Pharmacy Management System

CS F212
DBMS PROJECT

Submitted by:-

Anushika (2022A7PS0100P)
Kumari Soumya (2022A7PS1184P)
Madhurika Bhatt (2022A7PS0079P)
Nikhil Joshi (2022A7PS0041P)
Yuvraj Dhaka (2022A7PS0098P)

1.1 **Problem Description:**

This database system is designed to store, manage, query, and retrieve pharmacy-related data. It caters to the needs of managing various aspects of a pharmacy, such as inventory, sales, and customer records. The system is intended to allow efficient management of pharmacy operations. It is a comprehensive solution for organizing and handling the diverse data requirements within a pharmacy setting, contributing to streamlined and effective management of pharmaceutical activities.

1.2 **Features and Descriptions:**

This database system provide various features like:-

• **User Authentication:** Verify identity of employees and admin.
This feature ensures that only authorized users, such as employees and administrators, can access the system. It verifies their identity through methods like usernames, passwords, and possibly multi-factor authentication for added security.

• **Purchase Management**: Record purchases from companies
This feature allows the recording and tracking of purchases from companies. It may include details such as the company name, purchase date, items purchased, quantity, and cost. This data is crucial for inventory management and financial analysis.

• **Sales Management:** Record sale transactions to customers
Similar to purchase management, this feature records sale transactions to customers. It includes details like customer information, sale date, items sold, quantity, price, and total sale amount. This data helps in tracking sales performance and customer behavior.

• **Payment reports**: Give detailed payment reports for purchases, sales, profits etc.
This feature generates detailed reports related to payments. It can include reports on purchases, sales, profits, and other financial metrics. These reports provide insights into the financial health of the business and help in decision-making.

• **Inventory Handling**: Track stock levels and expiry dates.
This feature tracks stock levels and expiry dates of products. It helps in ensuring that the right amount of inventory is maintained, avoiding stockouts or overstock situations. It also helps in managing perishable goods by tracking their expiry dates.

• **Dashboard**: Provide an overview of purchases, sales, payment reports and inventory status.
The dashboard feature provides a summarized overview of various aspects of the business, such as purchases, sales, payment reports, and inventory status. It may include visualizations like charts and graphs to make data analysis easier for users. The dashboard provides a quick snapshot of the business's performance and helps in identifying trends and patterns.
Login Page: This page allows users to enter their credentials (username and password) to access the system. It verifies the credentials and grants access to the system if they are correct.

Admin Dashboard: The admin dashboard provides an overview of the system's status and allows the administrator to manage various aspects of the system, such as users, medicines, and sales.

Add User: This feature allows the administrator to add a new user to the system. It typically includes fields for entering the user's details, such as username, password, role, etc.

View User: This feature allows the administrator to view a list of all users in the system. It may include options to filter or search for specific users.

Update User: This feature allows the administrator to update the details of an existing user, such as their password, role, etc.

Profile: This feature allows users to view and update their own profile information, such as their name, contact information, etc.

Dashboard: This is a general dashboard that provides an overview of the system's key metrics and performance indicators.

Add Medicine: This feature allows the administrator to add a new medicine to the system. It typically includes fields for entering the medicine's details, such as name, quantity, price, etc.

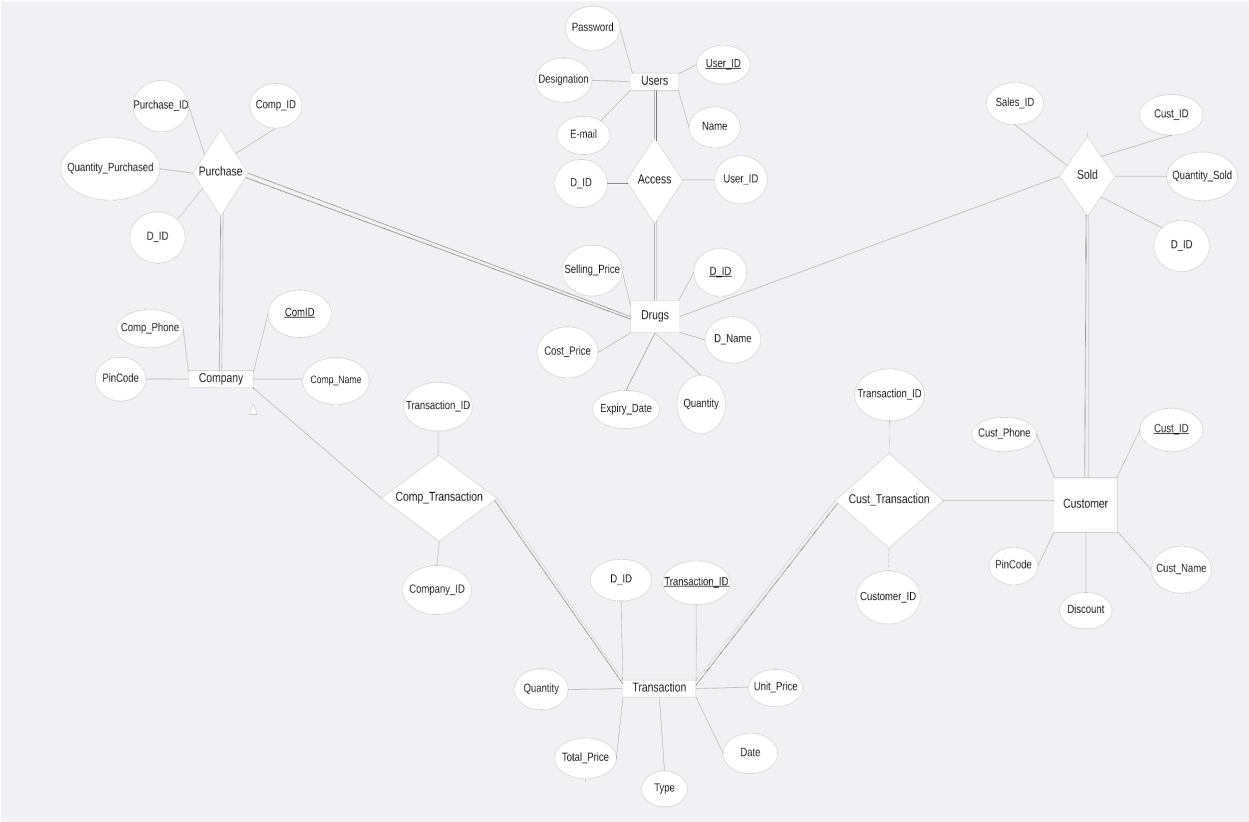
View Medicine: This feature allows the administrator to view a list of all medicines in the system. It may include options to filter or search for specific medicines.

Update Medicine: This feature allows the administrator to update the details of an existing medicine, such as its quantity, price, etc.

Sell Medicine: This feature allows users to sell medicines to customers. It typically includes a form for entering the details of the sale, such as the medicine sold, quantity, customer information,print, etc.

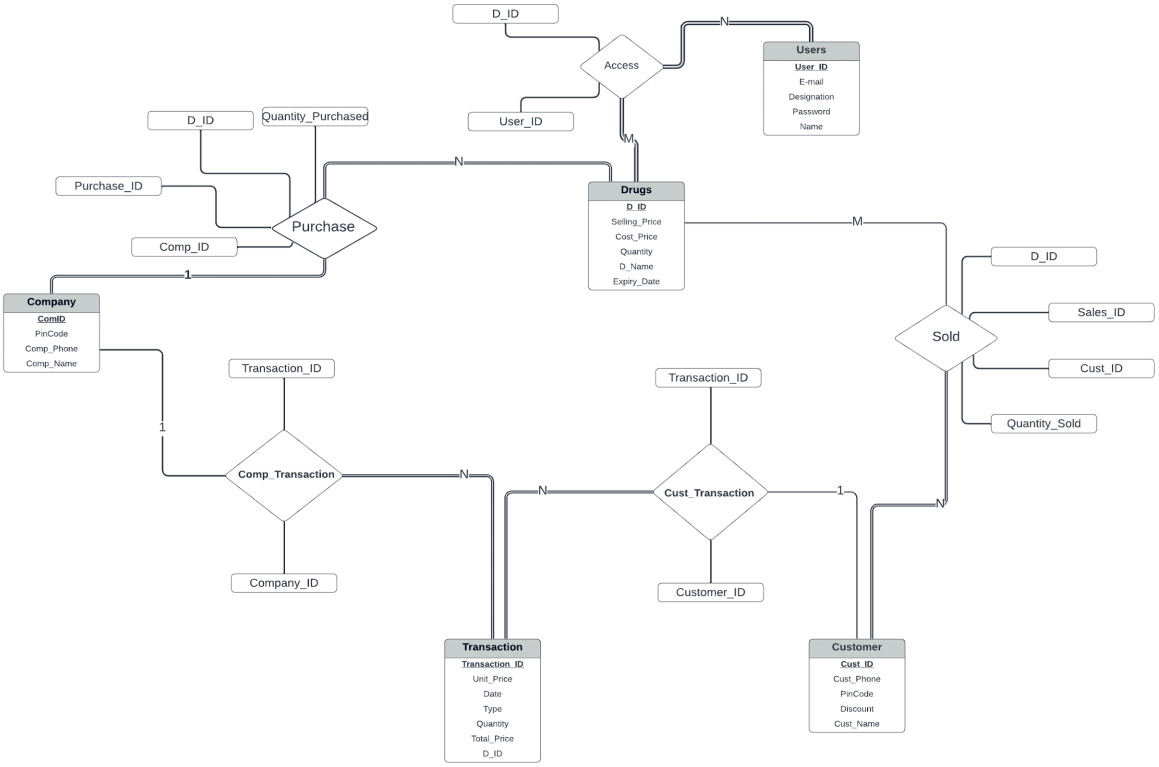
View Bill: This feature allows users to view the details of a sale, including the medicines sold, quantity, total amount, etc.

1.3 ER MODEL:



[Blank board: Lucidspark]

1.4 ER DIAGRAM:



[Database ER diagram (crow's foot): Lucidchart]

Relational Schema

- User(User_ID, Name, Email,Password, Designation)
- Company(ComID, Comp_Name, PinCode, Comp_Phone)
- Drugs(D_ID, D_Name, Quantity, Cost_Price, Selling_price, Expiry_Date)
- Customer(Cust_ID, Cust_Name, Discount, PinCode, Cust_Phone)
- Transaction(Transaction_ID, D_ID, Unit_Price, Date, Type,

  Total_Price, Quantity)
- Purchase(Comp_ID, Purchase_ID, D_ID, Quantity_Purchased)
- Access(User_ID,Company_ID)
- Comp_Transaction(Transaction_ID,Company_ID)
- Cust_Transaction(Transaction_ID,Customer_ID)
- Sold(D_ID,Sales_ID,Cust_ID,Quantity_Sold)

Explanation of entities:

a) **Drugs**: This entity stores the details of each drug bought and sold.

Attributes include:-

i) D_ID: stores the unique ID of a drug.
ii) D_name: stores the name of a drug

iii)Quantity: stores the quantity of drugs bought/sold

iv) Cost_price: stores price at which the drug is bought from the company
v) Selling_price: store price at which the drug is sold to the customer
vi) Expiry_Date: stores the expiry date of the drug Primary Key:- D_ID

b) **Company**: This entity stores the details of the companies that sell drugs to the pharmacy. Attributes include:-
i) Comp_ID:- stores the unique ID of the company.

ii) Comp_Name:-stores the name of the company.
iii) Pincode:-stores the physical location or mailing address of the company.
iv)Comp_Phone:-stores the contact number for the company.
Primary Key:- Comp_ID

c) **Customer**: This entity stores the information about the customers who buy drugs from the pharmacy.

Attributes include:-
i) Cust_ID:- stores the unique ID of the customer.
ii) Cust_Name:- stores the name of the customer.
iii)Pincode:- stores the customer's physical location or mailing address.
iv) Cust_Phone:-stores the contact number for the customer.
v) Discount:- stores the discount percentage granted to the customer.
Primary Key:-Cust_ID

d) **Transaction**:-records information related to the purchase or sale of drugs.

Attributes include:-
i) D_ID:- stores the unique ID of the drug involved in the transaction.
ii) Transaction_ID:- stores the unique ID for each transaction.
iii) Unit_price:- stores the cost or selling price of a single unit of drug involved in the transaction.
iv) Date:- stores the date when the transaction took place.
v) Type:- stores the nature of the transaction, indicating whether it is a purchase or sale.
vi)Total_Price:- stores the overall cost generated by the transaction.
vii)Quantity:- stores the quantity of drugs bought or sold.
Primary Key:-Transaction_ID

Foreign Keys:-
1) D_ID: referencing D_ID from Drugs.
2) Unit_Price: referencing Cost_Price and Selling_Price From Drugs

e) **Users**: This entity stores the information about the user who has logged into the system. Attributes include:-
i) User_ID:- stores the unique ID of the user.
ii) Name:- stores the name of the user.
iii) Password:- stores the password required by the user to log into the system.
iv) E-mail:- stores the email id of the user.
v) Designation:- stores the user's designation (which could be admin/employee).
Primary Key:- User_ID

Explanation of the Relationships:

a) **Purchase**: This relationship stores the drugs bought from the companies.

Cardinality constraints:- This is a M : 1 relationship as each company can sell zero or more drugs and each drug is associated with a single company.

Participation Constraint:- Each company must sell drugs so there is total participation from the company side. Each drug must be bought from a company, so there is total participation from the drug side too.

b) **Sold**:
Cardinality constraints:- This is a M:N relationship as each customer can buy zero or more drugs and each drug can be sold to zero or more customers.

Participation Constraint:- Each customer must buy drugs so there is total participation from the customer side. Each drug may or may not be sold to a customer, so there is partial participation from the drug side too.

c) **Comp_Transaction:**

Cardinality constraints:- This is a 1:N relationship as each company can be associated with zero or more transactions and each transaction is associated with only one company.

Participation Constraint:- Each company must participate in a transaction so there is total participation from the company side. However each transaction might involve a company, so there is partial participation from the transaction side.

d) **Cust_Transaction:**

Cardinality constraints:- This is a 1:N relationship as each customer can be associated with zero or more transactions and each transaction is associated with exactly one customer.

Participation Constraint:- Each customer must participate in a transaction so there is total participation from the customer side. Each transaction might involve a customer so there is partial participation from the transaction side.

e) **Access**:

Cardinality constraints:- This is a M:N relationship as each user can access many (all) drugs and each drug can be accessed by many users.

Participation constraints: Each user must access all drugs and all drugs must be accessed by users, so this is a total participation from both sides.

Conversion of ER to Relational Model:

1.Drugs.

D_id(PK)	D_name	Quantity	Cost_price	Selling_ Price	Expiry_ Date

2.Company:

Comp_ID(PK)	Comp_Name	Pincode	Comp_Phone

3.Customer:

Cust_ID(PK)	Cust_Name	Pincode	Cust_Phone	Discount

4.Transaction:

D_ID	Transaction_ID (PK)	Unit_ Price	Date	Type	Total_ Price	Quantity

5.Users:

User_ID(PK)	Name	Password	E-mail	Designation

6.Purchase:

Purchase_ID	D_ID	Comp_ID	Quantity_ Purchased

7.Comp_Transaction:

Transaction_ID	Company_ID

8.Cust_Transaction:

Customer_ID	Transaction_ID

9.Sold

Sales_ID	Cust_ID	D_ID	Quantity_Sold

10.Access:

User_ID	D_ID

Normalization:

To normalize the given ER model up to 3NF, we need to analyze functional dependencies and ensure that each non-prime attribute is fully dependent on the primary key.

● First Normal Form (1NF) : All tables are already in 1NF since there are no repeating groups.

● Second Normal Form (2NF) : All non-prime attributes are fully functionally dependent on the entire primary key, so this ER model is in 2NF.

● Third Normal Form (3NF) : There are no transitive dependencies in this table, this schema is now normalized up to 3NF.

Here are the final tables after normalization up to the 3NF:

User :

● User (User_ID, Name, Email, Password, Designation)

Company :

● Company (ComID, Comp_Name ,PinCode, Comp_Phone)

Drugs :

● Drugs (D_ID, D_Name, Quantity, Cost_Price, Selling_Price, Expiry_Date)

Customer :

● Customer (Cust_ID, Cust_Name, Discount, PinCode, Cust_Phone)

Transaction :

● Transaction (Transaction_ID, D_ID, Unit_Price, Date, Type, Total_Price, Quantity) Purchase (Composite key of Comp_ID and Purchase_ID) :

● Purchase (Comp_ID, Purchase_ID, D_ID, Quantity_Purchased)

Access (Composite key of User_ID and Company_ID) :

● Access (User_ID, Company_ID) Comp_Transaction :

● Comp_Transaction (Transaction_ID, Company_ID)

Cust_Transaction :

● Cust_Transaction (Transaction_ID, Customer_ID) Sold (Composite key of D_ID and Sales_ID) :
● Sold (D_ID, Sales_ID, Cust_ID, Quantity_Sold)

Domain constraints:

A domain constraint defines the allowable values for a specific attribute. It restricts the range of values that an attribute can take.

User_ID : varchar, not null
Name: varchar, default null
Email: varchar
Password: integer, default null
Designation: varchar
ComID: varchar, not null
Comp_name: varchar, default null
PinCode: integer
Comp_Phone: integer, default null
D_ID: varchar, not null
Selling_Price: integer
Cost_Price: integer
D_Name: varchar, default null
Expiry_Data: date time
Transaction_ID: varchar, not null
Unit Price: integer
Date: date time
Type: varchar
Quantity : integer
Total_Price: integer
Cust_ID: varchar, not null
Cust_Phone: integer,default null
Discount : integer
Cust_Name : varchar, default null

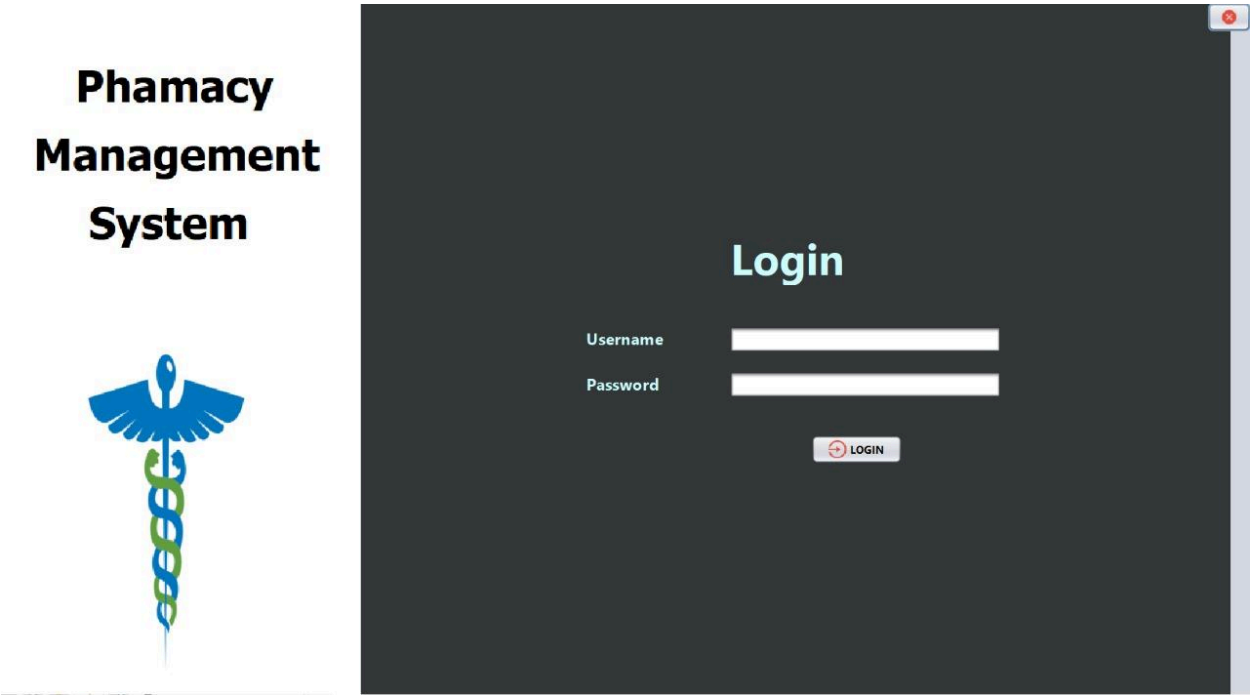
Quantity_Sold : integer
Quantity_Purchased : integer
Sales_ID : varchar, default null

1.5 TECHNICAL DETAILS:

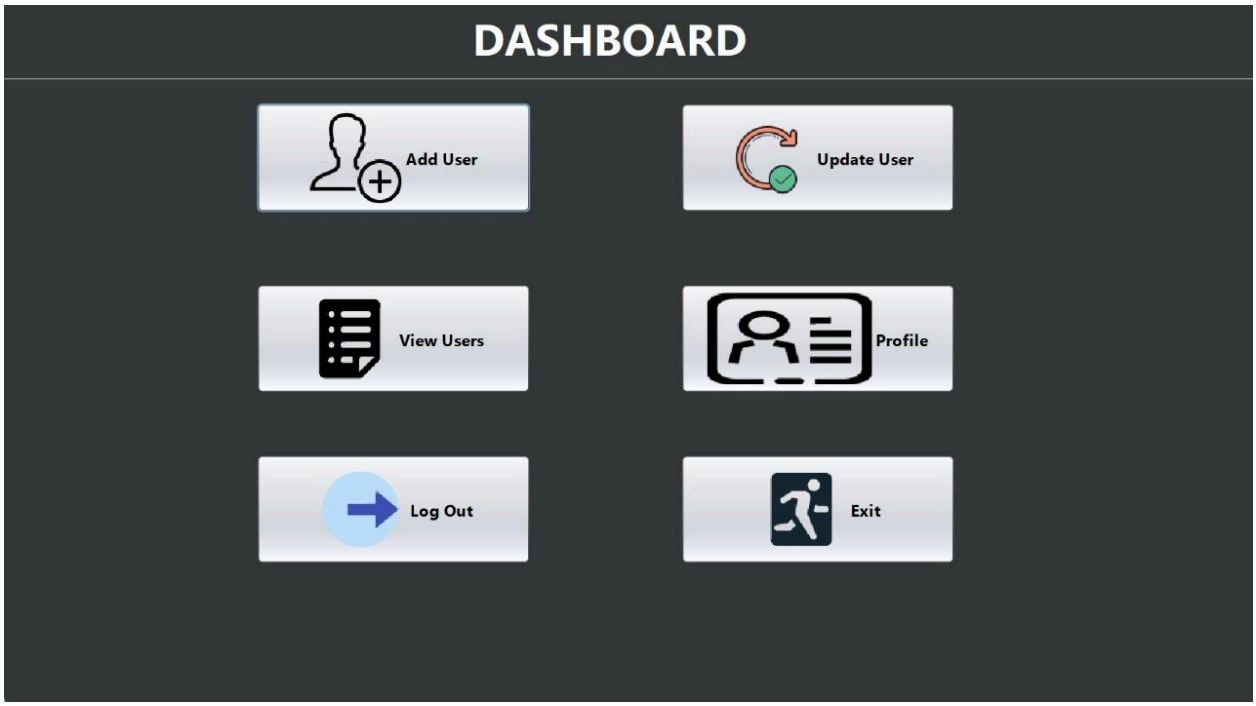
Project Front-End code

Project Panels

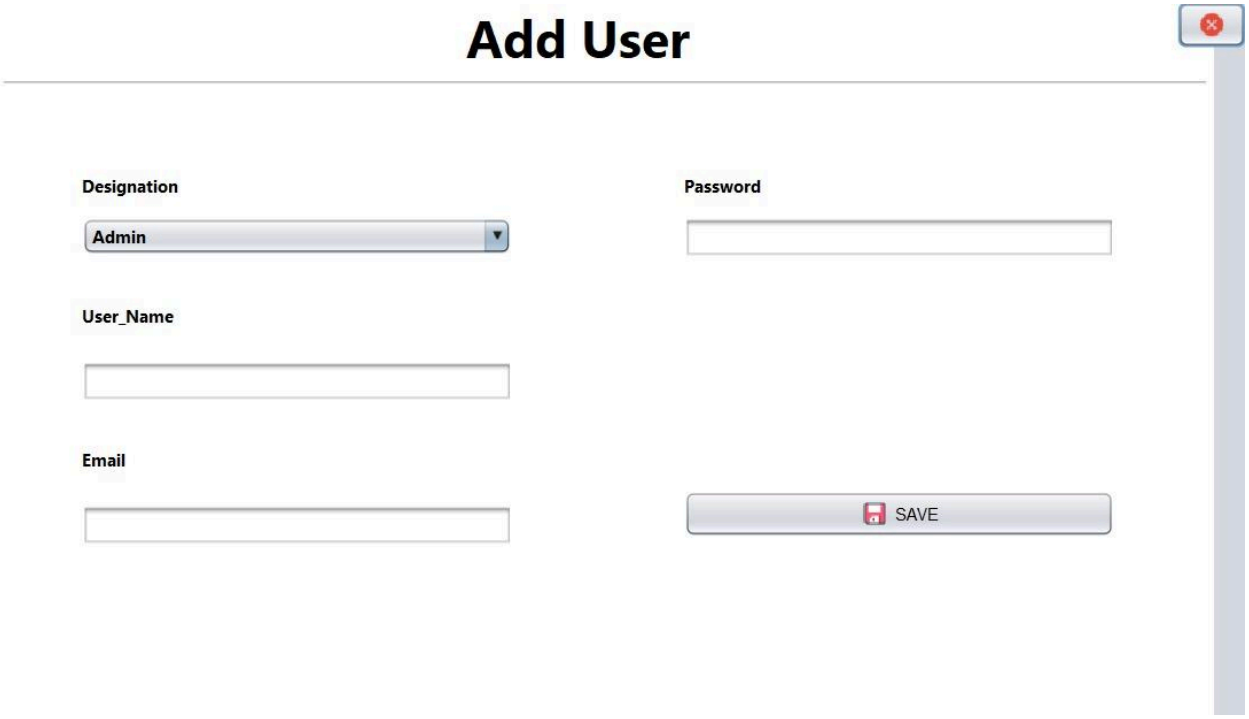
1. Login page:



2. Admin Dashboard:

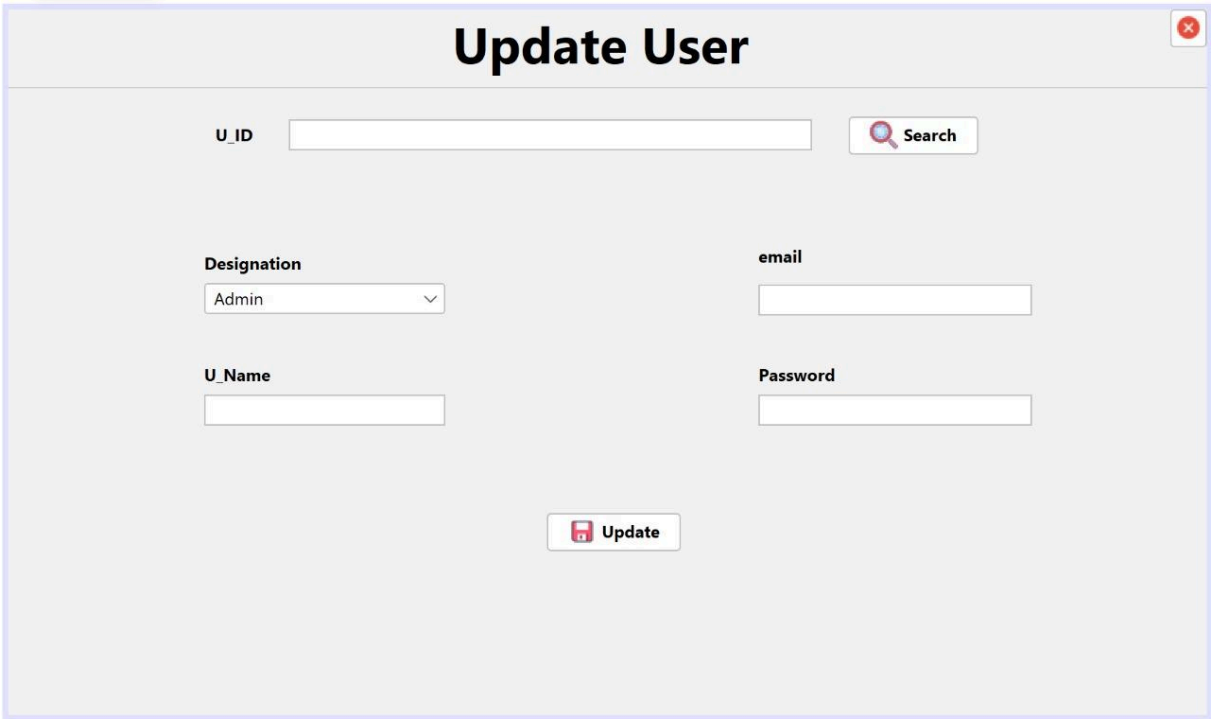


3. Add User

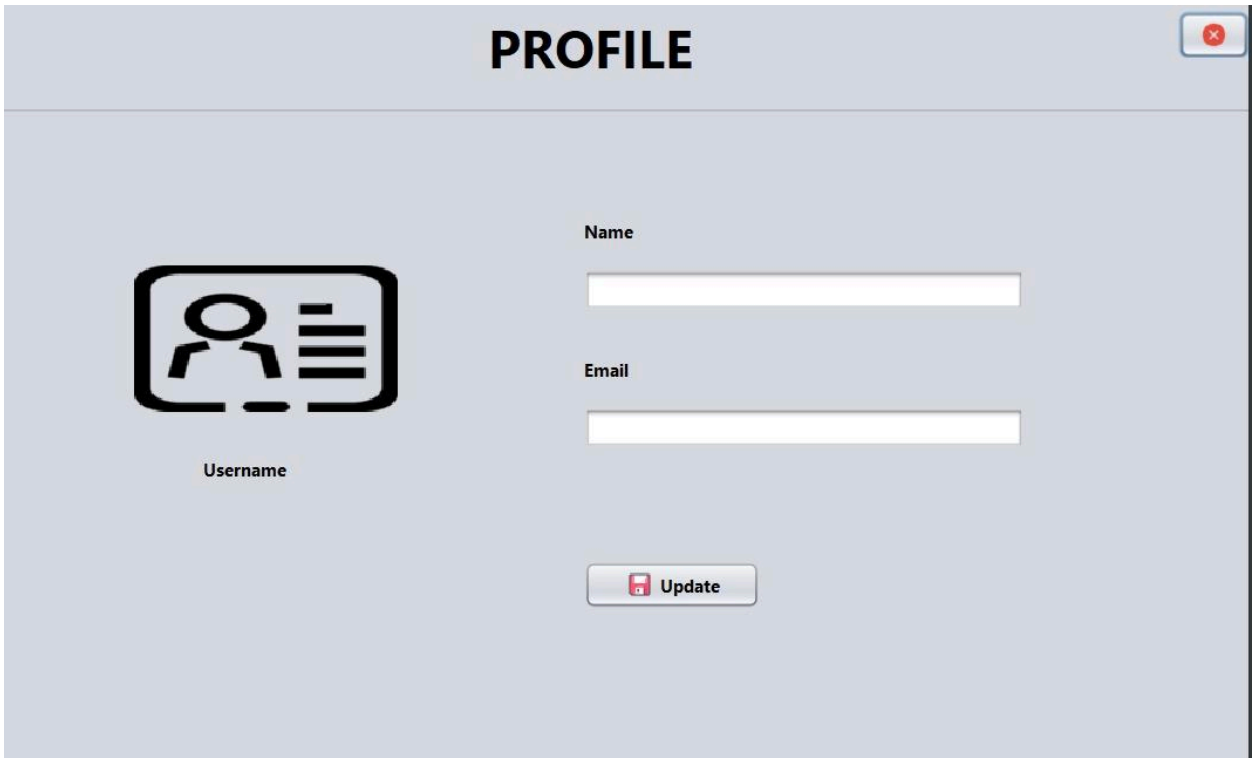


4. **View User**

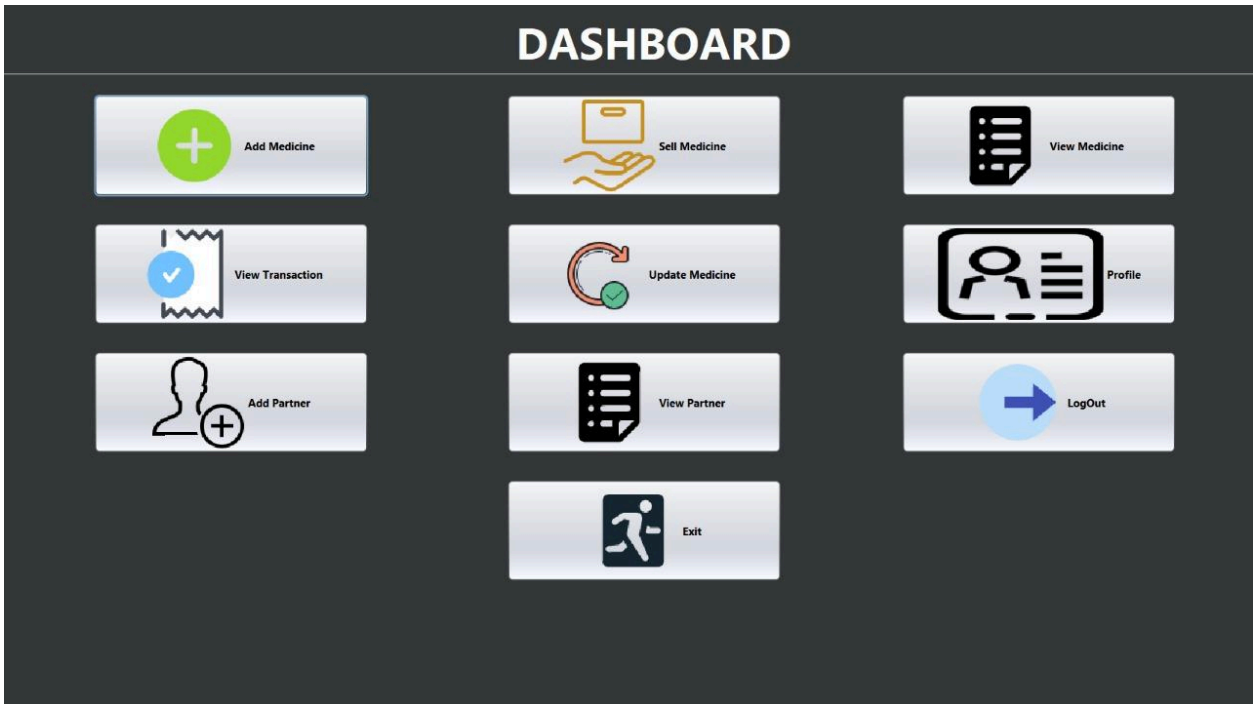
5. **Update user:**



6. **Profile:**



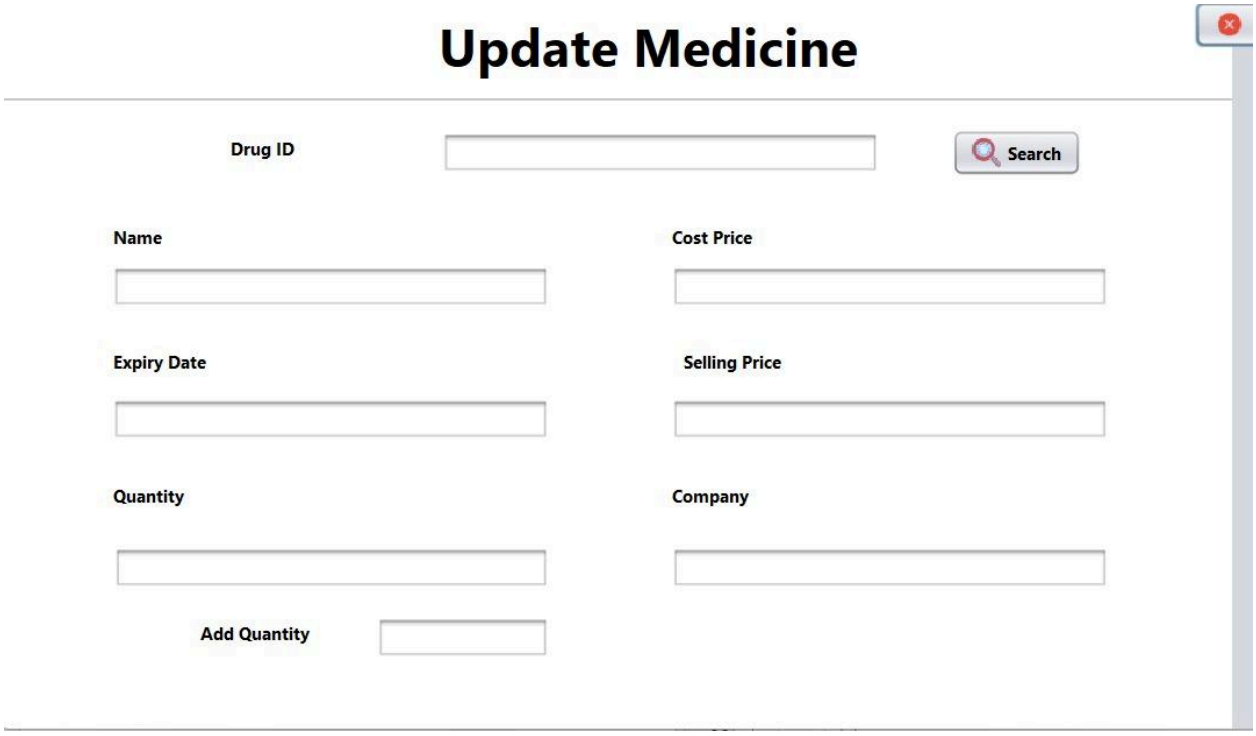
7. **Dashboard:**



8. **Add Medicine:**

9. **View Medicine:**

10. **Update Medicine:**



11. **Sell Medicine:**

12. **View Bill:**

SQL queries:

The following section contains the sample queries, theri implementation and their outputs.

1.Create all the necessary tables such as customer table, Drugs table, Transactions table etc.

CREATE QUERIES:

```
CREATE TABLE customer (
 Cust_ID varchar(20) NOT NULL,
 cust_name varchar(30) DEFAULT NULL,
```

```
  discount integer DEFAULT NULL,
  PinCode integer DEFAULT NULL,
  Cust_phone bigint DEFAULT NULL,
  PRIMARY KEY (Cust_ID) );


CREATE TABLE Drugs (
  D_ID varchar(15) NOT NULL,
  D_name varchar(20) DEFAULT NULL,
  Quantity integer default null,
  Expiry_Date date DEFAULT NULL,
  Cost_price decimal(10,2) DEFAULT NULL,
  Selling_price decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (D_ID),
  UNIQUE KEY D_name (D_name)  );

CREATE TABLE Transactions (
    Transaction_ID varchar(10) PRIMARY KEY,
    D_ID varchar(20),
    Unit_Price DECIMAL(10, 2),
    Date DATE,
    Type VARCHAR(50),
    Total_Price DECIMAL(10, 2),
    Quantity INTEGER,
    FOREIGN KEY (D_ID) REFERENCES Drugs(D_ID)  );

CREATE TABLE Purchase (
 Purchase_ID integer NOT NULL AUTO_INCREMENT,
 D_ID varchar(15) DEFAULT NULL,
 ComID varchar(15) DEFAULT NULL,
 Qty_purchased integer unsigned DEFAULT NULL,
 price integer unsigned DEFAULT NULL,
 PRIMARY KEY (Purchase_ID),
 FOREIGN KEY (ComID) REFERENCES Company(ComID),
```

```
 FOREIGN KEY (D_ID) REFERENCES Drugs(D_ID)  );

CREATE TABLE User (
  User_ID integer(20) NOT NULL,
  U_name varchar(50) DEFAULT NULL,
  email varchar(30) DEFAULT NULL,
  password varchar(30) DEFAULT NULL,
  designation varchar(50) DEFAULT NULL,
  PRIMARY KEY (User_ID)
  );

CREATE TABLE Company (
  ComID varchar(15) NOT NULL,
  Comp_name varchar(20) DEFAULT NULL,
  PinCode integer(128) DEFAULT NULL,
  Comp_phone bigint DEFAULT NULL,
  PRIMARY KEY (ComID)
);

CREATE TABLE Comp_Transaction (
    Transaction_ID varchar(10) NOT NULL,
    ComID varchar(20),
    FOREIGN KEY (Transaction_ID) REFERENCES
Transactions(Transaction_ID),
    FOREIGN KEY (ComID) REFERENCES Company(ComID),
    PRIMARY KEY (Transaction_ID, ComID)  );

CREATE TABLE Cust_Transaction (
    Transaction_ID varchar(10) NOT NULL,
    Cust_ID varchar(20) NOT NULL,
    FOREIGN KEY (Transaction_ID) REFERENCES
Transactions(Transaction_ID),
    FOREIGN KEY (Cust_ID) REFERENCES Customer(Cust_ID),
```

```
    PRIMARY KEY (Transaction_ID, Cust_ID)  );

CREATE TABLE Sold (
    D_ID varchar(20) NOT NULL,
    Sales_ID varchar(10),
    Cust_ID varchar(20) NOT NULL,
    Qty_Sold INT,
    FOREIGN KEY (D_ID) REFERENCES Drugs(D_ID),
    FOREIGN KEY (Cust_ID) REFERENCES Customer(Cust_ID)  );

create table Access(
D_ID varchar(20) NOT null,
User_ID integer NOT null,
foreign key (D_ID) references Drugs(D_ID),
foreign key (User_ID) references User(User_ID)  );
```

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:40:46	create database pharmacy	1 row(s) affected	0.015 sec
2	23:40:48	use Pharmacy	0 row(s) affected	0.000 sec
3	23:41:05	CREATE TABLE User (User_ID integer NOT NULL, U_name varchar(50) DEFAULT NULL, em...	0 row(s) affected	0.016 sec
4	23:41:09	CREATE TABLE customer (Cust_ID varchar(20) NOT NULL, cust_name varchar(30) DEFAULT ...	0 row(s) affected	0.015 sec
5	23:41:11	CREATE TABLE Company (ComID varchar(15) NOT NULL, Comp_name varchar(20) DEFAULT...	0 row(s) affected	0.015 sec
6	23:41:14	CREATE TABLE Drugs (D_ID varchar(15) NOT NULL, D_name varchar(20) DEFAULT NULL, ...	0 row(s) affected	0.031 sec
7	23:41:18	CREATE TABLE Transactions (Transaction_ID varchar(10) PRIMARY KEY, D_ID varchar(20...	0 row(s) affected	0.063 sec
8	23:41:22	CREATE TABLE Purchase (Purchase_ID integer NOT NULL AUTO_INCREMENT, D_ID varcha...	0 row(s) affected	0.047 sec
9	23:41:26	CREATE TABLE Comp_Transaction (Transaction_ID varchar(10), ComID varchar(20), FO...	0 row(s) affected	0.047 sec
10	23:41:29	CREATE TABLE Cust_Transaction (Transaction_ID varchar(10), Cust_ID varchar(20), FO...	0 row(s) affected	0.063 sec
11	23:41:33	CREATE TABLE Sold (D_ID varchar(20), Sales_ID varchar(10), Cust_ID varchar(20), Qt...	0 row(s) affected	0.062 sec
12	23:41:43	create table Access(D_ID varchar(20) not null, User_ID integer not null, foreign key (D_ID) referen...	0 row(s) affected	0.063 sec

INSERT QUERIES

2.Insert a new user record:

```
INSERT INTO `User` (`User_ID`, `U_name`, `email`, `password`, `designation`) VALUES
(1, 'Tont Stark', 'stark@mail', 3000, 'admin'),
(2, 'mark', 'mark@mail', 2000, 'employee'),
(3, 'clark', 'clark@mail', 4000, 'employee'),
(4, 'Adam', 'pich@mail', 50000, 'employee');
```

Output:

```
9 •   select * from user;
10
```

Result Grid | Filter Rows: | Edit:

User_ID	U_name	email	password	designation
1	Tont Stark	stark@mail	3000	admin
2	mark	mark@mail	2000	employee
3	clark	clark@mail	4000	employee
4	Adam	pich@mail	50000	employee
NULL	NULL	NULL	NULL	NULL

3. Insert a new company record:

```
INSERT INTO `company` (`ComID`,`Comp_name`, `PinCode`, `Comp_phone`) VALUES
(1,'Cipla', 12340, '12903'),
(2,'Sun Pharma', 56780, '01289078443'),
(3,'Med_City', 67890, '010114367832');
```

Output:

```
select * from company;
```

Result Grid | Filter Rows: | Edit

ComID	Comp_name	PinCode	Comp_phone
1	Cipla	12340	12903
2	Sun Pharma	56780	1289078443
3	Med_City	67890	10114367832
NULL	NULL	NULL	NULL

UPDATE QUERIES:

4.Update a drug discovery:

```
select * from drugs;
```

Result Grid | Filter Rows: | Edit: | Export/

D_ID	D_name	Quantity	Expiry_Date	Cost_price	Selling_price
1	Novalo	40	2025-03-03	200.00	300.00
2	Asprin	12	2025-04-06	50.00	75.00
3	Dolo	5	2027-02-19	100.00	135.00
4	Azithromycin	32	2027-12-08	300.00	345.00
5	Citrazine	50	2024-08-09	100.00	120.00
6	Paracetamol	40	2026-08-06	70.00	90.00
7	novafol	27	2026-01-01	330.00	400.00
NULL	NULL	NULL	NULL	NULL	NULL

output:

```
update drugs set quantity = 20 where D_name like 'A%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

D_ID	D_name	Quantity	Expiry_Date	Cost_price	Selling_price
1	Novalo	40	2025-03-03	200.00	300.00
2	Asprin	20	2025-04-06	50.00	75.00
3	Dolo	5	2027-02-19	100.00	135.00
4	Azithromycin	20	2027-12-08	300.00	345.00
5	Citrazine	50	2024-08-09	100.00	120.00
6	Paracetamol	40	2026-08-06	70.00	90.00
7	novafol	27	2026-01-01	330.00	400.00
NULL	NULL	NULL	NULL	NULL	NULL

DELETE QUERIES:

5.Delete a drug record:

Before

```
select * from drugs;
```

Result Grid | Filter Rows: | Edit: | Export/

D_ID	D_name	Quantity	Expiry_Date	Cost_price	Selling_price
1	Novalo	40	2025-03-03	200.00	300.00
2	Asprin	12	2025-04-06	50.00	75.00
3	Dolo	5	2027-02-19	100.00	135.00
4	Azithromycin	32	2027-12-08	300.00	345.00
5	Citrazine	50	2024-08-09	100.00	120.00
6	Paracetamol	40	2026-08-06	70.00	90.00
7	novafol	27	2026-01-01	330.00	400.00
NULL	NULL	NULL	NULL	NULL	NULL

After

```
15 •    delete from drugs where D_name like 'dolo';
```

Result Grid | Filter Rows: | Edit: | Export

D_ID	D_name	Quantity	Expiry_Date	Cost_price	Selling_price
1	Novalo	40	2025-03-03	200.00	300.00
2	Asprin	20	2025-04-06	50.00	75.00
4	Azithromycin	20	2027-12-08	300.00	345.00
5	Citrazine	50	2024-08-09	100.00	120.00
6	Paracetamol	40	2026-08-06	70.00	90.00
7	novafol	27	2026-01-01	330.00	400.00
NULL	NULL	NULL	NULL	NULL	NULL

(Complex queries)

1.List all transactions along with customer and drug details:-

```
SELECT T.Transaction_ID, T.Date, T.Type, T.Total_Price, C.Cust_ID,
      D.D_Name, D.Quantity, D.Cost_Price, D.Selling_price
FROM Transactions T
JOIN Cust_Transaction C ON T.transaction_ID = C.Transaction_ID
JOIN Drugs D ON T.D_ID = D.D_ID;
```

<u>Output:</u>

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Transaction_ID	Date	Type	Total_Price	Cust_ID	D_Name	Quantity	Cost_Price	Selling_price
2002	2024-04-15	Purchase	500.00	1002	Asprin	20	50.00	75.00
2003	2024-04-15	Sale	1080.00	1004	Azithromycin	20	300.00	345.00
2004	2024-04-14	Sale	3000.00	1003	Novalo	40	200.00	300.00

2.List all expired drugs:-

SELECT * FROM Drugs WHERE Expiry_Date < CURDATE();

<u>Output:</u>

Result Grid | Filter Rows: | Edit: | Export

D_ID	D_name	Quantity	Expiry_Date	Cost_price	Selling_price
1	Novalo	40	2023-02-02	200.00	300.00
7	novafol	27	2023-02-02	330.00	400.00
NULL	NULL	NULL	NULL	NULL	NULL

3.Find the total sales made by each company along with company name:

```
/*Find the total sales made by each company along with the company name: */
SELECT co.Comp_name, SUM(t.Total_Price) AS Total_Sales
FROM Company co
JOIN Comp_Transaction ct ON co.ComID = ct.ComID
JOIN Transactions t ON ct.Transaction_ID = t.Transaction_ID
WHERE t.Type = 'Purchase'
GROUP BY co.Comp_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Comp_name	Total_Sales
Sun Pharma	500.00

4.List all transactions for a specific company:-

```
SELECT T.Transaction_ID, T.Date, T.Type, T.Total_Price,
       D.D_Name, D.Quantity, D.Cost_Price, D.Selling_price
FROM Transactions T
JOIN Purchase P ON T.D_ID = P.D_ID
JOIN Company C ON P.ComID = C.ComID
JOIN Drugs D ON T.D_ID = D.D_ID
WHERE C.Comp_Name = 'YourCompany';
```

```sql
/* List all transactions for a specific company: */
SELECT T.Transaction_ID, T.Date, T.Type, T.Total_Price,
       D.D_Name, D.Quantity, D.Cost_Price, D.Selling_price
FROM Transactions T
JOIN Purchase P ON T.D_ID = P.D_ID
JOIN Company C ON P.ComID = C.ComID
JOIN Drugs D ON T.D_ID = D.D_ID
WHERE C.ComID = 1 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Transaction_ID	Date	Type	Total_Price	D_Name	Quantity	Cost_Price	Selling_price
2003	2024-04-15	Sale	1080.00	Azithromycin	20	300.00	345.00

5.Calculate the total profit for a specific time period:-

SELECT SUM(T.Total_Price - D.Cost_Price) AS Total_Profit
FROM Transactions T
JOIN Drugs D ON T.D_ID = D.D_ID
WHERE T.Date BETWEEN 'start_date' AND 'end_date';

<u>Output:</u>

```sql
/* Calculate the total profit for a specific time period: */
SELECT SUM(T.Total_Price - D.Cost_Price) AS Total_Profit
FROM Transactions T
JOIN Drugs D ON T.D_ID = D.D_ID
WHERE T.Date BETWEEN '2023-10-10' AND '2024-04-15';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Profit
4030.00

6.Calculate total sales amount of each customer:-

SELECT CT.Cust_ID, SUM(T.Total_Price) AS Total_Sales_Amt
FROM Transactions T
JOIN Cust_Transaction C ON T.Transaction_ID = C.Transaction_ID

```
JOIN Customer CT ON CT.Cust_ID = C.Cust_ID
GROUP BY CT.Cust_ID;
```

Output:

Result Grid | Filter Rows:

Cust_ID	Total_Sales_Amt
1002	500.00
1004	1080.00
1003	3000.00

Conclusion

The Pharmacy Management System created by group 33 is designed to efficiently manage various aspects of a pharmacy, including inventory, sales, customer records, and financial reporting. It provides a user-friendly interface for employees and administrators to perform their tasks securely and efficiently.

The system's key features include user authentication, purchase management, sales management, payment reporting, inventory handling, and a dashboard for an overview of key metrics. These features enable the system to streamline pharmacy operations, improve inventory management, and enhance customer service.

The database schema is designed to store data in a normalized form up to the 3rd Normal Form (3NF), ensuring data integrity and minimizing redundancy. Tables are created for users, companies, drugs, customers,

transactions, purchases, access control, and sales, with appropriate foreign key relationships to maintain data consistency.

The SQL queries provided demonstrate the creation of tables, insertion of data, updating records, and deleting records, showcasing the system's functionality. Overall, the Pharmacy Management System offers a comprehensive solution for managing wholesale pharmacy operations effectively and efficiently.

Key achievements:

1. Efficient Inventory Management: The system allows for tracking stock levels and expiry dates of medicines, ensuring that the right amount of inventory is maintained. This helps in avoiding stockouts and overstock situations, ultimately leading to cost savings and improved customer satisfaction.

2. Improved Sales Tracking: With the ability to record sale transactions to customers, the system provides valuable insights into sales performance and customer behavior. This information can be used to optimize sales strategies and improve profitability.

3. Enhanced Financial Reporting: The system generates detailed payment reports for purchases, sales, and profits. These reports provide a clear picture of the pharmacy's financial health, helping in better decision-making and planning for the future.

4. Secure User Authentication: The system ensures that only authorized users can access sensitive information, such as employees and administrators. This helps in maintaining data security and confidentiality.

5. User-Friendly Interface: The system provides a user-friendly interface for employees and administrators to perform their tasks efficiently. This includes features like the admin dashboard, user profiles, and easy access to relevant information.

6. Streamlined Operations: By integrating various aspects of pharmacy management into a single system, the Pharmacy Management System streamlines operations and reduces the need for manual intervention. This leads to improved efficiency and productivity.