CS432 Databases

Outlet Management System

Project Proposal

mayank.dahotre@iitgn.ac.in

1. Introduction

The Outlet Management System is designed to automate and optimize the operations of retail and food service outlets. This system enables real-time tracking of inventory, manages sales transactions, monitors employee performance, processes payments, and maintains a comprehensive historical record of orders and stock levels. The goal is to enhance operational efficiency, improve customer experience, and provide valuable insights through reporting and analytics.

2. Objective

* To automate order processing and billing for seamless transactions.
* To ensure real-time monitoring of inventory levels and stock management.
* To securely process and record payments through multiple payment modes.
* To maintain detailed sales, customer, and inventory history for reporting and analytics.
* To enforce data integrity through strict referential constraints.

3. Software Stack

• *Database*: MySQL.

• *Backend*: PHP, Flask(python) etc.

• *Frontend*: HTML, CSS, JS etc.

4. Functionalities

1. **Order Management:**
   * Generate unique order receipts at the time of purchase.
   * Record Order ID, Customer ID (if applicable), Item Details, Quantity, Total Amount, Order Status, and Timestamp.
   * Enforce business rules to prevent duplicate or incomplete orders.
2. **Payment Processing:**
   * Capture payments using methods such as cash, credit, debit, UPI, or digital wallets.
   * Record Payment ID, associated Order ID, Payment Date, Amount, Payment Method, and Transaction details.
3. **Inventory Management:**
   * Maintain a real-time inventory of products with details like Item Name, Category, Stock Quantity, Supplier Information, and Expiry Date (if applicable).
   * Automatically update stock levels upon order fulfillment.
   * Prevent sales of out-of-stock items.
4. **Employee & Shift Management:**
   * Track employee work hours, roles, and assigned shifts.
   * Link employee activity to transactions for accountability.
   * Prevent unauthorized access to sensitive system features.
5. **Reporting & Analytics:**
   * Generate reports on daily sales, inventory usage, and payment transactions.
   * Provide insights into product performance, peak sales hours, and revenue trends.
   * Monitor stock movement to optimize inventory restocking and reduce waste.

5. Tables

### 5.1 Inventory

Holds information about each product/item in the outlet.

CREATE TABLE Inventory (

Item\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Item\_Name VARCHAR(100) NOT NULL,

Category VARCHAR(50),

Stock\_Quantity INT NOT NULL,

Price DECIMAL(8,2) NOT NULL,

Supplier\_Name VARCHAR(100),

Expiry\_Date DATE

);

### 5.2 Customers

Stores details about customers (if applicable for loyalty programs or online orders).

CREATE TABLE Customers (

Customer\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Customer\_Name VARCHAR(100) NOT NULL,

Contact\_Number VARCHAR(15),

Email VARCHAR(100),

Address TEXT

);

### 5.3 Orders

Manages customer orders and records purchase details.

CREATE TABLE Orders (

Order\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Customer\_ID INT,

Order\_Date DATETIME DEFAULT CURRENT\_TIMESTAMP,

Total\_Amount DECIMAL(10,2) NOT NULL,

Order\_Status ENUM('Pending', 'Completed', 'Cancelled') NOT NULL DEFAULT 'Pending',

FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID) ON DELETE SET NULL

);

### 

### 

### 5.4 Order Details

Stores details of individual items in an order.

CREATE TABLE Order\_Details (

Order\_Detail\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Order\_ID INT NOT NULL,

Item\_ID INT NOT NULL,

Quantity INT NOT NULL,

Subtotal DECIMAL(10,2) NOT NULL,

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID) ON DELETE CASCADE,

FOREIGN KEY (Item\_ID) REFERENCES Inventory(Item\_ID) ON DELETE RESTRICT

);

### 5.5 Payments

Records payment transactions for orders.

CREATE TABLE Payments (

Payment\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Order\_ID INT NOT NULL,

Payment\_Date DATETIME DEFAULT CURRENT\_TIMESTAMP,

Amount DECIMAL(10,2) NOT NULL,

Payment\_Method ENUM('Cash', 'Credit', 'Debit', 'UPI', 'Mobile Wallet') NOT NULL,

Transaction\_ID VARCHAR(50),

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID) ON DELETE CASCADE

);

### 5.6 Sales History

Archives completed order records for historical reporting.

CREATE TABLE Sales\_History (

History\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Order\_ID INT NOT NULL,

Customer\_ID INT,

Order\_Date DATETIME NOT NULL,

Total\_Amount DECIMAL(10,2) NOT NULL,

Payment\_Method ENUM('Cash', 'Credit', 'Debit', 'UPI', 'Mobile Wallet') NOT NULL,

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID) ON DELETE CASCADE,

FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID) ON DELETE SET NULL

);

### 

### 

### 5.7 Employee

Tracks Employee Details

CREATE TABLE Employees(

Employee\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Employee\_Name VARCHAR(100) NOT NULL,

Age INT NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Contact\_Number VARCHAR(15) NOT NULL,

Role ENUM(‘Manager’,’Cashier’,’Stock Keeper’) NOT NULL

);

### 5.8 Shifts

Tracks Employees Shifts information

CREATE TABLE Shistd(

Shift\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Employee\_ID INT NOT NULL,

Shift\_Start DATETIME NOT NULL,

Shist\_End DATETIME NOT NULL,

FOREIGN KEY (Employee\_ID) REFRENCES Employees(Employee\_ID) ON DELETE CASCADE

);

### 5.9 Suppliers

Stores Suppliers Details

CREATE TABLE Suppliers(

Supplier\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Supplier\_Name VARCHAR(100) NOT NULL,

Contact\_Number VARCHAR(15),

Email VARCHAR(100)

);

### 5.10 Discounts

Manages Promotional Discounts

CREATE TABLE Discounts(

Discount\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Discount\_Code VARCHAR(20) UNIQUE NOT NULL,

Discount\_Percentage DECIMAL(5,2) NOT NULL,

Expiry\_Date DATE NOT NULL

);

6. SQL Queries

### 6.1 Retrieve Active Orders with Customer and Total Amount

SELECT O.Order\_ID, C.Customer\_Name, O.Order\_Date, O.Total\_Amount, O.Order\_Status

FROM Orders O

LEFT JOIN Customers C ON O.Customer\_ID = C.Customer\_ID

WHERE O.Order\_Status = 'Pending';

### 6.2 Retrieve Payment Details for a Specific Order

SELECT \*

FROM Payments

WHERE Order\_ID = 101;

### 6.3 Find All Items with Low Stock (Below 10 Units)

SELECT \*

FROM Inventory

WHERE Stock\_Quantity < 10;

### 6.4 Count the Total Number of Orders Processed Today

SELECT COUNT(\*) AS Total\_Orders\_Today

FROM Orders

WHERE DATE(Order\_Date) = CURDATE();

### 6.5 Retrieve Complete Sales History with Detailed Information

SELECT SH.History\_ID, C.Customer\_Name, O.Order\_Date, O.Total\_Amount, SH.Payment\_Method

FROM Sales\_History SH

JOIN Orders O ON SH.Order\_ID = O.Order\_ID

LEFT JOIN Customers C ON SH.Customer\_ID = C.Customer\_ID;

This schema ensures **data integrity**, **efficient tracking of sales and inventory**, and **historical record-keeping** for better insights into outlet performance. Let me know if you need any modifications