**A Nail-Salon Web App Point-of-Sales System**

**Database Design**

Thi Da Thuy Pham

**Overview**

I will choose SQL Server for my POS system because I am more familiar with it than other options. SQL Server is widely adopted in enterprise environments and offers powerful features for managing relational data. Furthermore, it provides a comprehensive and integrated environment for data management and analysis, with robust security, scalability, and reliability. I can also take advantage of SQL Server Database Projects to manage the database schema, including tables, stored procedures, and functions, and efficiently deploy these projects to a SQL Server instance.

**Data Specifications**

1. Customers:

The **Customers** table will be the central storage point for all customer-related information in my web application. This includes basic personal details like: customer id , customer\_phone , customer\_name.

Document Structure:

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

customer\_phone VARCHAR(15) UNIQUE,

customer\_name VARCHAR(255),

);

1. Staffs:

The **Staffs** table will hold all the details of the employees or staff members who work at my store. This will allow get staff name into order and will help staff can track their orders.

Document Structure:

CREATE TABLE Staffs (

staff\_id INT PRIMARY KEY,

staff\_name VARCHAR(255) NOT NULL

);

1. Products:

The **Products** table will store all the essential information about the items available in your store or system. This includes basic details like the product name, price, and stock quantity.

Document Structure:

CREATE TABLE Products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2),

stock\_quantity INT

);

1. Orders:

The **Orders** table will store information about each individual order placed by a customer. This includes the customer who placed the order, the total amount, order status.

Document Structure:

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

total\_amount DECIMAL(10, 2),

status VARCHAR(50),

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

1. OrderItems:

The **OrderItems** table will also be linked to the **Orders** table to track the specific products in the order.

Document Structure:

CREATE TABLE OrderItems (

order\_item\_id INT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

price DECIMAL(10, 2), -- Price per item

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

**Purpose, Implementation and interactions**

1. Customers:

* Purpose: The customer management to storing and organizing customer data to facilitate easier ordering and customer tracking.
* Implementation: Stores customer data, which will be used for associating orders with a customer. When a cashier create new order, they can input new customer or old customer by phone number into order.
* Interaction: Cashier can create new customer or update name of customer.

1. Staffs:

* Purpose: The **Staff** table's purpose is to store and organize all staff-related data in the system to facilitate easier management of staff interactions with customers and orders.
* Implementation: The **Staff** table will store basic information about each employee. When a cashier create new order, they can input staff name into order.
* Interaction: If a new employee joins the business, a manager can create a new staff record. This can store name of staffs.

1. Products:

* Purpose: The **Products Table** serves as the core component of the Point of Sale (POS) system for managing the inventory of items or services available for sale.
* Implementation: Create new product into database, it’ll store products detail like name, price, stock quantity.
* Interaction: When a new product is added to the store's inventory, a store administrator or manager can input the product’s details (name, price, stock quantity, etc.) into the system. And some products is items and have quantity it maybe has been update after check-out successful.

1. Orders:

* Purpose: Each order placed by a customer is recorded, making it possible to track and manage orders over time by view Order History.
* Implementation: Collect order after cashier create new order. The order\_id is unique, reference to the **Customers Table**, associating each order with a specific customer.
* Interaction: When cashier places an order, the system will insert a new record into the Orders Table.

1. OrderItems:

* Purpose: It tracks which products are included in a specific order, how many units of each product were purchased, and their prices.
* Implementation: The **OrderItems Table** will contain several key columns to store the details of each product in the order. It reference to the order to which the item belongs, and reference to the product being ordered.
* Interaction: When an order is placed, multiple items might be included in the order. Each item will be recorded in the **OrderItems Table**.

**Entity-Relationship diagram:**

**A screenshot of a computer

AI-generated content may be incorrect.**