NATIONAL ENGINEERING COLLEGE, K.R.NAGAR, KOVILPATTI - 628 503

(An Autonomous Institution – Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

23CS45C – DATABASE MANAGEMENT SYSTEM

A

MICRO PROJECT

Report on

PHARMACY MANAGEMENT SYSTEM

Team Members:	Course Instructor
Dhanvanth S-2312011	D. VIJAYAKUMAR
Immanuel A-2312020	Asst. Prof (Sr.Grade)
Kishore V-2312104	Dept of CSE, NEC
Subash Chandra Bose K-2312403	

INTRODUCTION

In the rapidly evolving healthcare industry, effective and reliable systems are essential for managing pharmacies. The Pharmacy Management System is a database-driven application designed to handle the daily operations of a pharmacy. This includes managing medicine inventory, customer records, and sales transactions. The system aims to simplify data storage and retrieval, minimize human errors, and enhance the speed of service delivery.

This project utilizes the principles of Database Management Systems (DBMS) to ensure efficient handling of structured data. With the help of MySQL as the backend database and Java (using JDBC or Spring Boot) as the frontend or interface layer, the system offers a practical implementation of CRUD (Create, Read, Update, Delete) operations on pharmacy data.

OBJECTIVE

The Pharmacy Management System aims to revolutionize traditional pharmacy operations through comprehensive digital transformation, creating a streamlined and efficient workflow environment for pharmacy professionals. This Java-based application will serve as an integrated solution to address the multifaceted challenges of modern pharmacy management, from inventory control to patient care. By digitalizing prescription processing, medication tracking, and patient record management, the system will significantly reduce manual paperwork and minimize human error. The project focuses on developing a secure, user-friendly interface that enables pharmacists and staff to quickly access critical information, process transactions, and maintain regulatory compliance while improving overall patient service quality. Beyond operational efficiency, the system will provide valuable business intelligence through customizable reporting and analytics, allowing pharmacy managers to make data-driven decisions regarding inventory optimization, sales trends, and business growth. Through meticulous database design and robust Java implementation, this project will deliver a scalable and sustainable solution that can adapt to evolving pharmaceutical regulations while maintaining the highest standards of data security and patient confidentiality. Ultimately, the Pharmacy Management System aims to enhance medication safety, increase operational productivity, and elevate the standard of pharmaceutical care delivery.

ER DIAGRAM

An Entity-Relationship (ER) diagram is a visual representation of the data structure and relationships within a database system. The ER diagram you provided for your Pharmacy Management System illustrates the conceptual schema of your database design, showcasing how different data elements relate to one another. This diagramming technique uses specific symbols and notations to represent entities, attributes, and relationships.

1. **Medicine Entity**: Contains key attributes for medication inventory management:
   - id: Unique identifier for each medicine
   - name: Name of the medication
   - price: Cost of the medicine
   - quantity: Available stock amount
   - expiry_date: When the medication expires
   - manufacture: The pharmaceutical company that produces it
2. **Customer Entity**: Stores essential customer information:
   - id: Unique identifier for each customer
   - address: Physical location of the customer
   - email: Customer's email contact
   - phone: Customer's phone number
3. **Sales Relationship**: The diamond shape represents the relationship between Medicine and Customer entities. This indicates transactions where customers purchase medicines, connecting the two main entities in a many-to-many relationship (a customer can buy multiple medicines, and a medicine can be purchased by multiple customers).

This diagram provides a simple but effective structure for tracking medicine inventory and customer information, with sales transactions linking them together

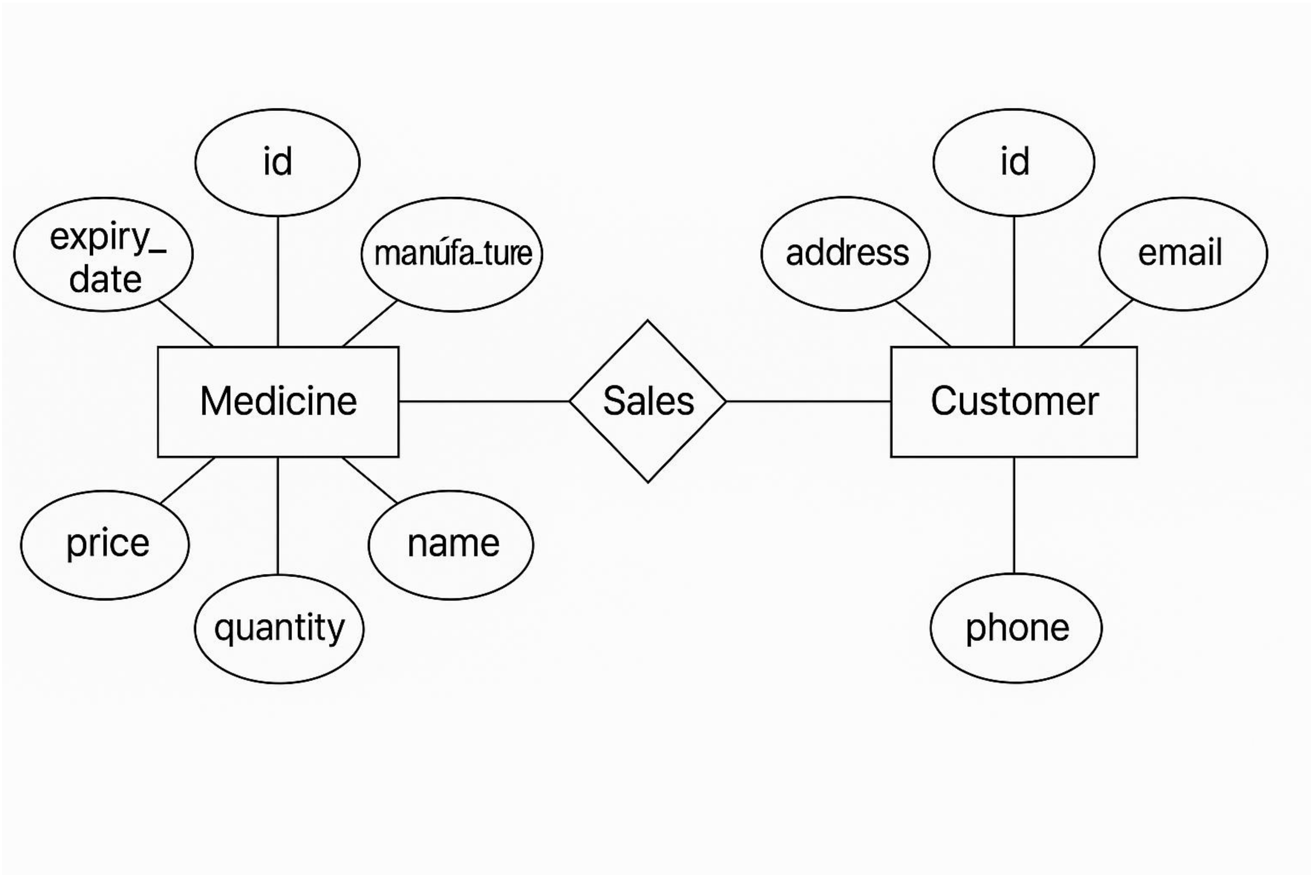
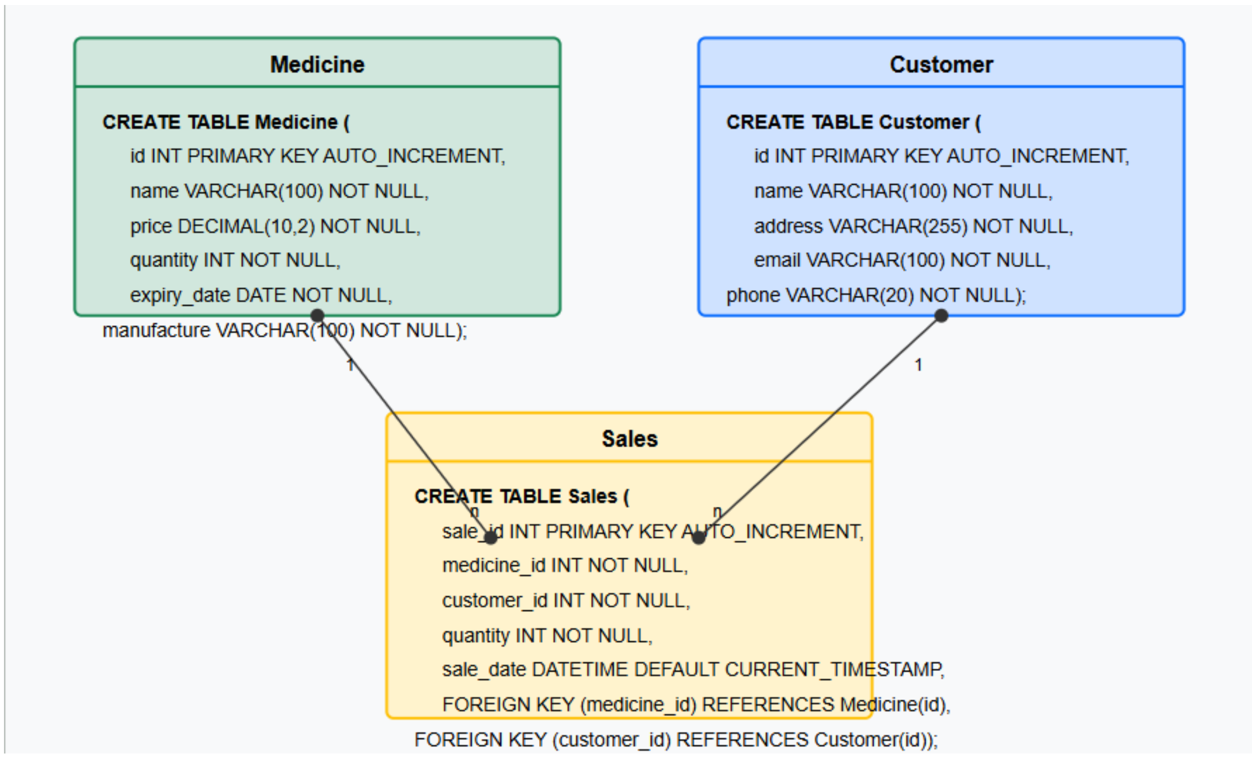


Fig 1: Entity Relationship Diagram for Pharmacy Management System.

SCHEMA DIAGRAM



```
CREATE TABLE Medicine (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    expiry_date DATE NOT NULL,
manufacture VARCHAR(100) NOT NULL);
```

```
CREATE TABLE Customer (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL,
phone VARCHAR(20) NOT NULL);
```

```
CREATE TABLE Sales (
    sale_id INT PRIMARY KEY AUTO_INCREMENT,
    medicine_id INT NOT NULL,
    customer_id INT NOT NULL,
    quantity INT NOT NULL,
    sale_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (medicine_id) REFERENCES Medicine(id),
FOREIGN KEY (customer_id) REFERENCES Customer(id));
```

Fig 2: Schema Diagram

This schema directly implements the ER diagram you provided earlier and includes the attributes you've specified. The Sales table serves as the junction table that implements the many-to-many relationship between Medicine and Customer.

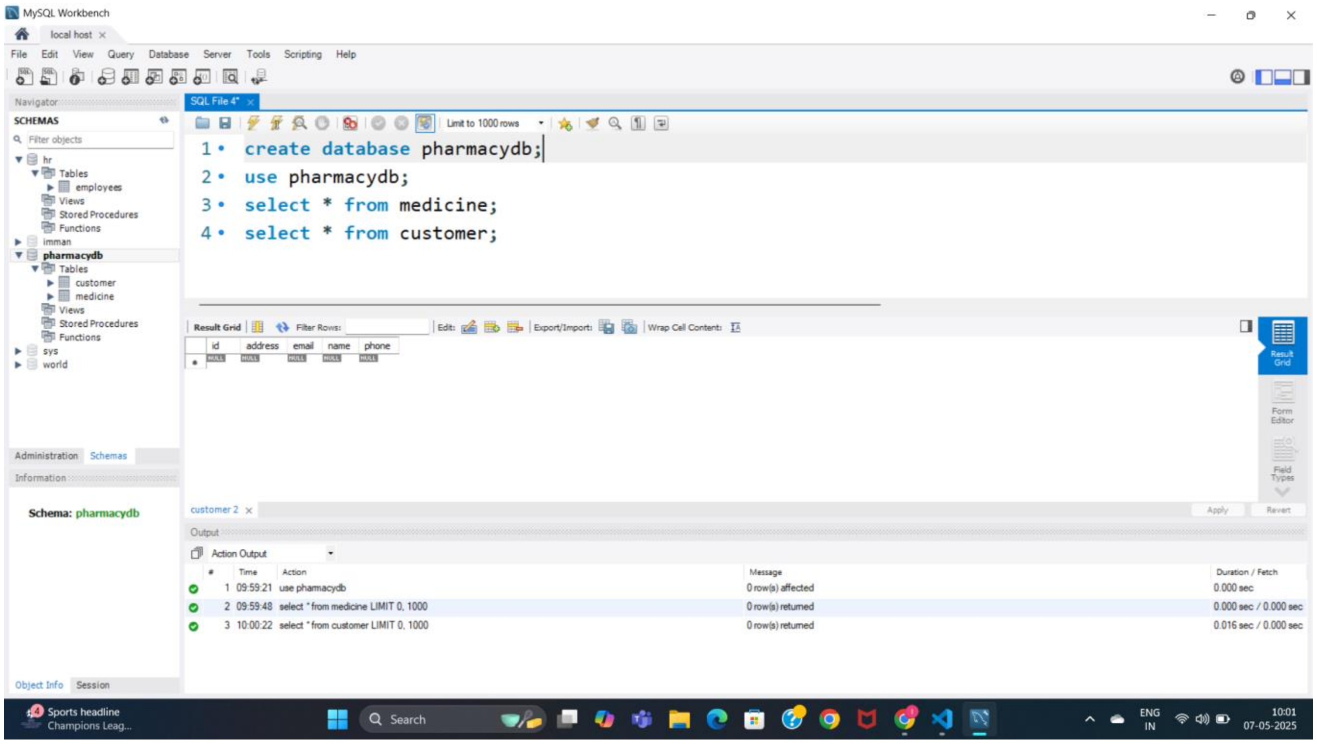
1. **Medicine Table**:
    - Primary Key: id
    - Attributes:
        - name (VARCHAR)
        - price (DECIMAL)
        - quantity (INT)
        - expiry_date (DATE)

        - manufacture (VARCHAR)

2. **Customer Table**:
    - Primary Key: id
    - Attributes:
        - name (VARCHAR) - note that I added this attribute as specified in your request
        - address (VARCHAR)
        - email (VARCHAR)
        - phone (VARCHAR)
3. **Sales Table** (Junction Table):
    - Primary Key: sale_id
    - Foreign Keys:
        - medicine_id (references Medicine.id)
        - customer_id (references Customer.id)
    - Other Attributes:
        - quantity (INT)
        - sale_date (DATETIME)

The relationships show:

- One medicine can be in many sales (1 relationship)
- One customer can make many purchases (1 relationship)



```
create database pharmacydb;
use pharmacydb;
select * from medicine;
select * from customer;
```

	id	address	email	name	phone
*	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                   Message              Duration / Fetch
1  09:59:21  use pharmacydb                           0 row(s) affected    0.000 sec
2  09:59:48  select * from medicine LIMIT 0, 1000     0 row(s) returned    0.000 sec / 0.000 sec
3  10:00:22  select * from customer LIMIT 0, 1000     0 row(s) returned    0.016 sec / 0.000 sec
```

Fig 3: Database Creation in MySQL

Login Module (UI Design):

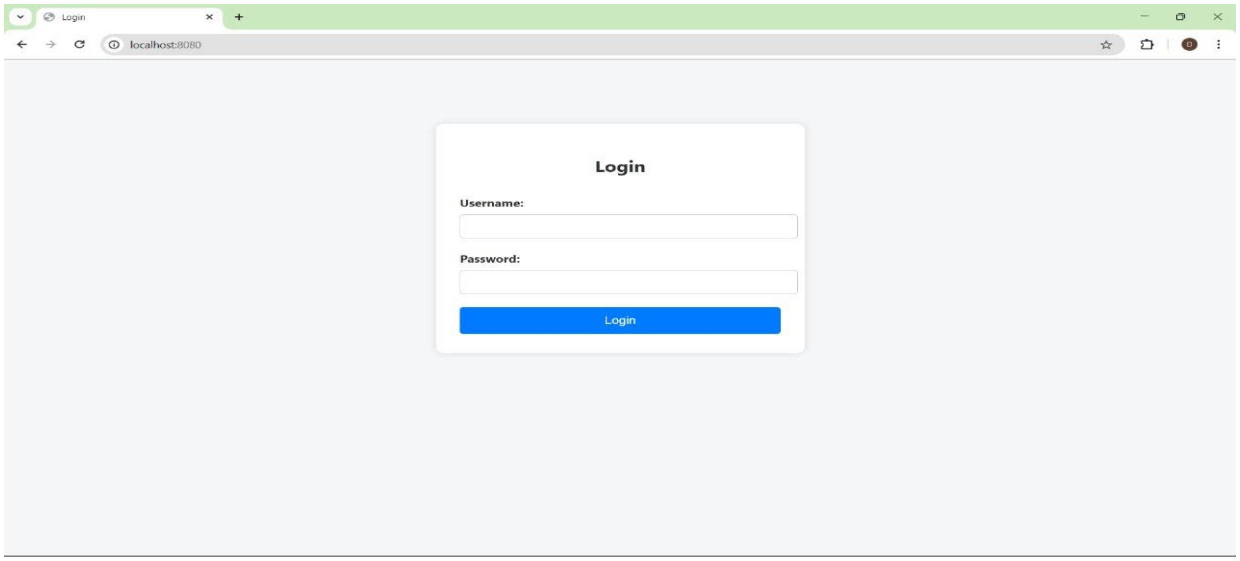
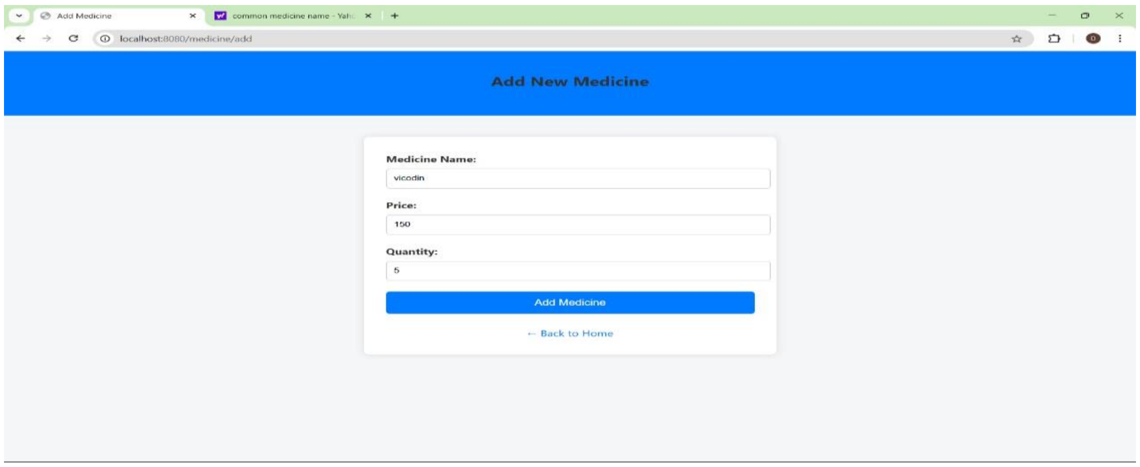


Fig 4: Login Module

The login interface is the entry point to the Pharmacy Management System. Designed using a simple and user-friendly layout, it provides secure access for authorized users such as administrators, pharmacists, and cashiers.

Add New Medicine Module:

The **Add New Medicine** interface is a key component of the Pharmacy Management System. It allows authorized users to add new medicines to the pharmacy database through a simple and clean user interface. This module ensures that the inventory is regularly updated with the latest stock entries.



Medicine Name: vicodin

Price: 150

Quantity: 5

Fig 5: Insert a New Medicine Module

Functionalities and Features:

- **Input Fields:**
  - **Medicine Name:** The name of the medicine to be added (e.g., Vicodin).
  - **Price:** The selling price of a single unit.
  - **Quantity:** The number of units being added to the stock.
- **Database Integration:**
  - Upon clicking the **"Add Medicine"** button, the entered details are stored directly into the MySQL database under the medicines table.
  - Validations are applied to prevent blank submissions or invalid data.
- **Navigation:**
  - A **"Back to Home"** link is provided to improve user navigation and enhance user experience.

Update Medicine Stock Module:

The **Update Medicine Stock** interface allows pharmacy staff to efficiently update the quantity of existing medicines in the inventory. This is crucial for maintaining up-to-date stock levels and ensuring the availability of medicines at all times

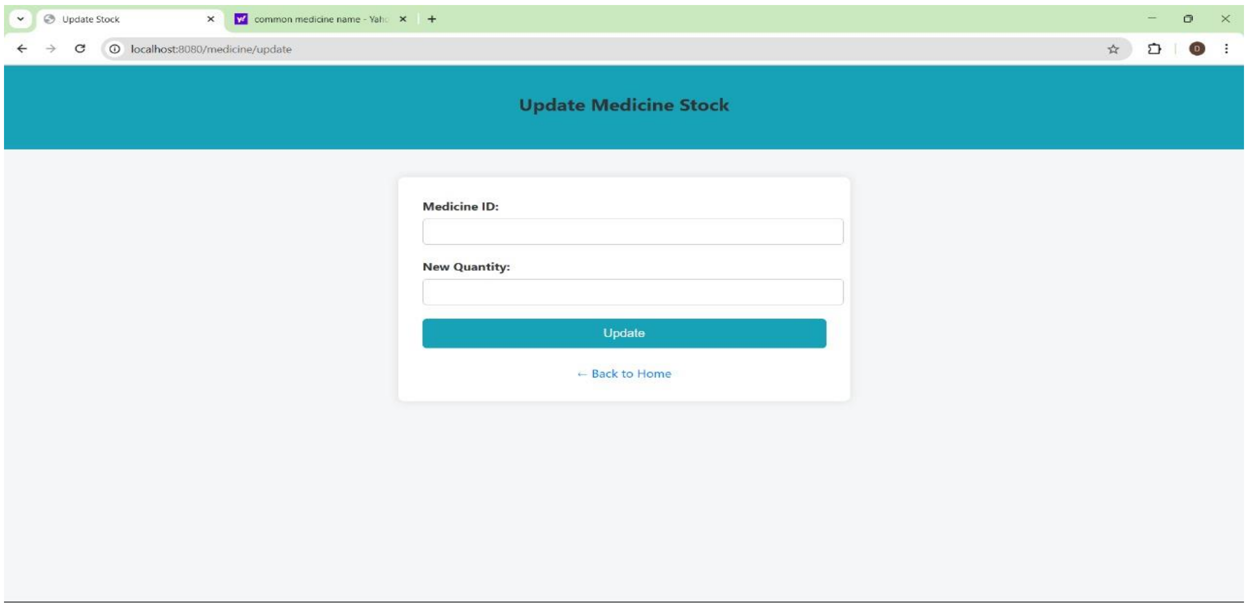


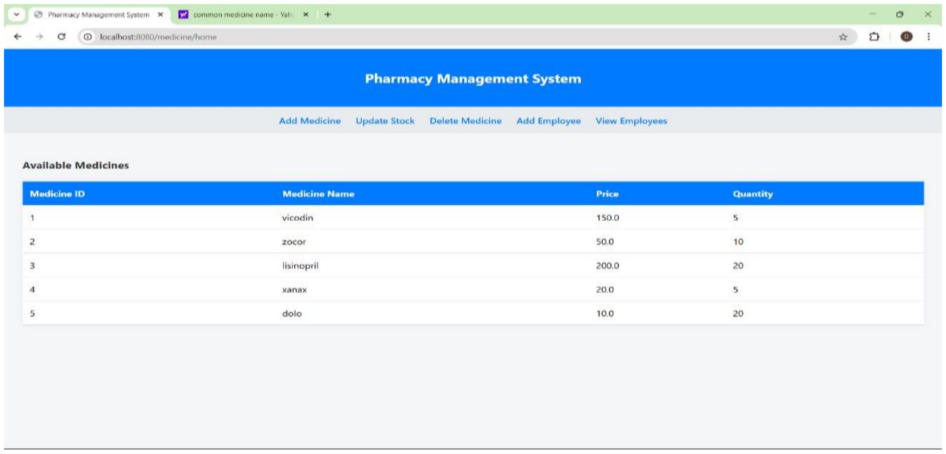
Fig 6: Update Medicine Stock

Functionalities and Features:

- **Input Fields:**
    - **Medicine ID:** A unique identifier of the medicine to be updated.
    - **New Quantity:** The updated quantity to be added or set for the medicine.
- **Database Operation:**
    - When the **"Update"** button is clicked, the system executes an SQL UPDATE command to modify the quantity of the specified medicine in the MySQL database.
    - The application ensures the medicine exists before updating, reducing errors.
- **User Navigation:**
    - A **"Back to Home"** link is provided for quick navigation to the main dashboard.

Available Medicines Module:

The Available Medicines **page is the central dashboard of the Pharmacy Management System that provides a clear, real-time view of the current medicine inventory.**


Pharmacy Management System
common medicine name - Yah
localhost:8080/medicine/home
Pharmacy Management System
Add Medicine
Update Stock
Delete Medicine
Add Employee
View Employees

Available Medicines

Medicine ID	Medicine Name	Price	Quantity
1	vicodin	150.0	5
2	zocor	50.0	10
3	lisinopril	200.0	20
4	xanax	20.0	5
5	dolo	10.0	20

Fig 7: Available Medicine Module

Functionalities and Features:

- **Data Display Table:**
  - Shows a tabular list of all medicines available in the database.
  - Columns include:
    - **Medicine ID** – Unique identifier for each medicine.
    - **Medicine Name** – Generic or brand name of the medicine.
    - **Price** – Cost per unit of the medicine.
    - **Quantity** – Number of units currently available in stock.
- **Navigation Bar:**
  - Includes quick access links to:
    - Add Medicine
    - Update Stock
    - Delete Medicine
    - Add Employee
    - View Employees

Backend Operation:

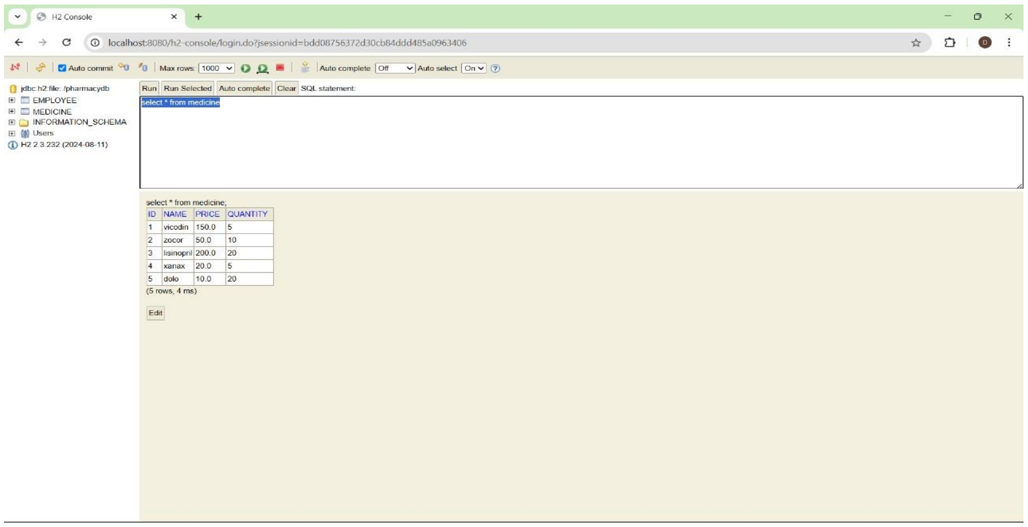
- This table is populated by fetching data using an SQL SELECT query from the medicine table in the MySQL database.
- Data is dynamically rendered using the backend server, ensuring updated stock levels are always displayed.

H2 Console Integration:

The **H2 Database Console** is a web-based interface that allows developers and administrators to directly interact with the H2 database used in the Pharmacy Management System.

Key Features:

- **Web Interface:** Accessible via localhost:8080/h2-console, allowing convenient browser-based database access during
development.



localhost:8080/h2-console/login.do?jsessionid=bdd08756372d30cb84ddd485a0963406

jdbc:h2:file:./pharmacydb
EMPLOYEE
MEDICINE
INFORMATION_SCHEMA
Users
H2 2.3.232 (2024-08-11)

```
select * from medicine
```

select * from medicine;

ID	NAME	PRICE	QUANTITY
1	vicodin	150.0	5
2	zocor	50.0	10
3	lisinopril	200.0	20
4	xanax	20.0	5
5	dolo	10.0	20

(5 rows, 4 ms)

Fig 8**:** H2 Console Integration

- It retrieves all the records from the MEDICINE table and displays columns such as ID, NAME, PRICE, and QUANTITY.

Purpose in Project:

- **Database Verification:** Enables real-time checking of data inserted, updated, or deleted through the application UI.
- **Debugging Support:** Useful during development to verify backend operations like INSERT, UPDATE, DELETE, and SELECT.
- **Testing Queries:** Helps test and validate SQL queries before implementing them in the codebase.

Tables Shown:

- **MEDICINE Table:** Stores medicine data like name, price, and available quantity.
- **EMPLOYEE Table:** Likely holds employee details (used in the employee management module).
- **INFORMATION_SCHEMA:** System tables provided by H2 for metadata access.

The H2 console is a powerful tool that enhances transparency, debugging, and control over the database operations in the Pharmacy Management System.

SOURCE CODE

Application.properties:

```
spring.application.name=pharmacy
spring.datasource.url=jdbc:mysql://localhost:3306/pharmacydb
spring.datasource.username=root
spring.datasource.password=1503
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

CustomerService.Java:

```
package com.example.pharmacy.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.pharmacy.model.Customer;
import com.example.pharmacy.repository.CustomerRepository;

@Service
public class CustomerService {
    @Autowired
    private CustomerRepository customerRepository;

    public List<Customer> getAllCustomers() {
            return customerRepository.findAll();
   }
    public Customer createCustomer(Customer customer) {
            return customerRepository.save(customer);
   }
    public Customer updateCustomer(Long id, Customer customer) {
```

```
             Customer existing = customerRepository.findById(id).orElseThrow();
             existing.setName(customer.getName());
             existing.setEmail(customer.getEmail());
            existing.setPhone(customer.getPhone());
            existing.setAddress(customer.getAddress());
            return customerRepository.save(existing);
    }
    public void deleteCustomer(Long id) {
            customerRepository.deleteById(id);
   }
}
```

MedecineService.java:

```
package com.example.pharmacy.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.pharmacy.model.Medicine;
import com.example.pharmacy.repository.MedicineRepository;

@Service
public class MedicineService {
    @Autowired
    private MedicineRepository medicineRepository;

    public List<Medicine> getAllMedicines() {
              return medicineRepository.findAll();
   }
    public Medicine createMedicine(Medicine medicine) {
              return medicineRepository.save(medicine);
    }
     public Medicine updateMedicine(Long id, Medicine medicine) {

             Medicine existing = medicineRepository.findById(id).orElseThrow();
             existing.setName(medicine.getName());
             existing.setManufacturer(medicine.getManufacturer());
             existing.setPrice(medicine.getPrice());
             existing.setQuantity(medicine.getQuantity());
             existing.setExpiryDate(medicine.getExpiryDate());
             return medicineRepository.save(existing);
     }
     public void deleteMedicine(Long id) {
            medicineRepository.deleteById(id);
     }
  }
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

I. VISION

To produce globally competent, innovative and socially responsible computing professionals

II. MISSION

- To provide world-class teaching-learning and research facilities
- To stimulate students' logical thinking, creativity, and communication skills
- To cultivate awareness about emerging trends through self-initiative
- To instill a sense of societal and ethical responsibilities
- To collaborate with industries and government organizations

III. PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Achieve their professional career in industry/academia by applying the acquired knowledge of computer science and engineering.

PEO 2: Engage in life-long learning and enhance their capabilities by embracing cutting edge technical advancements.

PEO 3: Excel in collaboration with interdisciplinary teams and diverse stake holders for persevering successful start-ups.

IV.PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Build domain specific expertise by showcasing deliverables in the field of Application development, Business Intelligence, Computational Intelligence and Cyber Security

PSO2: Build knowledge base for students to solve complex technical problems through participation in global contests and hackathons.

PO Statement:

PO 1: Apply knowledge of mathematics, natural science, computing and engineering fundamentals, with specializations in computational intelligence, web development, business intelligence, and cyber security to develop solutions to complex engineering problems

PO 2: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences with holistic considerations for sustainable development

PO 3: Design creative solutions for complex engineering problems and design systems, components or processes to meet identified needs with appropriate consideration for public health and safety, whole-life cost, net zero carbon as well as resource, cultural, societal, and environmental considerations as required

PO 4: Conduct investigations of complex engineering problems using research methods including research-based knowledge, design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions

PO 5: Create, select and apply, and recognize limitations of appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to complex engineering problems

PO 6: When solving complex engineering problems, analyze and evaluate sustainable development impacts to: society, the economy, sustainability, health and safety, legal frameworks, and the environment

PO 7: Apply ethical principles and commit to professional ethics and norms of engineering practice and adhere to relevant national and international laws. Demonstrate an understanding of the need for diversity and inclusion

PO 8: Function effectively as an individual, and as a member or leader in diverse and inclusive teams and in multi-disciplinary, face-to-face, remote and distributed settings

PO 9: Communicate effectively and inclusively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, taking into account cultural, language, and learning differences

PO10: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one‘s own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments

PO11: Recognize the need for, and have the preparation and ability for i) independent and lifelong learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change

Possible PO’s Mapped with Micro Project:

PHARMACY MANAGEMENT SYSTEM						
PO1	**PO2**	**PO3**	**PO5**	**PO8**	**PO9**	**PO11**