Yingtong Lin

StudentID: 991747356

Café database design

About the project

The system design follows object-oriented principles to improve flexibility, scalability and maintainability. Classes like MenuItem, Order, Inventory and Employee encapsulate specific functionality and interact with each other through well-defined relationships. Using enum for classes like manager and staff ensures type safety and simplifies role management.

Use Case Narratives

Scenario 1: Staff place a new order in the system for the customer.

1. The staff selects the "New Order" from the menu.
2. The system displays the menu items.
3. The staff selects items and specifies quantities.
4. The system calculates the total cost of the order.
5. The staff confirms the order and processes payment.
6. The system saves the order details in the Orders and OrderItems tables and displays a confirmation.

Scenario 2: Staff to update items in the menu.

1. The staff selects the "Menu Management" on the main menu.
2. The system displays the current menu.
3. The staff clicks to add a new item or edit/remove an existing item.
    a. Add: The staff enters the item name, category, price.
    b. Edit: The staff modifies the selected item’s details.
    c. Remove: The staff confirms the deletion of the item.
4. The system updates the MenuItems table and displays a confirmation message.

Scenario 3: Staff manage item inventory information through the system.

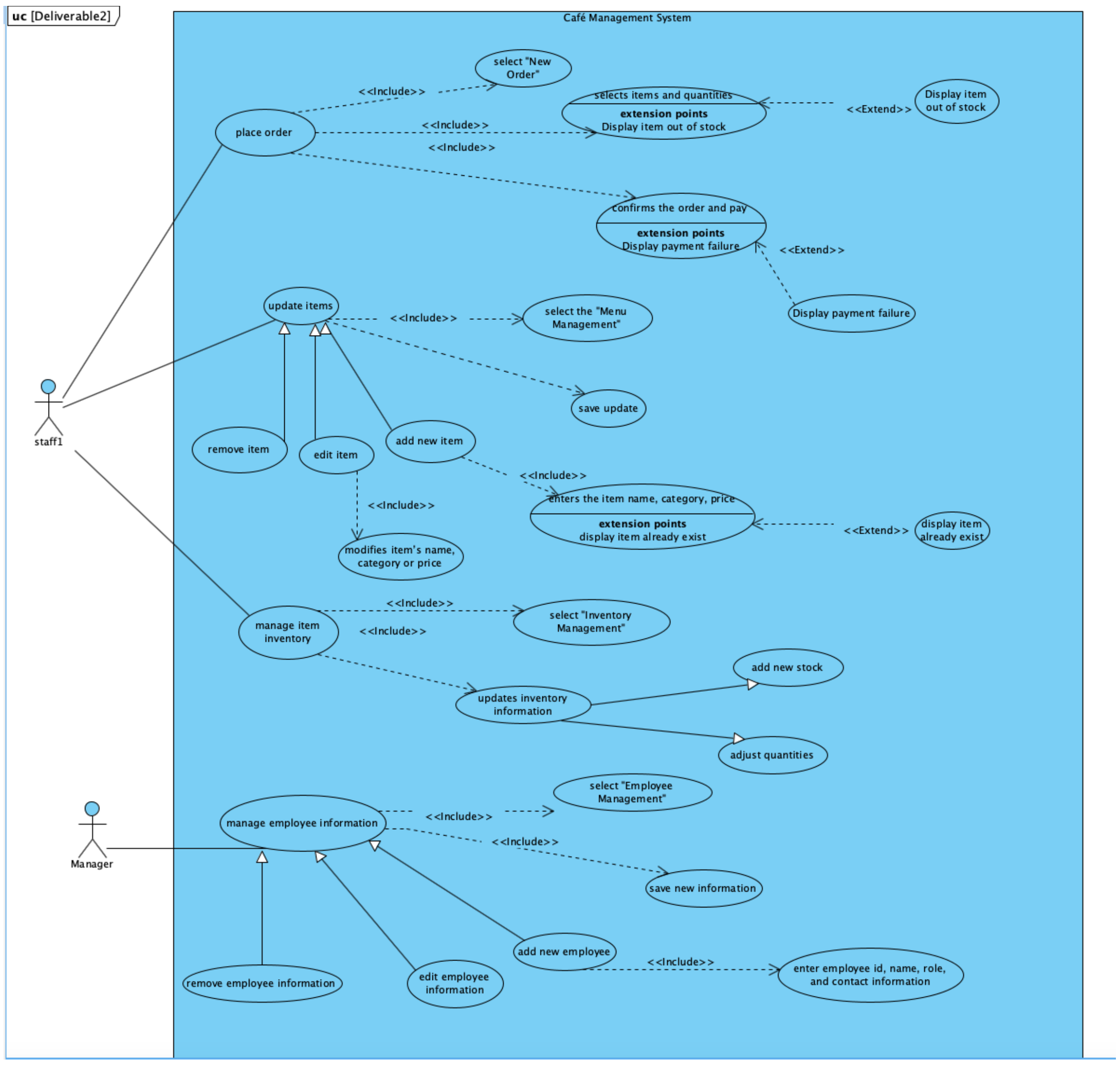
1. Staff selects the "Inventory Management" from the main menu.

2. The system displays current inventory, including item names, quantities, and expiration dates.
3. Staff updates inventory information by adding new stock or adjusting quantities for existing items.
4. The system updates the Inventory table in the database and displays a confirmation message.

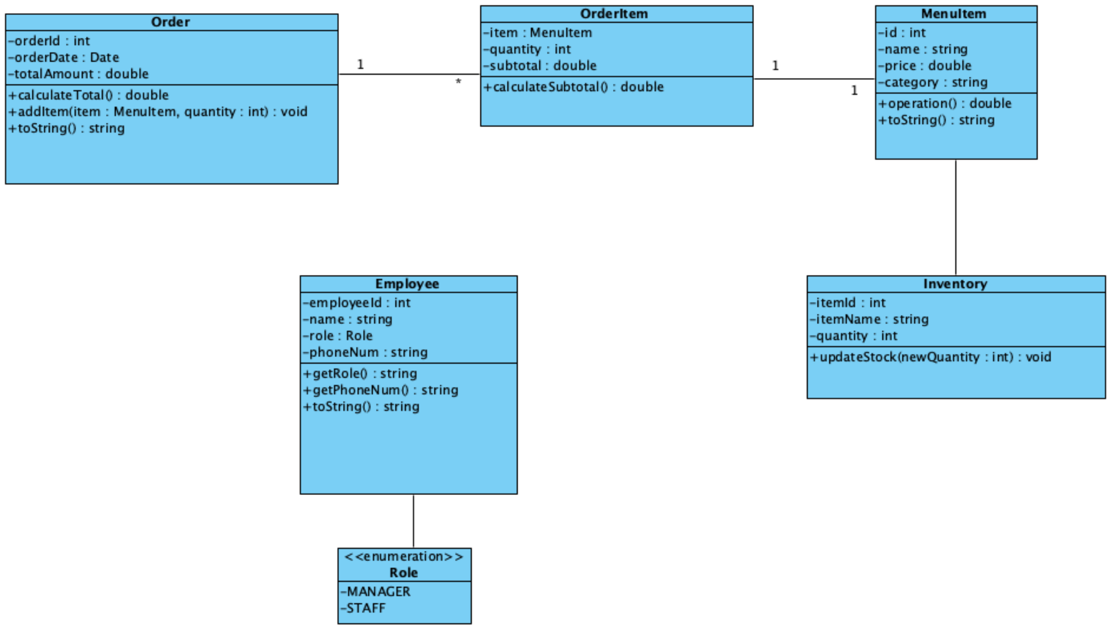
Scenario 4: Manager manages employee information through the system.

1. The manager selects the "Employee Management" from the main menu.
2. The system displays a list of employees.
3. The manager adds a new employee, edit existing employee information, or remove an employee.
    a. Add: The manager enters employee information such as name, role, and contact information.
    b. Edit: The manager modifies the selected employee’s details.
    c. Remove: The manager confirms the deletion of the employee record.
4. The system updates the Employees table and displays a confirmation.

Use Case Diagram


uc [Deliverable2]
Café Management System
select "New Order"
<<Include>>
place order
selects items and quantities
extension points
Display item out of stock
Display item out of stock
<<Extend>>
<<Include>>
<<Include>>
confirms the order and pay
extension points
Display payment failure
<<Extend>>
Display payment failure
update items
<<Include>>
select the "Menu Management"
save update
staff1
remove item
edit item
add new item
<<Include>>
enters the item name, category, price
extension points
display item already exist
<<Extend>>
display item already exist
<<Include>>
modifies item's name, category or price
manage item inventory
<<Include>>
select "Inventory Management"
<<Include>>
add new stock
updates inventory information
adjust quantities
Manager
manage employee information
<<Include>>
select "Employee Management"
<<Include>>
save new information
add new employee
<<Include>>
enter employee id, name, role, and contact information
remove employee information
edit employee information

Class Diagram



Order: a customer order, containing orderId, orderDate, and totalAmount. It has a one-to-many association with OrderItem. In this class, the calculateTotal() method can get the total price of this order. addItem() method can add a new item, which will create an OrderItem instance to this order.

OrderItem: contains quantity and subtotal. It includes methods to calculate the subtotal for each item based on the quantity and menu item price.

MenuItem: with attributes like id, name, category, and price. It is associated with OrderItem, as each order contains one or more menu items.

Inventory: contains items with attributes like itemName, quantity, and threshold. It helps ensure stock management.

Employee: staff members with attributes such as employeeId, name, phoneNumber and role. The role is defined using an enum (e.g., MANAGER, STAFF) to ensure consistency and type safety.

The toString() method in every class can provide a concise description of each class, showing their details information.

Encapsulation ensures that each class in the system hides its internal data and exposes only necessary functionalities through public methods. For example, attributes like price in MenuItem or totalAmount in Order are private and accessed via getter and setter methods.

The system follows the principle of delegation to distribute responsibilities across multiple classes. The Order class delegates the task of calculating subtotals to the OrderItem class through its calculateSubtotal() method.

Also, each class has a single, well-defined purpose. This follows the principle of Cohesion.

A specific roles like Manager or Cashier extend from a base Employee class, they inherit shared attributes like employeeId and name.