**A**

**MINI PROJECT REPORT**

**ON**

**“PharmacyManagement System”**

**Submitted by**

1. Sanskruti Thakare(31262)
2. Vrashali Chavan(31266)
3. Vishakha Wanare (31271)
4. Gayatri Yeole(31275)

Under the Guidance of

**Prof. R. J. Sapkal**



**DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

**BHARATI VIDYAPEETH’S COLLEGE OF ENGINEERING FOR WOMEN,**

**PUNE-43**



**SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE**

**YEAR 2023-24**

**CERTIFICATE**

This is to certify that the TE Project report of

**“ Pharmacy Management System ”**

Submitted by,

1. Sanskruti Thakare(31262)
2. Vrashali Chavhan(31266)
3. Vishakha Wanare (31271)
4. Gayatri Yeole(31275)

Of TE (Electronics & Telecommunication Engineering ) is a bonafide work carried out by them under the guidance of **Prof. R. J. Sapkal** and it is approved for the partial fulfilment of the requirement of Savitribai Phule Pune University of Engineering in Electronics & Telecommunication Engineering at **Bharati Vidyapeeth’s College of Engineering for Women, Dhankawdi,Pune-43**

Prof.R. J. Sapkal Prof. Dr. S. R. Patil

Guide H.O.D (E&TC)

Date:-

Place: - Pune

**ACKNOWLEDGEMENT**

On the submission of our Project report on “Pharmacy Management system”, we would like to extend our gratitude and sincere thanks to our **HOD Prof. Dr. S.R. Patil** and guide **Prof .R.J.Sapkal**, Department of Electronics and Telecommunication Engineering for their constant motivation and support during the course of our work. We truly appreciate and value their esteemed guidance and encouragement from the beginning. We are indebted for having helped us shape the problem and providing insights towards the solution. And for providing a solid background for our studies and research thereafter. Our guide has been a great source of inspiration to us and we thank her from the bottom of our heart. Above all, we would like to thank you all friends who’s direct and indirect support helped us.

Project Team

1. Sanskruti Thakare(31262)
2. Vrashali Chavan(31266)
3. Vishakha Wanare (31271)
4. Gayatri Yeole(31275)

**LIST OF FIGURES**

**INDEX**

CERTIFICATE………………………………………………………………………….……

ACKNOWLEDGEMENT ……………………………………………………….…………...

LIST OF FIGURES………………………………………………………………….………...

ABSTRACT…………………………………………………………………………….………

1. INTRODUCTION……………………………………………………………………............

2. SCOPE………………………………………………………………………………………

3. REQUIRMENTS……………………………………………………………………………

4. DATA MODELING FEATURES…………………………………………………………..

5. DATA DICTIONARY………………………………………………………………………

6. RELATIONAL DATABASE DESIGN…………………………………………………….

7. E-R DIAGRAM……………………………………………………………………………..

8. DATABASE NORMALIZATION………………………………………………………….

9. GRAPHICAL USER INTERFACE…………………………………………………………

10. SOURCE CODE……………………………………………………………………………

11. TESTING DOCUMENT……………………………………………………………………

12. RESULT……………………………………………………………………………………

13. CONCLUSION……………………………………………………………………………

14. REFERENCES…………………………………………………………………………….

**ABSTRACT**

Pharmacy Management System application to help pharmacist to manage pharmacy in the systematic ways. Pharmacy Management System can make the work easier by giving the details of medicine when its name is entered. A computer gives the details of the medicine like rate of medicine, and expiry date of the medicine. It becomes very difficult in big medical stores to handle the details of all the medicines manually, so by using this pharmacy management system We can maintain the records of all the medicines. It is fed with the information whenever new medicines are brought and it is provided with expire date with search option. When we entire the name of medicine it gives the details of medicine .One of the most important responsibilities of pharmacy management is to supervise and manage the pharmacy employees in order to ensure healthy working relationships and outcomes. Each of these functions is critical to the pharmacy’s operation and should be explained by the management. However, most pharmacies faced problems such as insufficient service promotions, lack of coherence of pharmacy services in hospitals, poor drug information systems and the inconsistency of the pharmacy information management due to its manual processes. Now these are the problems that must be solved with this Pharmacy Management

**CHAPTER 1**

**INTRODUCTION**

Pharmacy management is a crucial component of the healthcare industry, responsible for the safe and efficient handling of pharmaceutical products and patient-related data. In an era marked by technological advancements and evolving patient expectations, the development of a robust pharmacy management system becomes imperative.

This report documents the design, development, and implementation of a comprehensive Pharmacy Management System, a software application tailored to streamline operations within a pharmacy setting. The primary goal of this project is to create a user-friendly, secure, and efficient system that helps pharmacists, healthcare professionals, and administrators manage various aspects of pharmacy operations, including the tracking of medicines, sales, suppliers, prescriptions, and staff management.

In this report, we provide a detailed account of the system's architecture, features, and functionality. We also discuss the database design, data modeling, and user interactions. Throughout the project, we have adhered to best practices in software development to ensure data integrity, security, and an intuitive user experience.

Our Pharmacy Management System offers several key benefits, including:

* Inventory Management: Efficiently manage medicine inventory, reducing the risk of overstocking or understocking essential pharmaceuticals.
* Sales Tracking: Record and analyze sales data to facilitate better decision-making, from sales trends to profit margins.
* Supplier Management: Simplify interactions with suppliers by maintaining detailed records and facilitating communication.
* Prescription Management :Enable healthcare professionals to manage patient prescriptions with ease and precision.
* Staff Administration: Streamline staff management, from hiring to role assignments and contact information.

As the pharmaceutical industry continues to evolve, the role of pharmacy management systems becomes increasingly prominent. This report serves as a comprehensive guide to our Pharmacy Management System, detailing its various components, functionality, and the advantages it offers to the healthcare sector. It is our hope that this system will contribute to the enhancement of pharmacy operations and, ultimately, the well-being of patients in our care This introduction provides an overview of the project's objectives and the importance of a pharmacy management system in the modern healthcare landscape. It sets the tone for the rest of the report, which can delve into technical details, implementation, and other aspects of the project.

**CHAPTER2**

**SCOPE**

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Pharmacy Management System. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

• In computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.

• In computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.

• To assist the staff in capturing the effort spent on their respective working areas.

• To utilize resources in an efficient manner by increasing their productivity through automation

**CHAPTER 3**

**REQUIREMENTS**

**Software Requirements**

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. The minimal software requirements are as follows,

H/w Requirement :

1. Core i5 processor.

2. 4GB Ram.

3. 20GB of hard disk space in terminal machines.

4. 1TB hard disk space in Server Machine.

S/w Requirement :

1. Front end : Python IDLE / Pycharm
2. Back end : MySql Workbench.

**SOFTWARE** **USED**

**Python IDLE**

Every Python installation comes with an **Integrated Development and Learning Environment**, which you’ll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently. While there are many [IDEs](https://realpython.com/python-ides-code-editors-guide/) for you to choose from, Python IDLE is very bare-bones, which makes it the perfect tool for a beginning programmer.

Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.

**MySql Workbench**

MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeler needs for creating complex ER models, forward and reverse engineering, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.

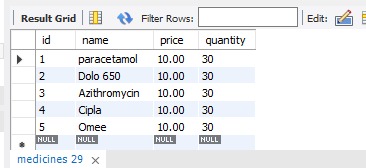
**CHAPTER 4**

**DATA MODELLING**

Medicines Table:

- Stores information about available medicines in the pharmacy.

- Primary Key: Medicine\_ID

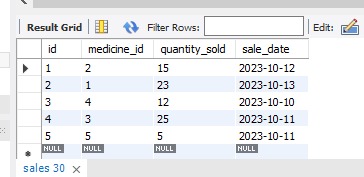


2. Sales Table:

- Records sales transactions, including the medicines sold and quantities.

- Primary Key: Sale\_ID

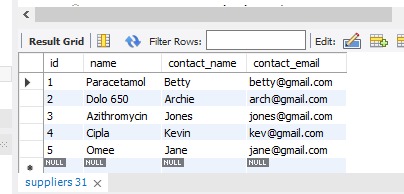
- Foreign Key: Medicine\_ID (references Medicines)



3.Suppliers Table:

- Contains details about the suppliers of medicines.

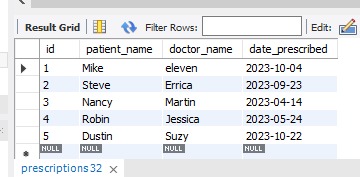
- Primary Key: Supplier\_ID



4. Prescriptions Table:

- Stores prescription records, including patient information and prescribed medicines.

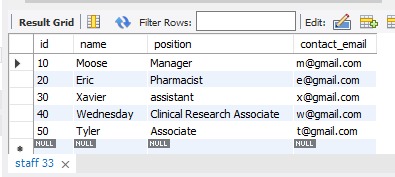
- Primary Key: Prescription\_ID



5.Staff Table:

- Stores information about pharmacy staff members.

- Primary Key: Staff\_ID



**CHAPTER 5**

**DATA DICTIONARY**

In a pharmacy management system project, you'll typically work with data stored in dictionaries or data structures. Below is the of list of data dictionaries that are used in a project. These dictionaries represent data for different aspects of the system:

1. \*Medicine Data Dictionary:\*

- Each medicine in the inventory can be represented as a dictionary.

- Keys: 'ID', 'Name', 'Price', 'Quantity'

- Example:

python

medicine = {'ID': 1, 'Name': 'Aspirin', 'Price': 5.99, 'Quantity': 100}

2. \*Sales Data Dictionary:\*

- Each sale transaction can be represented as a dictionary.

- Keys: 'ID', 'Medicine\_ID', 'Quantity Sold', 'Sale Date'

- Example:

python

sale = {'ID': 1, 'Medicine\_ID': 1, 'Quantity Sold': 10, 'Sale Date': '2023-10-15'}

3. \*Supplier Data Dictionary:\*

- Each supplier's information can be stored in a dictionary.

- Keys: 'ID', 'Name', 'Contact Name', 'Contact Email'

- Example:

python

supplier = {'ID': 1, 'Name': 'ABC Pharmaceuticals', 'Contact Name': 'John Doe', 'Contact Email': 'john@example.com'}

4. \*Prescription Data Dictionary:\*

- Each prescription can be represented as a dictionary.

- Keys: 'ID', 'Patient Name', 'Doctor Name', 'Date Prescribed'

- Example:

python

prescription = {'ID': 1, 'Patient Name': 'Alice', 'Doctor Name': 'Dr. Smith', 'Date Prescribed': '2023-10-10'}

5. \*Staff Data Dictionary:\*

- Information about each staff member can be stored in a dictionary.

- Keys: 'ID', 'Name', 'Position', 'Contact Email'

- Example:

python

staff = {'ID': 1, 'Name': 'Jane', 'Position': 'Pharmacist', 'Contact Email': 'jane@example.com'}

These data dictionaries can be used to represent and manage information in the pharmacy management system. Depending on your specific requirements, you may have additional data dictionaries or more complex data structures for managing relationships between different data elements in the system.

**CHAPTER 6**

**RELATIONAL DATABASE DESIGN**

Designing a relational database for a pharmacy management system involves defining the tables, their relationships, and the attributes (columns) within those tables. Below is a example of a relational database design of project. The Sales table is related to the Medicines table through the Medicine\_ID foreign key, which links each sale to a specific medicine.The Sales table may also have a Staff\_ID foreign key to associate the sales with the staff member who made the sale.

Tables and Relationships:

1. Medicines Table:

- Stores information about available medicines in the pharmacy.

- Primary Key: Medicine\_ID

Attribute:

|  |
| --- |
| Medicine |
| Medicine\_ID (Primary Key)  Name  Price  Quantity |

2. Sales Table:

- Records sales transactions, including the medicines sold and quantities.

- Primary Key: Sale\_ID

- Foreign Key: Medicine\_ID (references Medicines)

Attributes

|  |
| --- |
| Sales |
| Sale\_ID (Primary Key)  Medicine\_ID (Foreign Key)  Quantity Sold  Sale Date |

3. Suppliers Table:

- Contains details about the suppliers of medicines.

- Primary Key: Supplier\_ID

Attributes:

|  |
| --- |
| Suppliers |
| Supplier\_ID (Primary Key)  Name  Contact Name  Contact Email |

4. Prescriptions Table:

- Stores prescription records, including patient information and prescribed medicines.

- Primary Key: Prescription\_ID

Attributes:

|  |
| --- |
| Prescriptions |
| Prescription\_ID (Primary Key)  Patient Name  Doctor Name  Date Prescribed |

5. \*Staff Table:\*

- Stores information about pharmacy staff members.

- Primary Key: Staff\_ID

Attributes:

|  |
| --- |
| Staff |
| Staff\_ID (Primary Key)  Name  Position  Contact Email |

**CHAPTER 7**

**E-R DIAGRAM**

**CHAPTER**

**SOURCE CODE**

Program to create

import mysql.connector

# Establish a connection to the MySQL database

connection = mysql.connector.connect(

host="localhost", # Replace with your MySQL server's host

user="root", # Replace with your MySQL username

password="mysql@2023", # Replace with your MySQL password

database="pharmacy\_db" # Specify the database you want to connect to

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Function to create a new column in a table

def create\_new\_column():

table\_name = input("Enter the name of the table where you want to create a new column: ")

new\_column\_name = input("Enter the name of the new column: ")

new\_column\_data\_type = input("Enter the data type of the new column (e.g., INT, VARCHAR(255), DECIMAL(10, 2)): ")

# Check if the table exists

cursor.execute("SHOW TABLES LIKE %s", (table\_name,))

table\_exists = cursor.fetchone()

if table\_exists:

# Create the new column

alter\_table\_query = f"ALTER TABLE {table\_name} ADD COLUMN {new\_column\_name} {new\_column\_data\_type}"

cursor.execute(alter\_table\_query)

connection.commit()

print(f"New column {new\_column\_name} with data type {new\_column\_data\_type} added to table {table\_name}.")

else:

print(f"Table {table\_name} does not exist.")

# Dictionary to handle different cases

options = {

'1': create\_new\_column,

}

while True:

print("Choose an option:")

print("1. Create a new column in a table")

print("2. Exit")

choice = input("Enter your choice: ")

if choice == '2':

break

selected\_option = options.get(choice)

if selected\_option:

selected\_option()

else:

print("Invalid choice. Please select a valid option.")

# Close the cursor and connection when done

cursor.close()

connection.close()

Program to delete

import mysql.connector

# Establish a connection to the MySQL database

connection = mysql.connector.connect(

host="localhost", # Replace with your MySQL server's host

user="root", # Replace with your MySQL username

password="mysql@2023", # Replace with your MySQL password

database="pharmacy\_db" # Specify the database you want to connect to

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Function to delete data from the medicines table

def delete\_medicine():

medicine\_id = int(input("Enter the ID of the medicine to delete: "))

cursor.execute("SELECT \* FROM medicines WHERE id = %s", (medicine\_id,))

existing\_medicine = cursor.fetchone()

if existing\_medicine:

delete\_medicine\_query = "DELETE FROM medicines WHERE id = %s"

cursor.execute(delete\_medicine\_query, (medicine\_id,))

connection.commit()

print("Medicine data deleted.")

else:

print("Medicine with ID", medicine\_id, "does not exist.")

# Function to delete data from the sales table

def delete\_sale():

sale\_id = int(input("Enter the ID of the sale to delete: "))

cursor.execute("SELECT \* FROM sales WHERE id = %s", (sale\_id,))

existing\_sale = cursor.fetchone()

if existing\_sale:

delete\_sale\_query = "DELETE FROM sales WHERE id = %s"

cursor.execute(delete\_sale\_query, (sale\_id,))

connection.commit()

print("Sale data deleted.")

else:

print("Sale with ID", sale\_id, "does not exist.")

# Function to delete data from the suppliers table

def delete\_supplier():

supplier\_id = int(input("Enter the ID of the supplier to delete: "))

cursor.execute("SELECT \* FROM suppliers WHERE id = %s", (supplier\_id,))

existing\_supplier = cursor.fetchone()

if existing\_supplier:

delete\_supplier\_query = "DELETE FROM suppliers WHERE id = %s"

cursor.execute(delete\_supplier\_query, (supplier\_id,))

connection.commit()

print("Supplier data deleted.")

else:

print("Supplier with ID", supplier\_id, "does not exist.")

# Function to delete data from the prescriptions table

def delete\_prescription():

prescription\_id = int(input("Enter the ID of the prescription to delete: "))

cursor.execute("SELECT \* FROM prescriptions WHERE id = %s", (prescription\_id,))

existing\_prescription = cursor.fetchone()

if existing\_prescription:

delete\_prescription\_query = "DELETE FROM prescriptions WHERE id = %s"

cursor.execute(delete\_prescription\_query, (prescription\_id,))

connection.commit()

print("Prescription data deleted.")

else:

print("Prescription with ID", prescription\_id, "does not exist.")

# Function to delete data from the staff table

def delete\_staff():

staff\_id = int(input("Enter the ID of the staff member to delete: "))

cursor.execute("SELECT \* FROM staff WHERE id = %s", (staff\_id,))

existing\_staff = cursor.fetchone()

if existing\_staff:

delete\_staff\_query = "DELETE FROM staff WHERE id = %s"

cursor.execute(delete\_staff\_query, (staff\_id,))

connection.commit()

print("Staff data deleted.")

else:

print("staff with ID", staff\_id, "does not exist.")

# Dictionary to handle different cases

options = {

'1': delete\_medicine,

'2': delete\_sale,

'3': delete\_supplier,

'4': delete\_prescription,

'5': delete\_staff

}

while True:

print("Choose a table to delete data from:")

print("1. Medicines")

print("2. Sales")

print("3. Suppliers")

print("4. Prescriptions")

print("5. Staff")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '6':

break

selected\_option = options.get(choice)

if selected\_option:

selected\_option()

else:

print("Invalid choice. Please select a valid option.")

# Close the cursor and connection when done

cursor.close()

connection.close()

Program to insert

import mysql.connector

# Establish a connection to the MySQL database

connection = mysql.connector.connect(

host="localhost", # Replace with your MySQL server's host

user="root", # Replace with your MySQL username

password="mysql@2023", # Replace with your MySQL password

database="pharmacy\_db" # Specify the database you want to connect to

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Function to insert data into the medicines table

def insert\_medicine():

medicine\_name = input("Enter medicine name: ")

medicine\_price = float(input("Enter medicine price: "))

medicine\_quantity = int(input("Enter medicine quantity: "))

new\_medicine = (medicine\_name, medicine\_price, medicine\_quantity)

insert\_medicine\_query = "INSERT INTO medicines (name, price, quantity) VALUES (%s, %s, %s)"

cursor.execute(insert\_medicine\_query, new\_medicine)

connection.commit()

print("Medicine data inserted.")

# Function to insert data into the sales table

def insert\_sale():

medicine\_id = int(input("Enter medicine ID for the sale: "))

quantity\_sold = int(input("Enter quantity sold: "))

sale\_date = input("Enter sale date (YYYY-MM-DD): ")

new\_sale = (medicine\_id, quantity\_sold, sale\_date)

insert\_sale\_query = "INSERT INTO sales (medicine\_id, quantity\_sold, sale\_date) VALUES (%s, %s, %s)"

cursor.execute(insert\_sale\_query, new\_sale)

connection.commit()

print("Sale data inserted.")

# Function to insert data into the suppliers table

def insert\_supplier():

supplier\_name = input("Enter supplier name: ")

contact\_name = input("Enter contact name: ")

contact\_email = input("Enter contact email: ")

new\_supplier = (supplier\_name, contact\_name, contact\_email)

insert\_supplier\_query = "INSERT INTO suppliers (name, contact\_name, contact\_email) VALUES (%s, %s, %s)"

cursor.execute(insert\_supplier\_query, new\_supplier)

connection.commit()

print("Supplier data inserted.")

# Function to insert data into the prescriptions table

def insert\_prescription():

patient\_name = input("Enter patient name: ")

doctor\_name = input("Enter doctor name: ")

date\_prescribed = input("Enter date prescribed (YYYY-MM-DD): ")

new\_prescription = (patient\_name, doctor\_name, date\_prescribed)

insert\_prescription\_query = "INSERT INTO prescriptions (patient\_name, doctor\_name, date\_prescribed) VALUES (%s, %s, %s)"

cursor.execute(insert\_prescription\_query, new\_prescription)

connection.commit()

print("Prescription data inserted.")

# Function to insert data into the staff table

def insert\_staff():

staff\_name = input("Enter staff name: ")

position = input("Enter position: ")

contact\_email = input("Enter contact email: ")

new\_staff = (staff\_name, position, contact\_email)

insert\_staff\_query = "INSERT INTO staff (name, position, contact\_email) VALUES (%s, %s, %s)"

cursor.execute(insert\_staff\_query, new\_staff)

connection.commit()

print("Staff data inserted.")

# Dictionary to handle different cases

options = {

'1': insert\_medicine,

'2': insert\_sale,

'3': insert\_supplier,

'4': insert\_prescription,

'5': insert\_staff

}

while True:

print("Choose a table to insert data:")

print("1. Medicines")

print("2. Sales")

print("3. Suppliers")

print("4. Prescriptions")

print("5. Staff")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '6':

break

selected\_option = options.get(choice)

if selected\_option:

selected\_option()

else:

print("Invalid choice. Please select a valid option.")

# Close the cursor and connection when done

cursor.close()

connection.close()

Main Program

while True:

print("Choose an operation:")

print("1. Insert")

print("2. Delete")

print("3. Update")

print("4. Read")

print("5. Create")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '1':

import insert

elif choice == '2':

import delete

elif choice == '3':

import update

elif choice == '4':

import read

elif choice == '5':

import create

elif choice=='6':

break;

else:

print("Invalid choice. Please select a valid option.")

Program to read

import mysql.connector

# Establish a connection to the MySQL database

connection = mysql.connector.connect(

host="localhost", # Replace with your MySQL server's host

user="root", # Replace with your MySQL username

password="mysql@2023", # Replace with your MySQL password

database="pharmacy\_db" # Specify the database you want to connect to

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Function to read data from the medicines table

def read\_medicine():

cursor.execute("SELECT \* FROM medicines")

medicines = cursor.fetchall()

if medicines:

print("Medicines table:")

for medicine in medicines:

print(f" ID: {medicine[0]}\n Name: {medicine[1]}\n Price: {medicine[2]}\n Quantity: {medicine[3]}")

else:

print("Medicines table is empty.")

# Function to read data from the medicines table

def read\_sale():

cursor.execute("SELECT \* FROM sales")

sales = cursor.fetchall()

if sales:

print("sales table:")

for sales in sales:

print(f" ID: {sales[0]}\n medicine\_id: {sales[1]}\n Price: {sales[2]}\n quantity\_sold: {sales[3]}\n sale\_date: {sales[4]}")

else:

print("sales table is empty.")

# Function to read data from the suppliers table

def read\_suppliers():

cursor.execute("SELECT \* FROM suppliers")

suppliers = cursor.fetchall()

if suppliers:

print("suppliers table:")

for suppliers in suppliers:

print(f" ID: {suppliers[0]}\n contact\_name: {suppliers[1]}\n contact\_email: {suppliers[2]}")

else:

print("suppliers table is empty.")

# Function to read data from the prescriptions table

def read\_prescriptions():

cursor.execute("SELECT \* FROM prescriptions")

prescriptions = cursor.fetchall()

if prescriptions:

print("prescriptions table:")

for prescriptions in prescriptions:

print(f" ID: {prescriptions[0]}\n patient\_name: {prescriptions[1]}\n doctor\_name: {prescriptions[2]}\n date\_prescribed: {prescriptions[3]}")

else:

print("prescriptions table is empty.")

# Function to read data from the prescriptions table

def read\_staff():

cursor.execute("SELECT \* FROM staff")

staff = cursor.fetchall()

if staff:

print("staff table:")

for staff in staff:

print(f" ID: {staff[0]}\n name: {staff[1]}\n position: {staff[2]}\n contact\_email: {staff[3]}")

else:

print("staff table is empty.")

# Dictionary to handle different cases

options = {

'1': read\_medicine,

'2': read\_sale,

'3': read\_suppliers,

'4': read\_prescriptions,

'5': read\_staff

}

while True:

print("Choose a table to delete data from:")

print("1. Medicines")

print("2. Sales")

print("3. Suppliers")

print("4. Prescriptions")

print("5. Staff")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '6':

break

selected\_option = options.get(choice)

if selected\_option:

selected\_option()

else:

print("Invalid choice. Please select a valid option.")

# Close the cursor and connection when done

cursor.close()

connection.close()

Program to Update

import mysql.connector

# Establish a connection to the MySQL database

connection = mysql.connector.connect(

host="localhost", # Replace with your MySQL server's host

user="root", # Replace with your MySQL username

password="mysql@2023", # Replace with your MySQL password

database="pharmacy\_db" # Specify the database you want to connect to

)

# Create a cursor object to execute SQL queries

cursor = connection.cursor()

# Function to update data in the medicines table

def update\_medicine():

medicine\_id = int(input("Enter the ID of the medicine to update: "))

column\_to\_update = input("Enter the column to update (e.g., name, price, quantity): ").strip()

new\_value = input(f"Enter the new value for {column\_to\_update}: ")

# Check if the medicine with the given ID exists

cursor.execute("SELECT \* FROM medicines WHERE id = %s", (medicine\_id,))

existing\_medicine = cursor.fetchone()

if existing\_medicine:

update\_medicine\_query = f"UPDATE medicines SET {column\_to\_update} = %s WHERE id = %s"

cursor.execute(update\_medicine\_query, (new\_value, medicine\_id))

connection.commit()

print("Medicine data updated.")

else:

print("Medicine with ID", medicine\_id, "does not exist.")

# Function to update data from the sales table

def update\_sale():

sale\_id = int(input("Enter the ID of the sale to update: "))

column\_to\_update1 = input("Enter the column to update (e.g., id, medicine\_id, quantity\_sold,sale\_date): ").strip()

new\_value1 = input(f"Enter the new value for {column\_to\_update1}: ")

# Check if the sale with the given ID exists

cursor.execute("SELECT \* FROM sales WHERE id = %s", (sale\_id,))

existing\_sale = cursor.fetchone()

if existing\_sale:

update\_sale\_query = f"UPDATE sales SET {column\_to\_update1} = %s WHERE id = %s"

cursor.execute(update\_sale\_query, (new\_value1, sale\_id))

connection.commit()

print("Sale data updated.")

else:

print("Sale with ID", sale\_id, "does not exist.")

# Function to update data from the suppliers table

def update\_supplier():

supplier\_id = int(input("Enter the ID of the supplier to update: "))

column\_to\_update2 = input("Enter the column to update (e.g., id, contact\_name, contact\_email): ").strip()

new\_value2 = input(f"Enter the new value for {column\_to\_update2}: ")

# Check if the sale with the given ID exists

cursor.execute("SELECT \* FROM suppliers WHERE id = %s", (supplier\_id,))

existing\_supplier = cursor.fetchone()

if existing\_supplier:

update\_supplier\_query = f"UPDATE suppliers SET {column\_to\_update2} = %s WHERE id = %s"

cursor.execute(update\_supplier\_query, (new\_value2,supplier\_id))

connection.commit()

print("Supplier data updated.")

else:

print("Supplier with ID", supplier\_id, "does not exist.")

# Function to update data from the prescriptions table

def update\_prescription():

prescription\_id = int(input("Enter the ID of the prescription to update: "))

column\_to\_update3 = input("Enter the column to update (e.g., id, patient\_name, doctor\_name,date\_prescribed): ").strip()

new\_value3= input(f"Enter the new value for {column\_to\_update3}: ")

# Check if the sale with the given ID exists

cursor.execute("SELECT \* FROM prescriptions WHERE id = %s", (prescription\_id,))

existing\_prescription = cursor.fetchone()

if existing\_prescription:

update\_prescription\_query = f"UPDATE prescriptions SET {column\_to\_update3} = %s WHERE id = %s"

cursor.execute(update\_prescription\_query, (new\_value3,prescription\_id))

connection.commit()

print("Prescription data updated.")

else:

print("Prescription with ID", prescription\_id, "does not exist.")

# Function to delete data from the staff table

def update\_staff():

staff\_id = int(input("Enter the ID of the staff member to update: "))

column\_to\_update4 = input("Enter the column to update (e.g., id, name, position,contact\_email): ").strip()

new\_value4= input(f"Enter the new value for {column\_to\_update4}: ")

# Check if the sale with the given ID exists

cursor.execute("SELECT \* FROM staff WHERE id = %s", (staff\_id,))

existing\_staff = cursor.fetchone()

if existing\_staff:

update\_staff\_query = f"UPDATE staff SET {column\_to\_update4} = %s WHERE id = %s"

cursor.execute(updatee\_staff\_query, (new\_value4,staff\_id))

connection.commit()

print("Staff data updated.")

else:

print("staff with ID", staff\_id, "does not exist.")

# Dictionary to handle different cases

options = {

'1': update\_medicine,

'2': update\_sale,

'3': update\_supplier,

'4': update\_prescription,

'5': update\_staff

}

while True:

print("Choose a table to update data from:")

print("1. Medicines")

print("2. Sales")

print("3. Suppliers")

print("4. Prescriptions")

print("5. Staff")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '6':

break

selected\_option = options.get(choice)

if selected\_option:

selected\_option()

else:

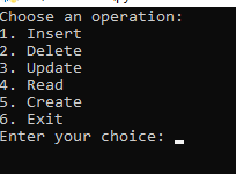
print("Invalid choice. Please select a valid option.")

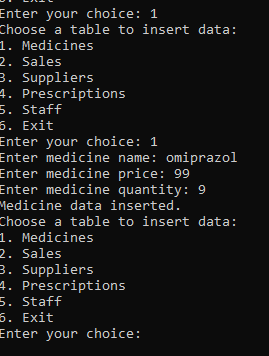
# Close the cursor and connection when done

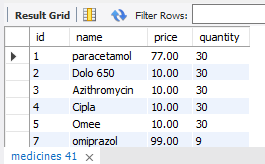
cursor.close()

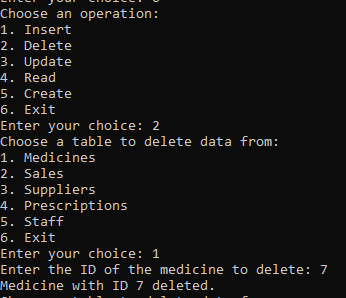
connection.close()

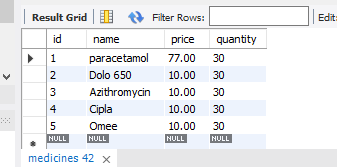
**CHAPTER**

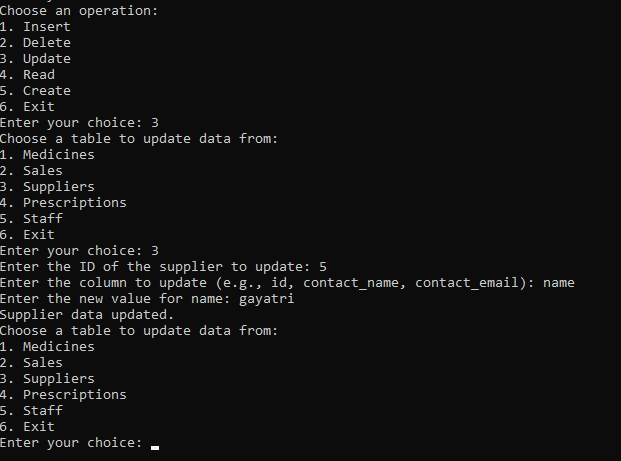
**RESULTS**

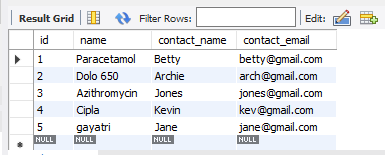
****

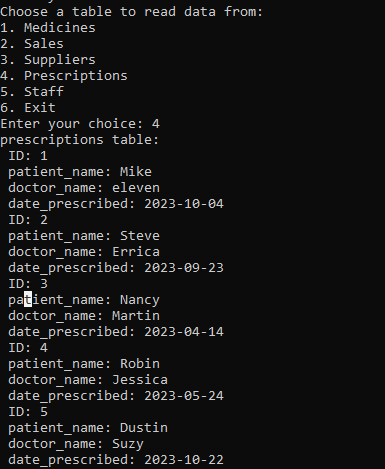
****

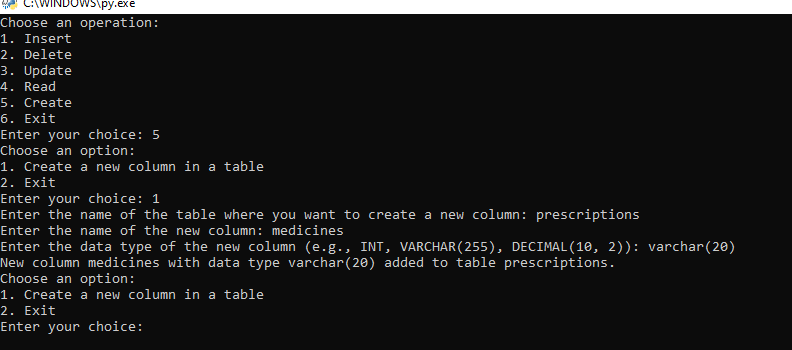
****

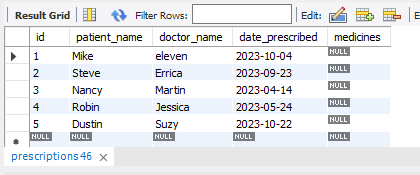
****

****

****

****

****

****

**CHAPTER**

**CONCLUSION**

Working on the project was an excellent experience. It helped us to understand the importance of planning, designing and implementation so far we have learnt in our theory books.

It helped us unleashing our creativity while working in a team. It also realized the importance of team working, communication as a part of this project.

The project was successfully completed after a lot of efforts and work hours. This project

underwent number of compiling, debugging, removing errors, making it bug free, adding

more facilities in Employee Leave Management System and interactivity making it more

reliable and useful. This project focused that scheduling a project and adhering to that

schedule creates a hard sense of time- management. It has also let us know that cooperative-teamwork always produce effective results. The entire project has been developed and deployed as per the requirements stated by the user. It is found to be bug free as per the

testing standards that are implemented.

Finally, we like to conclude that we put all our efforts throughout the development of our project and tried to fulfill most of the requirements of the user.