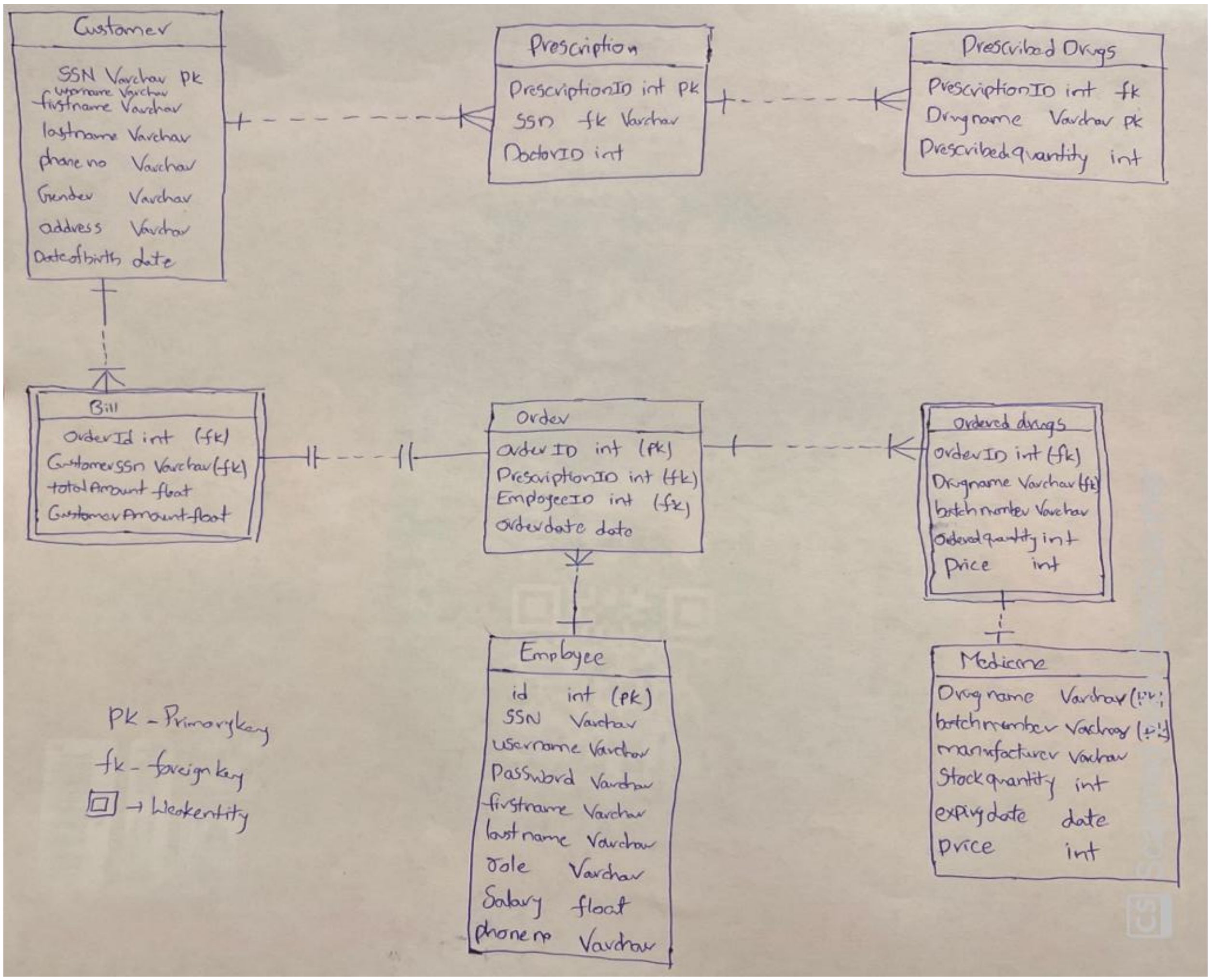
Web-based Database- part II

- Raghu Medarametla A20386265

ENTITY- RELATIONSHIP DIAGRAM:



The Entity Relationship Diagram (ERD) contains eight tables to create a pharmacy management system.

Customer Table: when a customer opens or arrives at the pharmacy. We identify them based on their SSN. If they are new Customers, then details regarding them are collected.

Employee Table: The employee is the person working in the pharmacy, either a cashier or pharmacist. Like the customer, collect their details if a new Employee has joined.

Medicine Table: This is similar to an inventory where we keep track of all the medicines available in our pharmacy.

Prescription and Prescribed drugs Tables: Each prescription has a doctorID to review if it is valid. Each prescription contains several prescribed drugs with the drug name and prescribed quantity.

Order and Ordered Drugs Tables: Order is created based on the prescription. Ordered drugs contain the number of drugs the customer buys based on their prescription.

Bill: we generate a bill for a particular employee based on the ordered drugs.

According to the code implementations, The ERD has the following relations.

Relations:

1. Each customer has multiple prescriptions. Thus, the relation between them is one-to-many relation.
2. Each prescription has multiple prescribed drugs. So, the relation between prescription and prescribed drugs table is one to many.
3. A single order can have multiple drugs. So, the relation between order and ordered drugs is one to many.
4. Each order generates a bill. So, the relation between bill and order is one to one.
5. Each customer can make multiple purchases. Hence the relationship between them is one to many.

Strong Entities: Our ER diagram has strong entities that don’t depend on any other table. For Instance: Employee, Customer, Medicine table has no dependencies on other tables.

Weak Entity: We also have weak entities in the ER Diagram, ordered drugs, which depend on the order table because it doesn’t have any primary key associated with it.

Boyce-Codd Normal Form

To show that our tables are in BCNF. It should follow: if and only every determinant is a BCNF candidate key.

We analyze each table separately by writing their dependencies.

Customer table: The customer table all the other attributes depend on SSN. So, there is no dependency here.

SSN → First Name, Last Name, Phone, Gender, Address, Date of Birth.

Employee table: The employee has a unique ID to this name. so, Id can be a candidate key that determines all other attributes

Employee ID →SSN, First Name, Last Name, Role, Salary, Phone Number, Date of Birth.

Medicine table: It has a unique drug name, batch number, and no functional dependencies on other attributes.

Drug Name, Batch Number → Manufacturer, Stock Quantity, Expiry Date, Price.

Prescription table: It has foreign key constraints with Customer SSN, but there doesn't exist any partial and transitive dependency.

Prescription ID → SSN, Doctor ID, Prescribed Date.

Prescribed Drugs: It has a foreign key constraint from the prescription table. Both prescription id and drug name determine the prescribed quantity of the drug. So, other attributes cannot determine any prime attributes.

Prescription ID, Drug Name → Prescribed Quantity.

Orders: This table has a unique order id and a foreign constraint employee id which explains which employee is taking care of the respective order. So, orderId determines every other attribute.

Order ID → Prescription ID, Employee ID, Order Date.

Ordered table: This table entirely depends on the order table and medicine table. This table doesn't contain any partial dependencies as all the three that determine ordered quantity, price

Order ID, Drug Name, Batch Number → Ordered Quantity, Price.

Bill table: This table contains a foreign key with customer ssn because each bill should be assigned to a customer and orderId because each reported order must get a bill.

Order ID, CustomerSSN →Total Amount, Customer Payment, Insurance Payments.

High-level Implementation, Tools used:

Database: MySQL database is used. It is used to create and insert tables. Create stored procedures, triggers, and views based on the table's requirements.

Back-End: I used high-level java language, which is used to develop REST APIs that will receive and process the REST queries from the front end of the application. Specifically, I used the mybatis framework in java. This framework makes it easier as it automates the mapping between SQL databases and java objects. It acts as a REST server. The tools used for the back end are visual studio code or spring suite tool.

Front-End: I used an angular framework as the application's front end. This acts as a REST client where a set of REST services requests for the queries, and the back end of the application process produces the output on the terminal. This, in turn, uses a type script, which helps connect to the server and client easily. HTML is used to render the view and user interface. It forms web applications like buttons, forms I used in my project, and other functionalities. CSS is used to make the application look colorful.

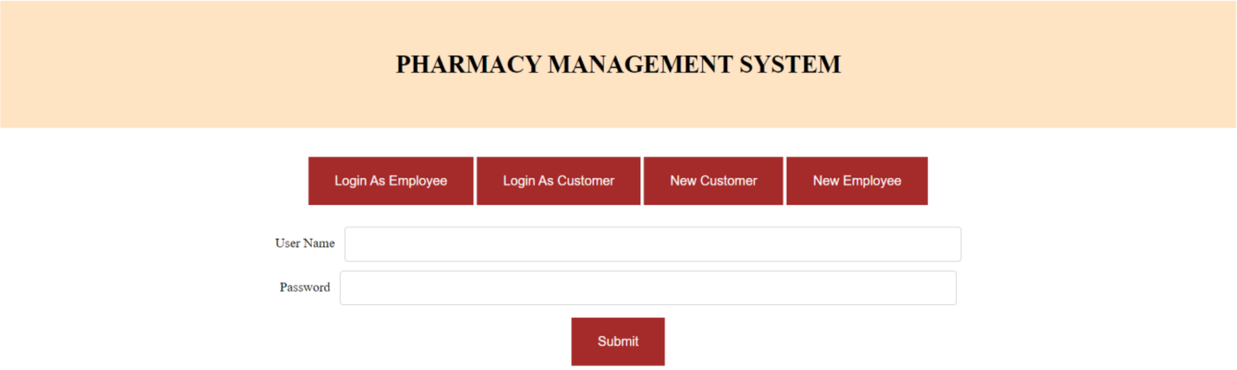
Implementation: The Java program contains the implementation of tables as classes, with each type having its controller, service, and mapper, wherein the mapper, we use the Mybatis framework to write SQL queries to retrieve the information or data. The angular framework uses the type script, which helps to rewrite the classes and code management easier. MySQL implemented view, trigger, and stored procedures.

View: In the code implementation, orders are created, and the employee id in the order table is not assigned in some cases as the order is not yet finished. Suppose some other employee logged on to the system. The employee can check unordered items in the drop-down and assign them.

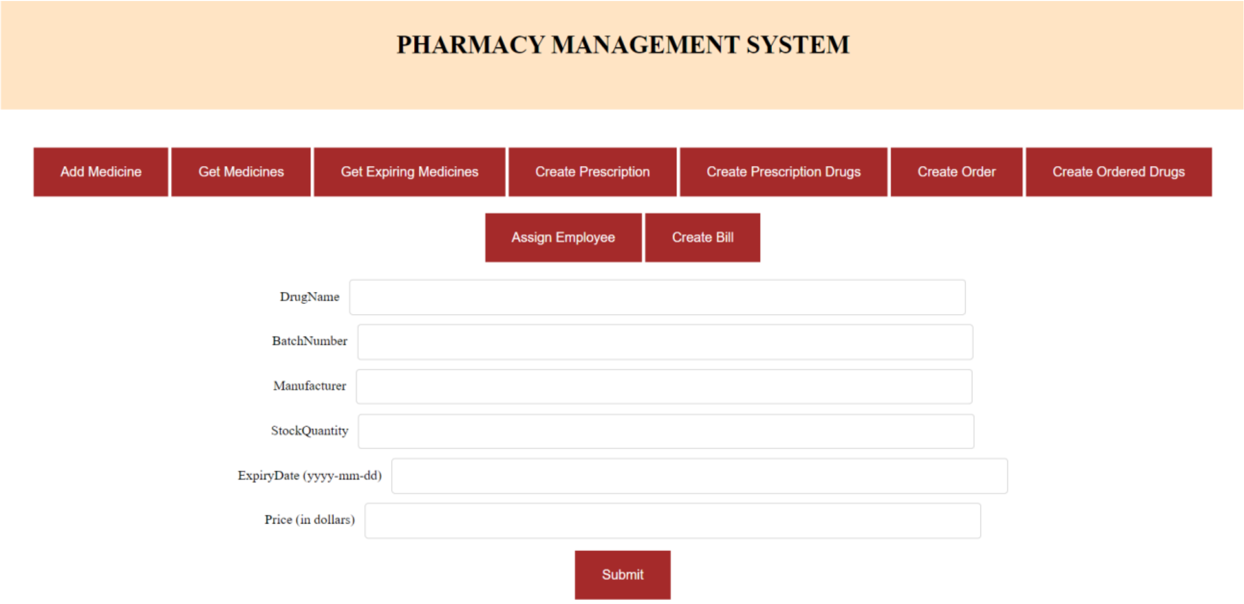
Trigger: In MySQL, we implemented a trigger whenever data is inserted into the order table. The order is created with an automatic date of the day that the order is generated.

Stored Procedure: In MySQL, we implemented generating a bill from the ordered drugs that were created from SQL code only. I also worked on a stored procedure to get the medicines that expire in less than 30 days from the current date.

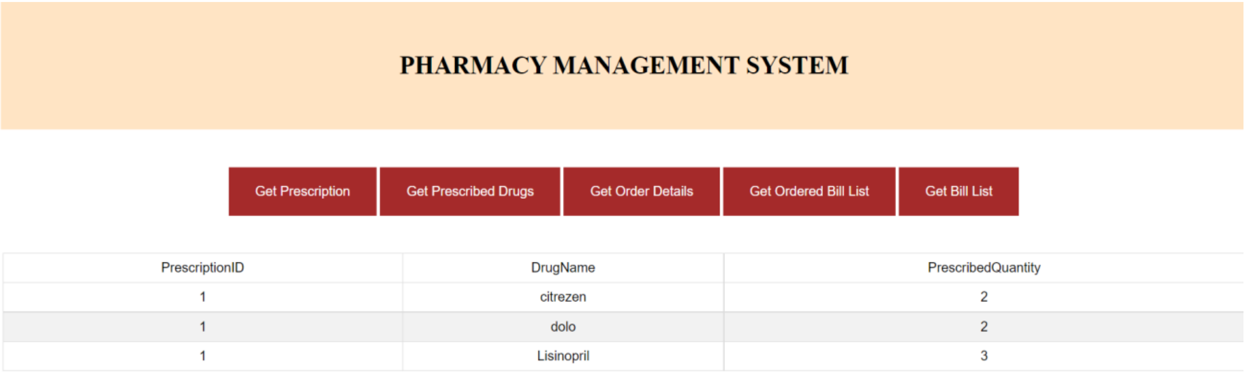
Appendix



This is the opening page of the project, which shows four buttons: login as Employee, login as Customer, if there is a new customer or a new Employee, then there are respective forms to fill which make them register.



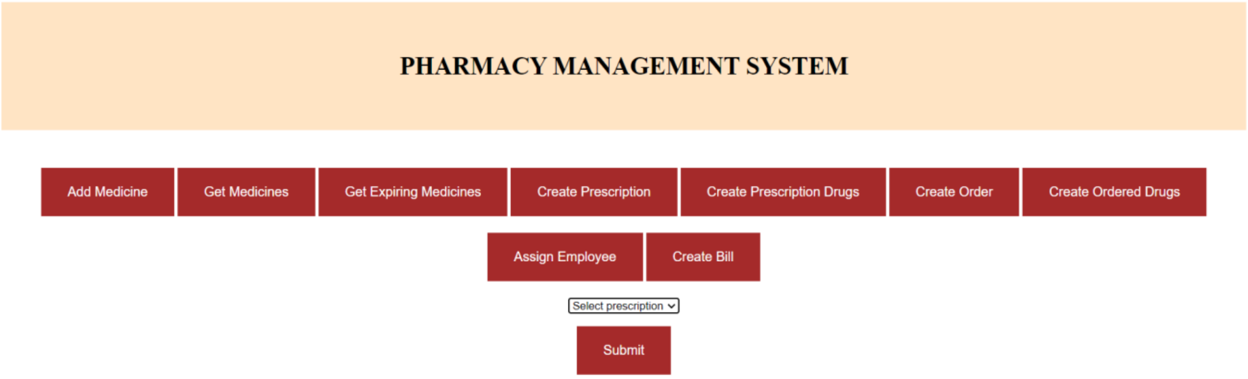
If the user logs in as an employee, the employee can add new medicines and create a prescription for prescribed drugs for the specific customer. Some other functionalities, like making bill forms, are also implemented.



PHARMACY MANAGEMENT SYSTEM

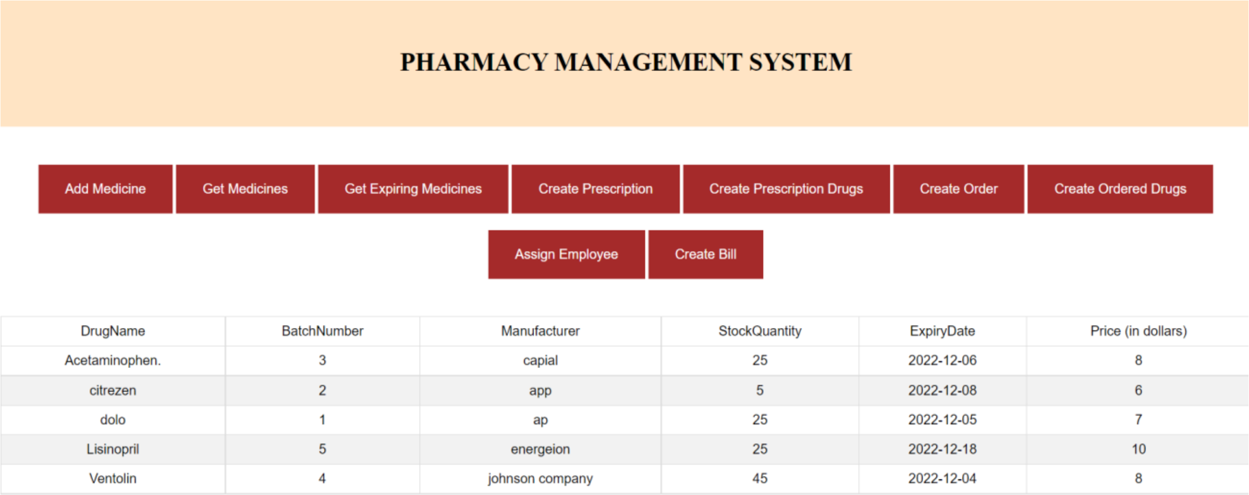
PrescriptionID	DrugName	PrescribedQuantity
1	citrezen	2
1	dolo	2
1	Lisinopril	3

This is the customer page after they login based on their credentials. The customer can be able to see their prescription, which prescribed drugs, what is ordered, the ordered drugs that are assigned for them, and also the total bill form according to their purchase.



PHARMACY MANAGEMENT SYSTEM

This functionality assigns employees where the orders are created, and some orders don’t have an assigned employee id. So, the employee, when logged onto his system, can be able to see the orders where the employee has not been assigned, and they can take up to complete the order.



DrugName	BatchNumber	Manufacturer	StockQuantity	ExpiryDate	Price (in dollars)
Acetaminophen.	3	capial	25	2022-12-06	8
citrezen	2	app	5	2022-12-08	6
dolo	1	ap	25	2022-12-05	7
Lisinopril	5	energeion	25	2022-12-18	10
Ventolin	4	johnson company	45	2022-12-04	8

A stored procedure is used to get the expiring medicines in the medicine/ Inventory; this procedure performs a function that whenever the expiry date of the medicine is less than 30 days from the current date, then the medicine is added to the list of expiring drugs, which helps the employee to keep a track of.