**Introduction:**

In the bustling world of coffee shops, managing orders, inventory, and customer information efficiently is paramount for success. A robust Coffee Shop Database System is the backbone of any modern cafe, facilitating seamless operations, enhancing customer experience, and providing valuable insights for business growth.

The primary objectives of our project are twofold: to conceptualize and materialize a database tailored for coffeshop operations and to craft an intuitive and user-friendly interface that facilitates efficient interaction with the stored data.

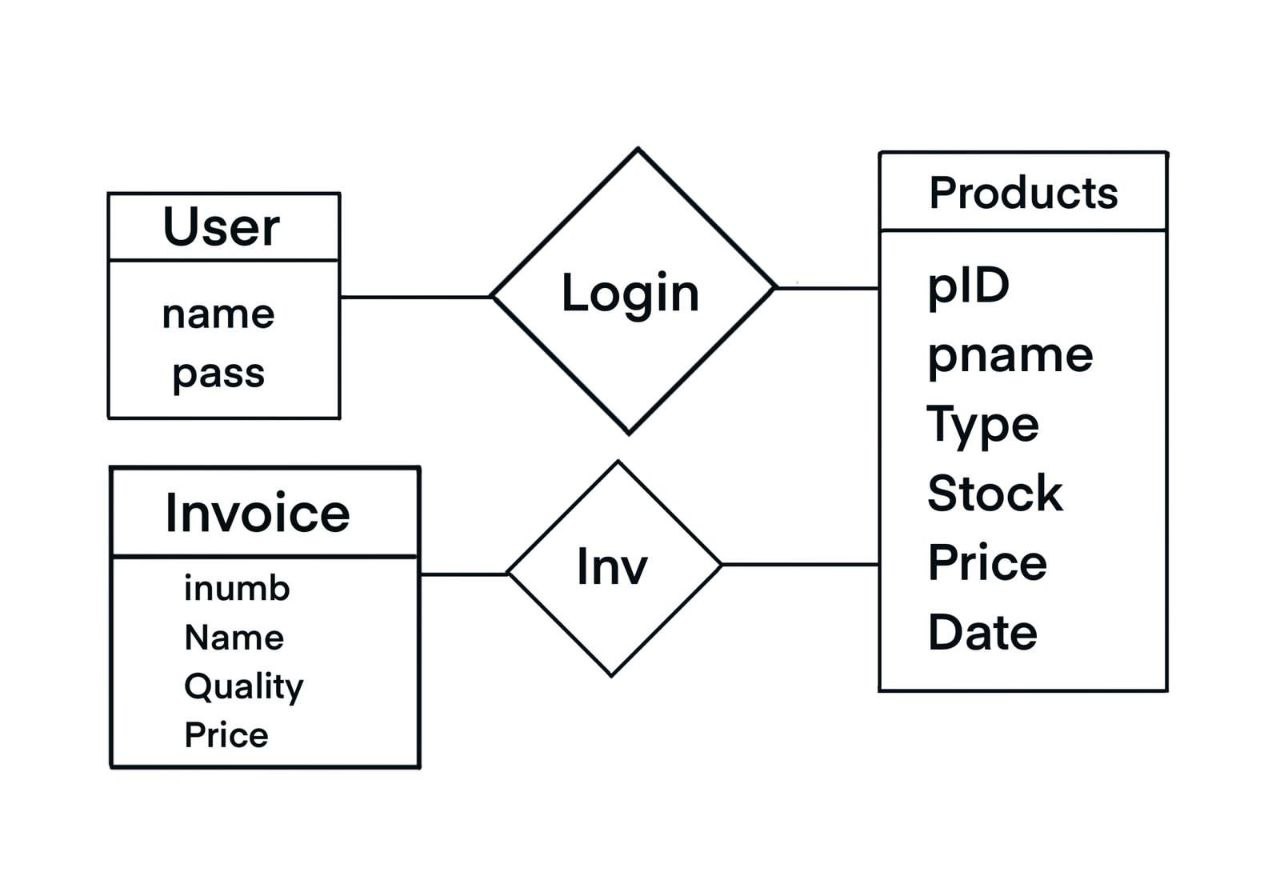
**Chapter 1: Application Architecture**

The architecture of our coffeshop database application is intricately designed to leverage the capabilities of MYSQL as the database engine and Java.fx for the development of the front-end interface. This chapter provides a comprehensive exploration of the architectural choices made to ensure an efficient, scalable, and maintainable solution.

At the core of our application architecture is the utilization of Java.fx as the integrated development environment (IDE) for creating a robust front end. The choice of Java.fx aligns with the seamless integration it offers with SQL Server, enabling smooth communication between the application's frontend and backend components.

**Chapter 2: Database Modeling**

The ER Diagram is as follows:



**The realization of our database model is achieved through the creation of Data Definition Language (DDL) scripts, which serve as the blueprint for our MYSQL database. These scripts encapsulate the structure of each table, define relationships, and establish constraints, ensuring data integrity and coherence.**

**Below is a snippet illustrating the creation of the Pharmacist table as an example:**

**CREATE TABLE invoice (**

**invoice\_number INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,**

**product\_name VARCHAR(30) NOT NULL,**

**quantity INT NOT NULL,**

**price INT NOT NULL**

**);**

**Chapter 3: Services and Interface Modeling**

A critical aspect of our Coffe shop database application is the careful design and implementation of services svfs. This chapter delves into the architecture that governs secure access to the system, ensuring that stakeholders interact with the data based on well-defined roles and permissions.Top of Form

Granular control over database operations is achieved through the assignment of privileges and permissions. Admins, for instance, may have the authority to dispense Product and update its price,.

Services:

The functionality of our CoffeShpo database application is encapsulated within well-defined services. These services represent the various operations that stakeholders can perform within the system. The key services include:

1. Login Service:

2- Add/remove/update product Service:

3- Payment Service

**Chapter 4: Implementation and Testing**

With the foundational design in place, this chapter delves into the practical implementation of our CoffeShop database. The creation of the database involves the execution of Data Definition Language (DDL) scripts, translating our well-crafted design into a tangible and structured database on the MYSQL database.

The DDL scripts, generated from the ERD to PDM transformation, are executed to create tables, define relationships, and enforce constraints. This step is crucial for establishing the structure that underpins our pharmacy management system.

Indexing is strategically applied to enhance query performance. By judiciously choosing which columns to index, we optimize data retrieval speed, balancing the benefits of indexing against the costs of maintenance.

Therefore we were able to implement indexes, besides the clustered index that is already instantiated in the database based on the primary keys, indexes based on attributes we consider to be essential when querying for data and would be of great use for the database management system in speeding up the querying process. These include indexes on the names of medicines, names of brands, and the names of suppliers.

Triggers are implemented to automate specific actions within the database. For instance, triggers may be employed to update inventory levels upon dispensation.

In addition, procedures are also implemented in this database including a procedure to view the total sales by customer.

inventory, querying customer purchase history, and managing supplier interactions.

The optimization of queries and database structures is an ongoing consideration to maintain optimal performance as the system scales.

Ensuring data consistency across different services and interfaces requires careful coordination and validation during the testing phase.

Reflecting on the implementation and testing phases, several key lessons are learned:

1. The importance of a well-structured and normalized database design.

2. The significance of thorough testing in uncovering potential issues and refining the application.

3. The iterative nature of development, where user feedback informs continuous improvement.

In the concluding chapter, we summarize the project's outcomes, draw conclusions from the development journey, and envision future enhancements for our pharmacy database application.