UNIVERSITY OF SINDH

NAME:	**MUZAMMIL SHIRAZ**
S/O:	**KAMRAN SHIRAZ**
CLASS:	**AI (3rd)SEMESTER**
ROLL NO:	**2K24/AI/72**

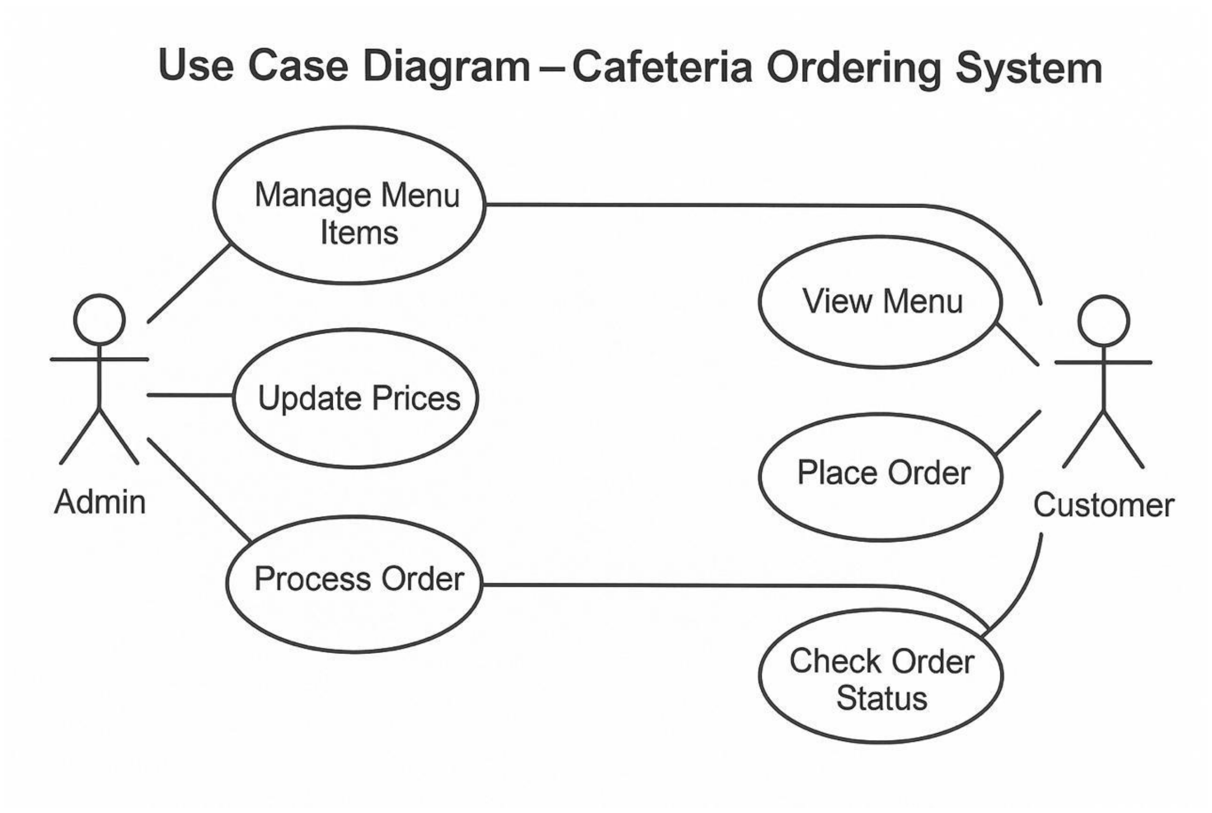
PROJECT TITLE : CAFETERIA ORDERING SYSTEM

SUBJECT: SOFTWARE ENGINEERING

SUBMITTED DATE: 15/MAY/2025

TEACHER: SIR YASIR NAWAZ

Use Case Diagram – Cafeteria Ordering System



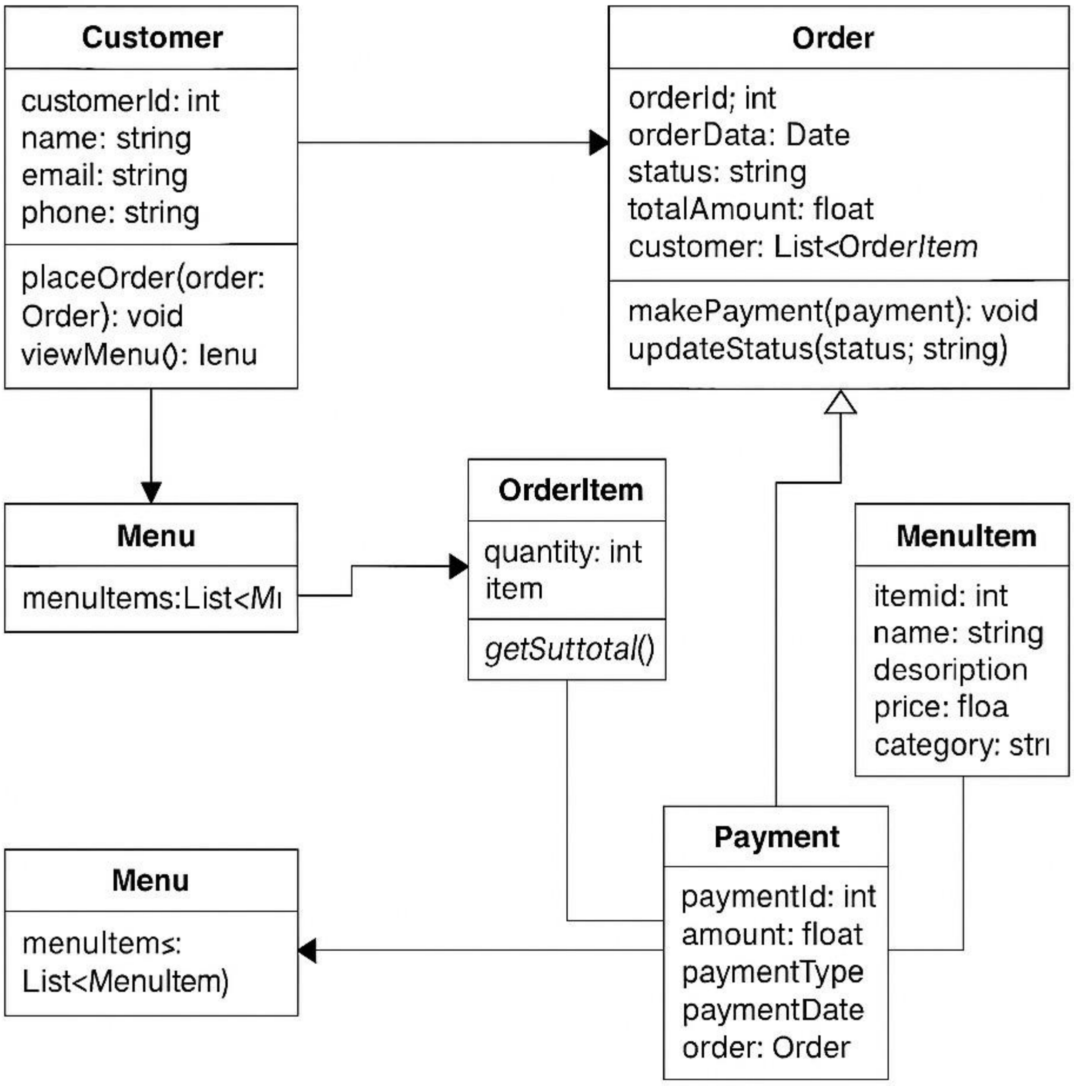
Actors:

- Admin
- Manage Menu Items: Add, update, or remove items from the cafeteria menu.
- Update Prices: Modify the prices of menu items.
- Process Order: Handle incoming orders and mark them as processed.
- Customer
- View Menu: Browse available food and beverage options.
- Place Order: Select items from the menu and place an order.
- Check Order Status: Monitor the status of their placed order (e.g., pending, in progress, ready).

Purpose:

This diagram illustrates the interactions between the Admin and Customer with the system. It helps identify system requirements and user expectations by showcasing the essential use cases.

Class diagram



1. Customer Class

Represents a user placing orders in the system.

- Attributes:
- customerId: int – Unique identifier.
- name: string – Customer’s full name.
- email: string – Email address.
- phone: string – Contact number.
- Methods:
- placeOrder(order: Order): void – Creates an order.

- viewMenu(): Menu – Lets the customer see the menu.

Relationship:

- Connected to Order (1 customer → many orders).
- Accesses Menu to select items.

2. Menu Class

Acts as a container for available items.

- Attributes:
- menuItems: List<MenuItem> – List of all available items.

Relationship:

- Linked to MenuItem (many items per menu).
- Accessed by Customer.

3. MenuItem Class

Represents individual food or drink items on the menu.

- Attributes:
- itemId: int – Unique item ID.
- name: string – Item name.
- description: string – Brief description.
- price: float – Price of the item.
- category: string – E.g., "beverage", "snack", "meal".

Relationship:

- Used in OrderItem to define what was ordered.

4. Order Class

Handles the main order transaction.

- Attributes:
- orderId: int – Unique order number.
- orderDate: Date – When the order was placed.
- status: string – Tracks progress (e.g., "Pending", "In Progress").

- totalAmount: float – Total cost.
- customer: List<OrderItem> – What items are in the order.
- Methods:
- makePayment(payment): void – Completes the payment.
- updateStatus(status: string) – Changes the order status.

<u>Relationship:</u>

- Associated with Customer and contains multiple OrderItem entries.

<u>5. OrderItem Class</u>

Breaks down an order into individual entries.

- Attributes:
- quantity: int – Quantity of the item ordered.
- item – Refers to the specific MenuItem.
- Methods:
- getSubtotal() – Calculates subtotal (quantity × price).

<u>Relationship:</u>

- Bridges Order and MenuItem.

<u>6. Payment Class</u>

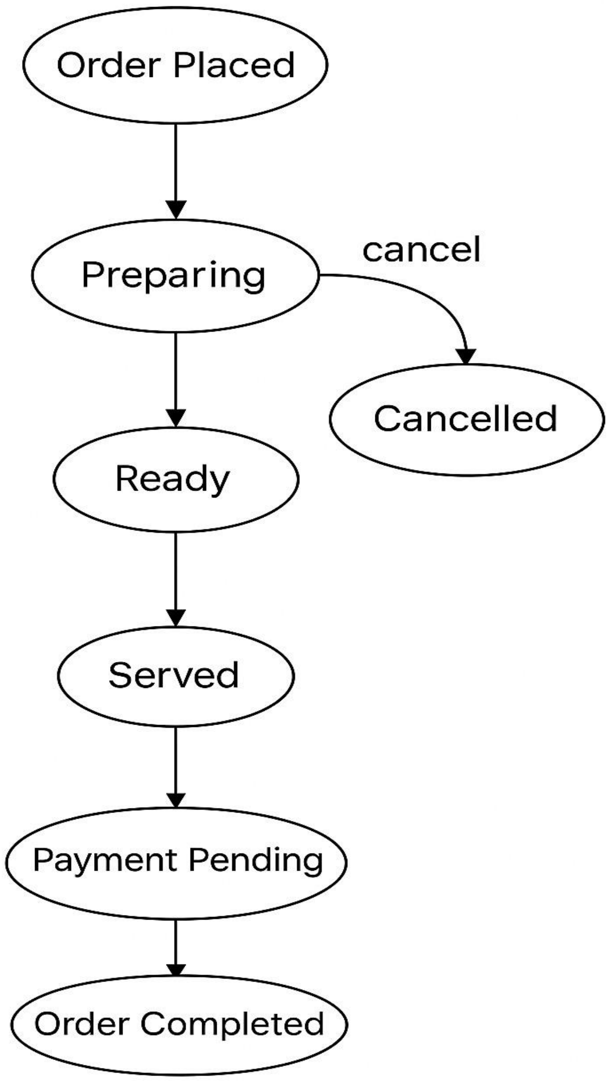
Tracks how an order was paid.

- Attributes:
- paymentId: int – Unique payment ID.
- amount: float – Amount paid.
- paymentType – Type of payment (e.g., cash, card).
- paymentDate – Date of payment.
- order: Order – The order associated with the payment.

<u>Relationship:</u>

- One payment is linked to one order.

State diagram



Order Flow:

1. Order Placed – Customer places an order.
2. Preparing – Kitchen starts preparing the order.
   - Can be Cancelled at this stage.
3. Ready – Order is prepared and waiting to be served.
4. Served – Order is delivered to the customer.
5. Payment Pending – Waiting for payment after serving.
6. Order Completed – Payment received; process finished.