockouts or the sale of expired products.

7. **Offer Management:** Lacks complex promotional campaigns or loyalty programs. This can reduce customer retention and sales opportunities.

8. **External Integration:** No integration with healthcare databases, EMR, or insurance systems. This limits the system's interoperability with other healthcare applications.

9. **User Authentication:** Absence of user authentication and role-based access control raises security concerns. This can jeopardize sensitive patient and pharmacy data.

10. **Scalability:** May face performance issues as the project grows in complexity with more transactions. This could impact system responsiveness and user satisfaction.

Code

app.py

```
import tkinter as tk
from tkinter import messagebox
import mysql.connector

# Function to check medicine availability
def check_availability():
    medicine_name = medicine_entry.get()

    # Connect to MySQL database
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root123",
        database="pharmacy_db"
    )

    cursor = connection.cursor()

    # Query to check if medicine is available
    query = "SELECT medicine_id, name, price, discounted_price, quantity FROM medicines WHERE
name = %s AND availability = 1"
    cursor.execute(query, (medicine_name,))
    medicine = cursor.fetchone()

    if medicine:
        # If medicine is available, display its details
        result_text.set(f"Medicine: {medicine[1]}\nPrice: {medicine[2]}\nDiscounted Price:
{medicine[3]}\nQuantity: {medicine[4]}")
    else:
        # If not available, check for alternatives
        check_alternatives(cursor, medicine_name)

    connection.close()

# Function to check alternatives if medicine is unavailable
def check_alternatives(cursor, medicine_name):
    query = "SELECT medicine_id FROM medicines WHERE name = %s"
    cursor.execute(query, (medicine_name,))
    org_medicine = cursor.fetchone()

    if org_medicine:
```

```
        org_medicine_id = org_medicine[0]
        query = """SELECT m.name, m.discounted_price, m.quantity
                   FROM alternatives a
                   JOIN medicines m ON a.alt_medicine_id = m.medicine_id
                   WHERE a.org_medicine_id = %s"""
        cursor.execute(query, (org_medicine_id,))
        alternatives = cursor.fetchall()

        if alternatives:
            result_text.set(f"No stock of {medicine_name}. Showing alternatives:\n")
            for alt in alternatives:
                result_text.set(result_text.get() + f"\nAlternative: {alt[0]}, Discounted Price: {alt[1]},
Quantity: {alt[2]}")
        else:
            messagebox.showinfo("Result", "No alternatives available")
    else:
        messagebox.showinfo("Result", f"Medicine {medicine_name} not found.")

# Function to handle billing
def add_to_bill():
    medicine_name = bill_medicine_entry.get()
    quantity = int(bill_quantity_entry.get())

    # Connect to MySQL database
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root123",
        database="pharmacy_db"
    )

    cursor = connection.cursor()

    # Query to get the discounted price and current quantity
    query = "SELECT discounted_price, quantity FROM medicines WHERE name = %s"
    cursor.execute(query, (medicine_name,))
    medicine = cursor.fetchone()

    if medicine:
        discounted_price, available_quantity = medicine

        if quantity <= available_quantity:
            total_cost = discounted_price * quantity
            total_bill_text.set(f"Total Bill for {quantity} of {medicine_name}: {total_cost:.2f}")

            # Update the quantity in the database
            new_quantity = available_quantity - quantity
```

```
            update_query = "UPDATE medicines SET quantity = %s WHERE name = %s"
            cursor.execute(update_query, (new_quantity, medicine_name))
            connection.commit()
            messagebox.showinfo("Success", f"{quantity} of {medicine_name} purchased successfully!")
        else:
            messagebox.showwarning("Quantity Error", "Requested quantity exceeds available stock.")
    else:
        messagebox.showinfo("Result", f"Medicine {medicine_name} not found.")

    connection.close()

# Create main window
root = tk.Tk()
root.title("Pharmacy Availability Checker")

# Set the size of the window
root.geometry("800x600")

# Create input field for checking medicine availability
medicine_label = tk.Label(root, text="Enter Medicine Name:")
medicine_label.pack(pady=5)

medicine_entry = tk.Entry(root, width=50)
medicine_entry.pack(pady=5)

check_button = tk.Button(root, text="Check Availability", command=check_availability, width=20)
check_button.pack(pady=10)

# Create result section
result_text = tk.StringVar()
result_label = tk.Label(root, textvariable=result_text, justify="left")
result_label.pack(pady=10)

# Billing section
billing_label = tk.Label(root, text="Billing Section", font=("Arial", 16))
billing_label.pack(pady=10)

bill_medicine_label = tk.Label(root, text="Enter Medicine Name for Billing:")
bill_medicine_label.pack(pady=5)

bill_medicine_entry = tk.Entry(root, width=50)
bill_medicine_entry.pack(pady=5)

bill_quantity_label = tk.Label(root, text="Enter Quantity:")
bill_quantity_label.pack(pady=5)

bill_quantity_entry = tk.Entry(root, width=10)
```

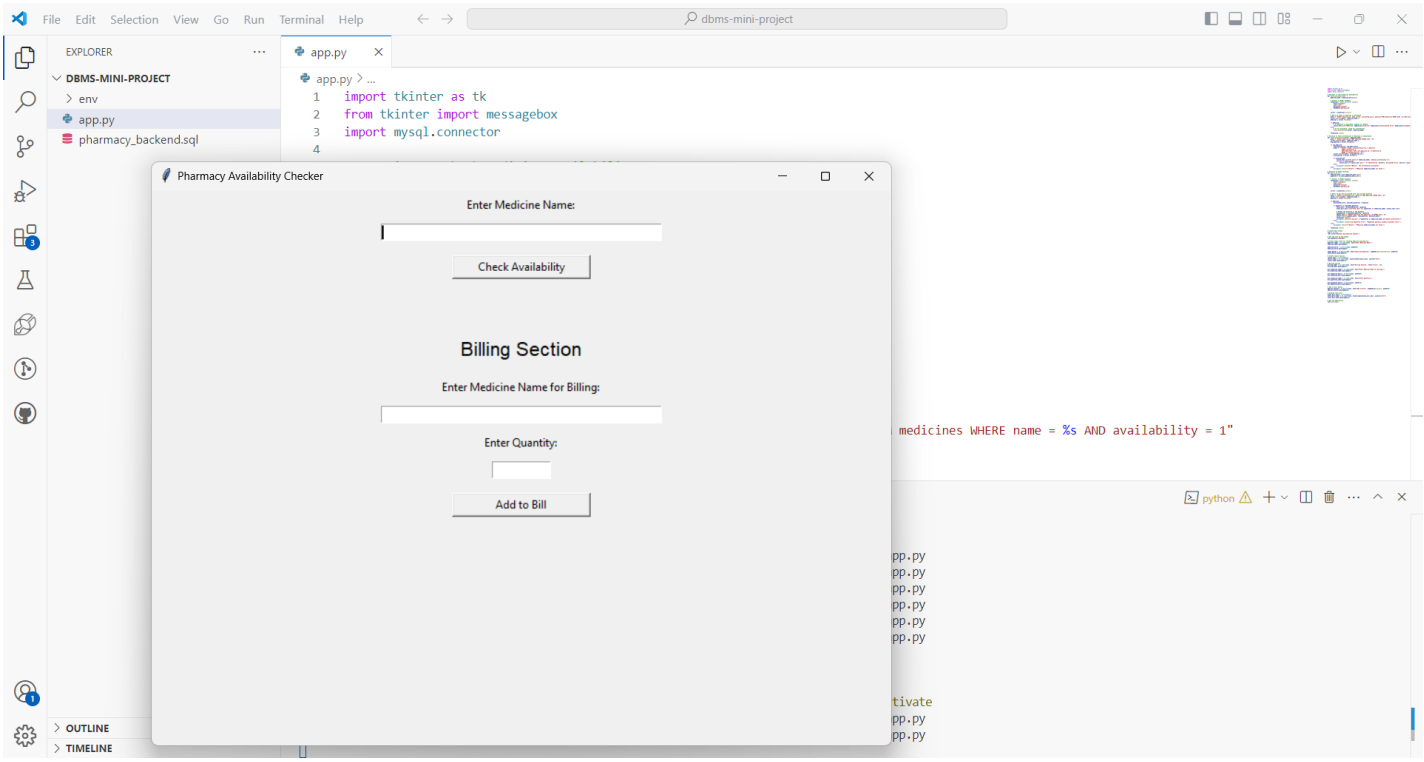
```
bill_quantity_entry.pack(pady=5)

# Add to bill button
add_bill_button = tk.Button(root, text="Add to Bill", command=add_to_bill, width=20)
add_bill_button.pack(pady=10)

# Display total bill
total_bill_text = tk.StringVar()
total_bill_label = tk.Label(root, textvariable=total_bill_text, justify="left")
total_bill_label.pack(pady=10)

# Run the application
root.mainloop()
```

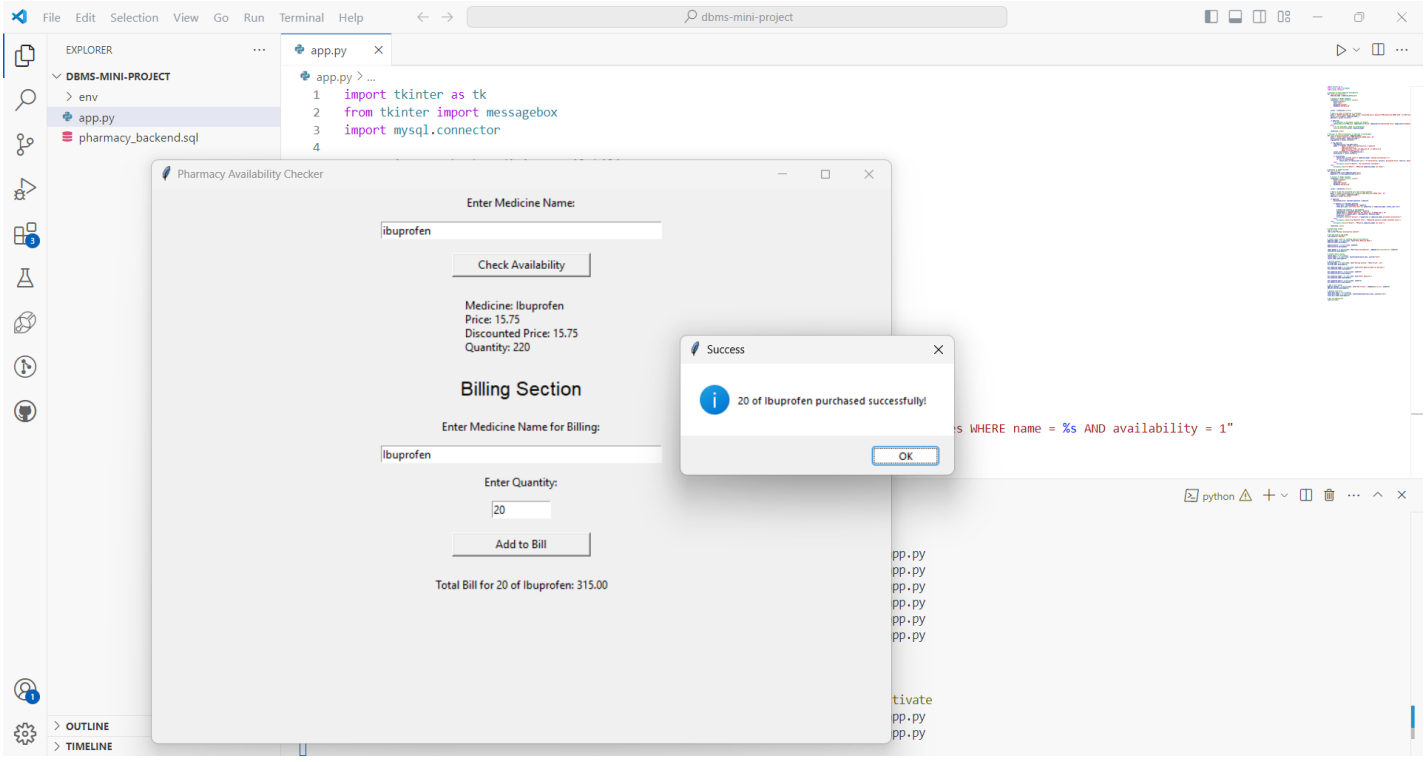
pharmacy_backend.sql

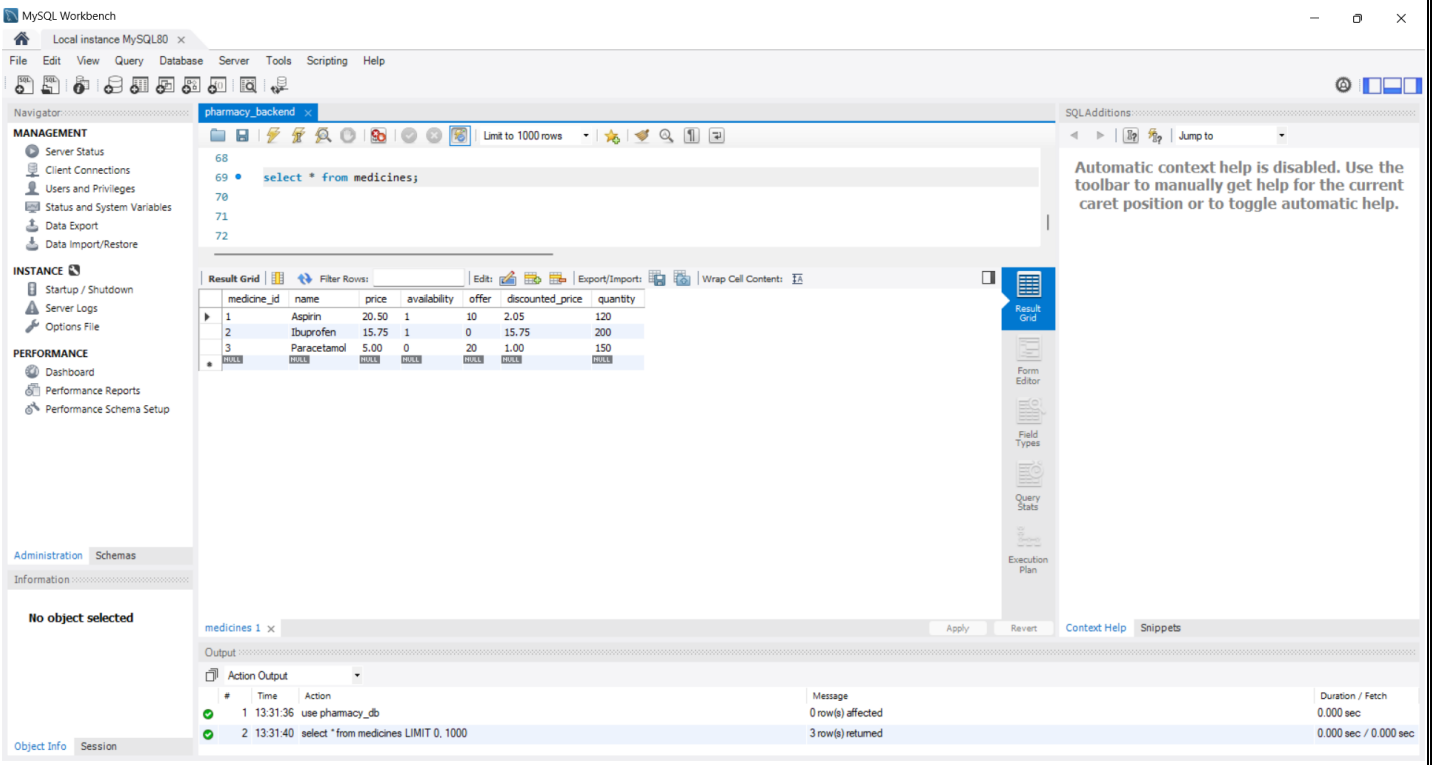
```
create database pharmacy_db;
use pharmacy_db;

create table medicines(
medicine_id int auto_increment primary key,
name varchar(100),
price decimal(10,2),
availability int default 1,
offer int default 0,
discounted_price decimal(10,2));

create table suppliers(
        supplier_id int auto_increment primary key,
   name varchar(100),
   contact_info varchar(255)
);

create table alternatives(
        alt_id int auto_increment primary key,
   org_medicine_id int,
   alt_medicine_id int,
   foreign key (org_medicine_id) references medicines(medicine_id),
   foreign key (alt_medicine_id) references medicines(medicine_id)
);

desc medicines;

delimiter //
create trigger calculate_discounted_price
before insert on medicines
for each row
```

```
begin
 if new.offer>0 then
        set new.discounted_price = new.price - (new.price * new.offer / 100);
 else
        set new.discounted_price = new.price;
 end if;
end; //
delimiter ;

insert into medicines (name, price, availability, offer) values
('Aspirin',20.50,1,10),
('Ibuprofen',15.75,1,0),
('Paracetamol',5.00,0,20);

insert into suppliers (name, contact_info) values
('Supplier A','1234 Street, City'),
('Supplier B','5678 Avenue, City'),
('Supplier C','7642 Shard, City');

insert into alternatives (org_medicine_id,alt_medicine_id) values
(1,2);

ALTER TABLE medicines
ADD quantity INT DEFAULT 0;

update medicines
set quantity = 120
where name = 'Aspirin';

update medicines
set quantity = 220
where name = 'Ibuprofen';

update medicines
set quantity = 150
where name = 'Paracetamol';
```

Output


Pharmacy Availability Checker
Enter Medicine Name:
Check Availability
Billing Section
Enter Medicine Name for Billing:
Enter Quantity:
Add to Bill
app.py
import tkinter as tk
from tkinter import messagebox
import mysql.connector
DBMS-MINI-PROJECT
env
app.py
pharmacy_backend.sql


Pharmacy Availability Checker
Enter Medicine Name:
ibuprofen
Check Availability
Medicine: Ibuprofen
Price: 15.75
Discounted Price: 15.75
Quantity: 220
Billing Section
Enter Medicine Name for Billing:
Ibuprofen
Enter Quantity:
20
Add to Bill
Total Bill for 20 of Ibuprofen: 315.00
Success
20 of Ibuprofen purchased successfully!
OK


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Information
No object selected
Object Info Session
pharmacy_backend
Limit to 1000 rows
68
69 • select * from medicines;
70
71
72
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
medicine_id name price availability offer discounted_price quantity
1 Aspirin 20.50 1 10 2.05 120
2 Ibuprofen 15.75 1 0 15.75 200
3 Paracetamol 5.00 0 20 1.00 150
NULL NULL NULL NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
medicines 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 13:31:36 use pharmacy_db 0 row(s) affected 0.000 sec
2 13:31:40 select * from medicines LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec

Conclusion

The Pharmacy Management System developed in this project provides an efficient solution for managing essential pharmacy operations, such as medicine availability, alternative suggestions, inventory tracking, and billing. Utilizing Python for the frontend interface and MySQL for the backend database ensures ease of data handling and flexibility in implementing core functionalities. By automating tasks like applying discounts through triggers, the system enhances both accuracy and speed in billing and inventory updates.

While the system successfully meets the basic requirements of pharmacy management, it is limited in its ability to scale, handle advanced analytics, and integrate with external systems. Future enhancements could focus on improving the user interface, enabling multi-store synchronization, and incorporating real-time analytics and advanced inventory features. Despite its limitations, this project demonstrates a strong understanding of database management and application development, offering a solid foundation for further improvements and real-world applications.