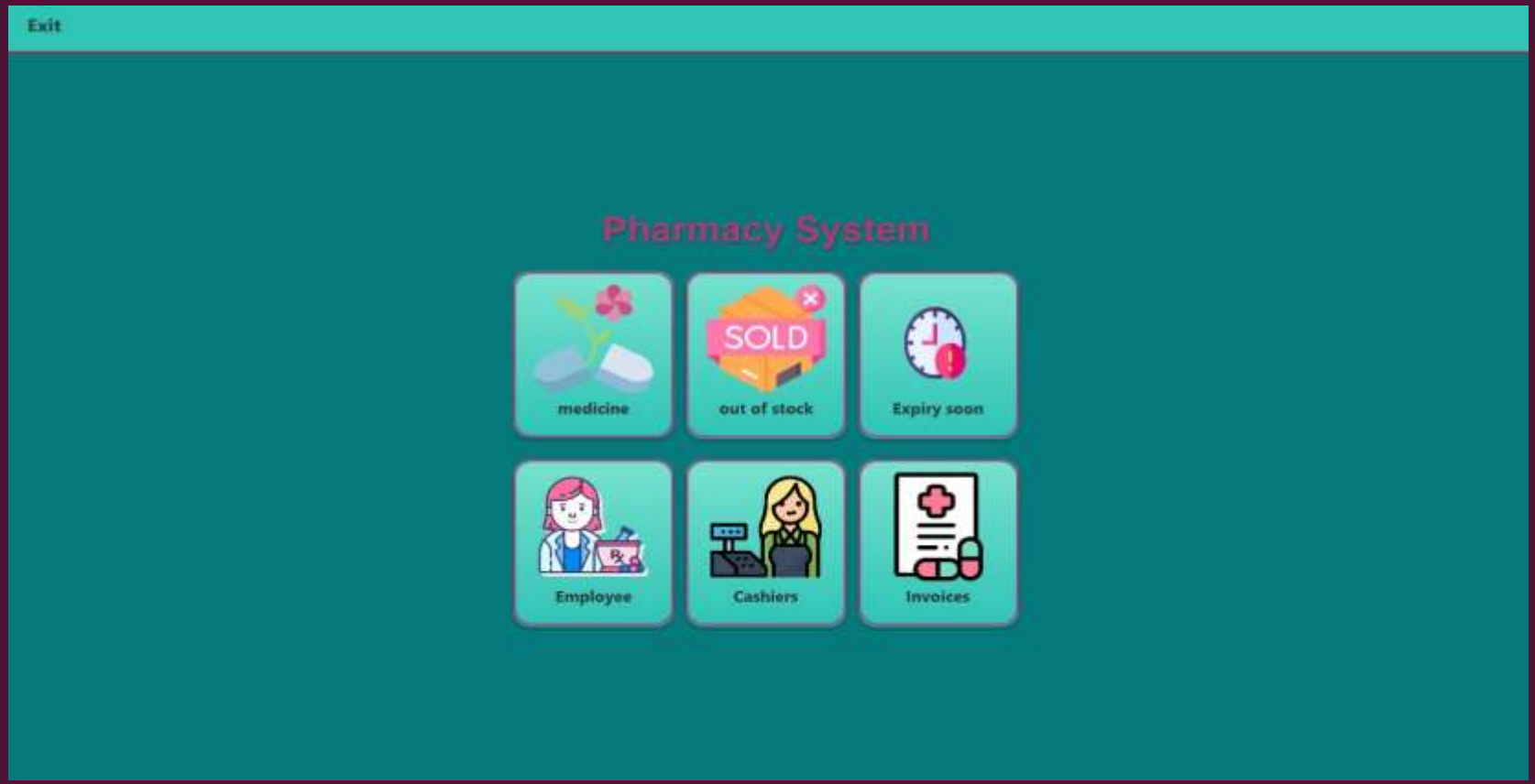
PHARMACY MANAGEMENT SYSTEM

PARTNER NAME : TALA YASER.
PARTNER ID : 1230412.
MY NAME : SHAYMAA ELAYAN.
MY ID : 1232590.
THE DOCTOR:NAIMEH HIRBAWI

PHARMACY SYSTEMS



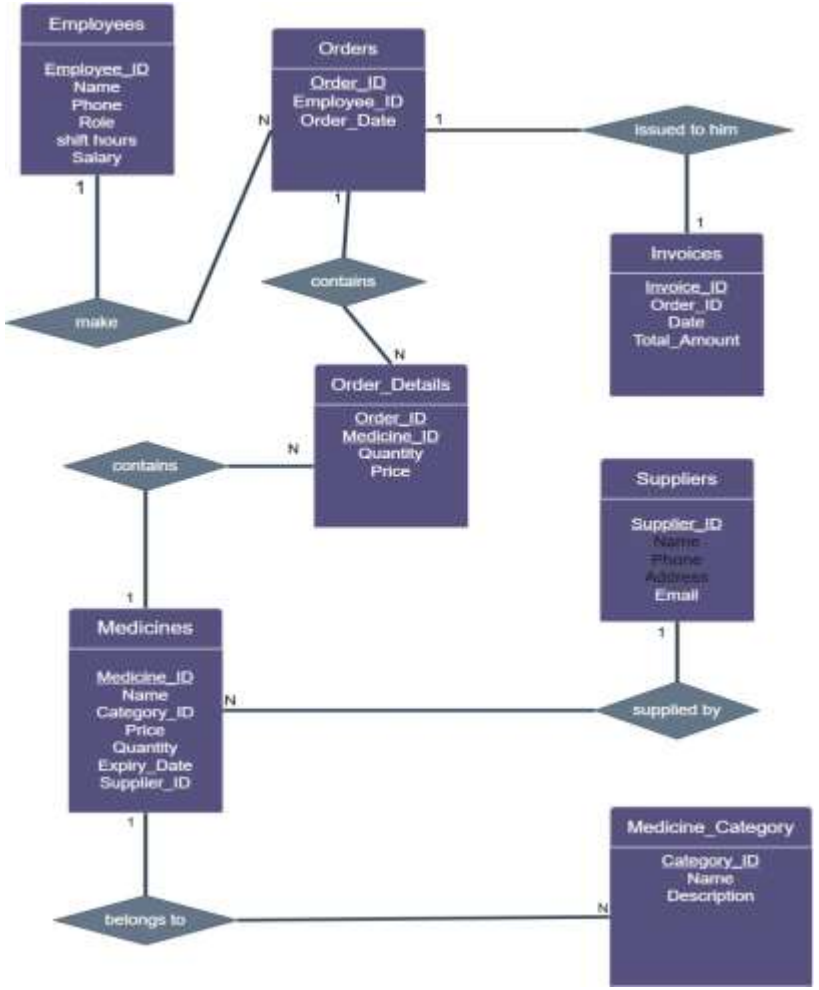
INTRODUCTION

- This project is a Pharmacy Management System developed as the final requirement for the course. The system is designed to assist pharmacists in managing medicine inventory, tracking sales, monitoring expiry dates, and handling customer transactions efficiently. The goal of the project is to automate the basic operations of a pharmacy and reduce the chances of human errors.
- The system provides features to add, update, and delete medicine records, manage suppliers, record sales transactions, and generate various reports related to stock levels and total sales. The project is developed using Java with a graphical user interface, and MySQL is used as the backend database.

PROJECT SCOPE

- *The Pharmacy Management System provides core functionalities required for daily pharmacy operations. The main modules and features include:*
- *- **Medicine Management**: Add, update, delete, and search for medicines. Each medicine includes details such as name, quantity, expiry date, price, and supplier.*
- *- **Order Details Management**: Record sales transactions, including the medicine sold, quantity, total price, and date of sale.*
- *- **Supplier Management**: Manage supplier details including name, contact information, and the medicines they supply.*
- *- **Employee and Cashier Management**: Add and manage employee and cashier records, including their roles and login credentials.*
- *- **Stock Monitoring**: Automatically track stock quantity and generate low-stock alerts.*
- *- **Reports and Queries**:*
- *- View total sales in a specific period.*
- *- Display expired medicines.*
- *- Check out-of-stock items.*
- *- Generate daily and monthly sales summaries.*

ER- DIAGRAM


Employees
Employee_ID
Name
Phone
Role
shift hours
Salary
Orders
Order_ID
Employee_ID
Order_Date
issued to him
Invoices
Invoice_ID
Order_ID
Date
Total_Amount
make
contains
Order_Details
Order_ID
Medicine_ID
Quantity
Price
contains
Suppliers
Supplier_ID
Name
Phone
Address
Email
Medicines
Medicine_ID
Name
Category_ID
Price
Quantity
Expiry_Date
Supplier_ID
supplied by
belongs to
Medicine_Category
Category_ID
Name
Description
1
N

CONVERSION TO RELATIONAL TABLES

```sql
-- table Medicines
CREATE TABLE  Medicines(
  Medicines_ID INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  Category_ID INT,
  Price DECIMAL(10,2) NOT NULL,
  Quantity INT NOT NULL,
  Expiry_Date DATE,
  Supplier_ID INT,
  FOREIGN KEY (Category_ID) REFERENCES Medicine_Category(Category_ID),
  FOREIGN KEY (Supplier_ID) REFERENCES Suppliers(Supplier_ID)
);
```

```sql
-- table Medicine_Category
CREATE TABLE Medicine_Category(
  Category_ID INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  Description TEXT
);
```

CONVERSION TO RELATIONAL TABLES

```sql
-- table Suppliers (only once)
CREATE TABLE Suppliers(
  Supplier_ID INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  phone VARCHAR(20),
  Address TEXT,
  Email VARCHAR(100)
);
```

```sql
-- table Employee
CREATE TABLE Employee (
  Employee_ID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  phone VARCHAR(20),
  role VARCHAR(50),
  shift_hours INT,
  salary DECIMAL(10,2)
);
```

CONVERSION TO RELATIONAL TABLES

```sql
-- table Invoices
CREATE TABLE Invoices(
  Invoice_ID INT AUTO_INCREMENT PRIMARY KEY,
  Order_ID INT,
  date DATETIME DEFAULT CURRENT_TIMESTAMP,
  Total_Amount DECIMAL(10,2) NOT NULL,
  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID) ON DELETE CASCADE
);
```

```sql
-- table Order_Details
CREATE TABLE Order_Details(
  Order_ID INT,
  Medicines_ID INT,
  Quantity INT NOT NULL,
  Price DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (Order_ID, Medicines_ID),
  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID) ON DELETE CASCADE,
  FOREIGN KEY (Medicines_ID) REFERENCES Medicines(Medicines_ID) ON DELETE CASCADE
);
```

CONVERSION TO RELATIONAL TABLES

```
-- table Orders
CREATE TABLE Orders(
  Order_ID INT AUTO_INCREMENT PRIMARY KEY,
  Employee_ID INT,
  Order_Date DATE NOT NULL,
  FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID) ON DELETE CASCADE
);
```

NORMALIZATION

- *First Normal Form (1NF) was achieved by ensuring that each field in the tables contains atomic values. For example, in the Order_Details table, each row represents a single medicine associated with a specific order, instead of storing multiple medicines in a single field. This guarantees that all columns contain indivisible values and that the data is well-structured for processing and queries. Second Normal Form (2NF) was satisfied by eliminating partial dependencies in tables with composite primary keys. For instance, in the Order_Details table, only attributes that are fully dependent on both Order_ID and Medicine_ID were included, while details like Medicine_Name were placed in the Medicine table. Third Normal Form (3NF) was also achieved by removing transitive dependencies. Each non-key attribute depends only on the primary key. For example, in the Supplier table, contact details and address information depend directly on Supplier_ID, ensuring logical and minimal table structure.*