DIGITAL ASSIGNMENT 1 (Theory)

BCSE301L_SOFTWARE-ENGINEERING

WIN 2024-2025

Slot: G2+TG2

Name: Karan Sehgal

Reg No: 22BCE3939

Date: 13.03.2025

Submitted to: Dr. Mehfooza M

Food Ordering System: UML Diagrams

1) Behavioural Model

State Transition for Food Delivery System

Introduction

The state transition table provides a structured representation of how the system progresses through various states based on user interactions and internal processes. It details the transitions triggered by specific events, ensuring a clear understanding of the workflow from user login to order fulfilment and resolution. By mapping out these states, we can identify parallel processes, dependencies, and critical decision points that enhance the efficiency of the food ordering and delivery system.

1) Identified States:

- Idle State
- Register
- Login
- User Authentication
- Customer Dashboard
- Food Selection
- Cart Ready
- Payment Processing
- Payment Retry
- Order Confirmation
- Order Processing
- Notification Sent
- Delivery Agent Assignment
- Ready for Pickup
- Out for Delivery
- Order Delivered

- Feedback Recorded
- Order History Updated
- Complaint Resolution
- Order Closed
- System Logout

2) Events and Transitions:

- User Registration: Transition from Idle State to Register.
- User Login: Transition from Idle State to Login.
- User Authentication: Transition from Login to User Authentication.
- Dashboard Access: Transition from Authentication to Customer Dashboard.
- Food Selection: User selects food and transitions to Food Selection.
- Cart Ready: Items added to the cart transition to Cart Ready.
- Payment Processing: User proceeds to checkout and enters Payment Processing.
- Payment Failure: If payment fails, transition to Payment Retry.
- Payment Success: Successful payment transitions to Order Confirmation.
- Order Confirmation: Restaurant receives the order and transitions to Order Processing.
- Forking into Parallel States:
    - Order Processing
    - Notification Sent
    - Delivery Agent Assignment
- Joining Condition: All parallel states must finish before transitioning to Ready for Pickup.
- Ready for Pickup: Once food is prepared, it moves to Ready for Pickup.
- Out for Delivery: Delivery agent picks up the order.

- Order Delivered: Customer receives the order.
- Post-Delivery:
    - If the customer provides feedback, transition to Feedback Recorded.
    - If no action is taken, transition to Order History Updated.
    - If the customer reports an issue, transition to Complaint Resolution.
- Order Closure: After feedback recording, complaint resolution, or no action, transition to Order Closed.
- User Logout: User logs out, transitioning to System Logout.

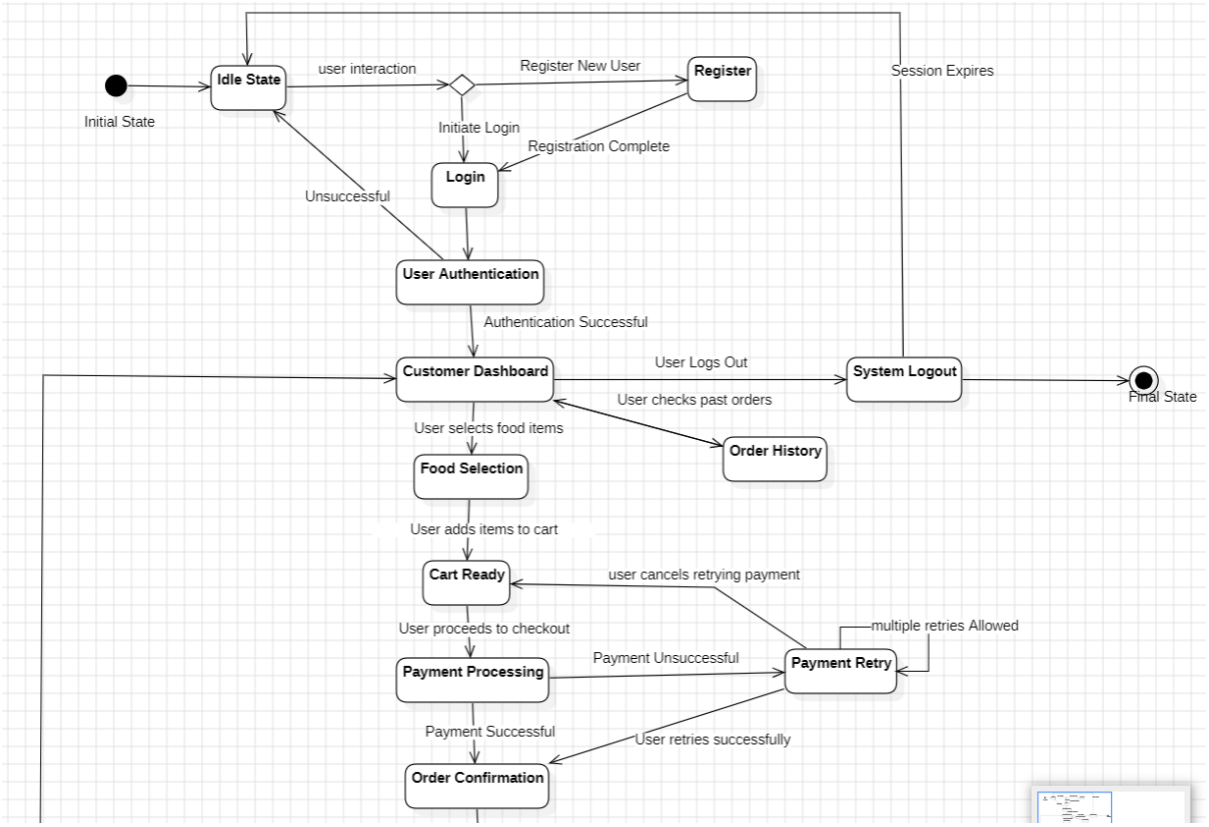
3) Explanation of State Transitions:

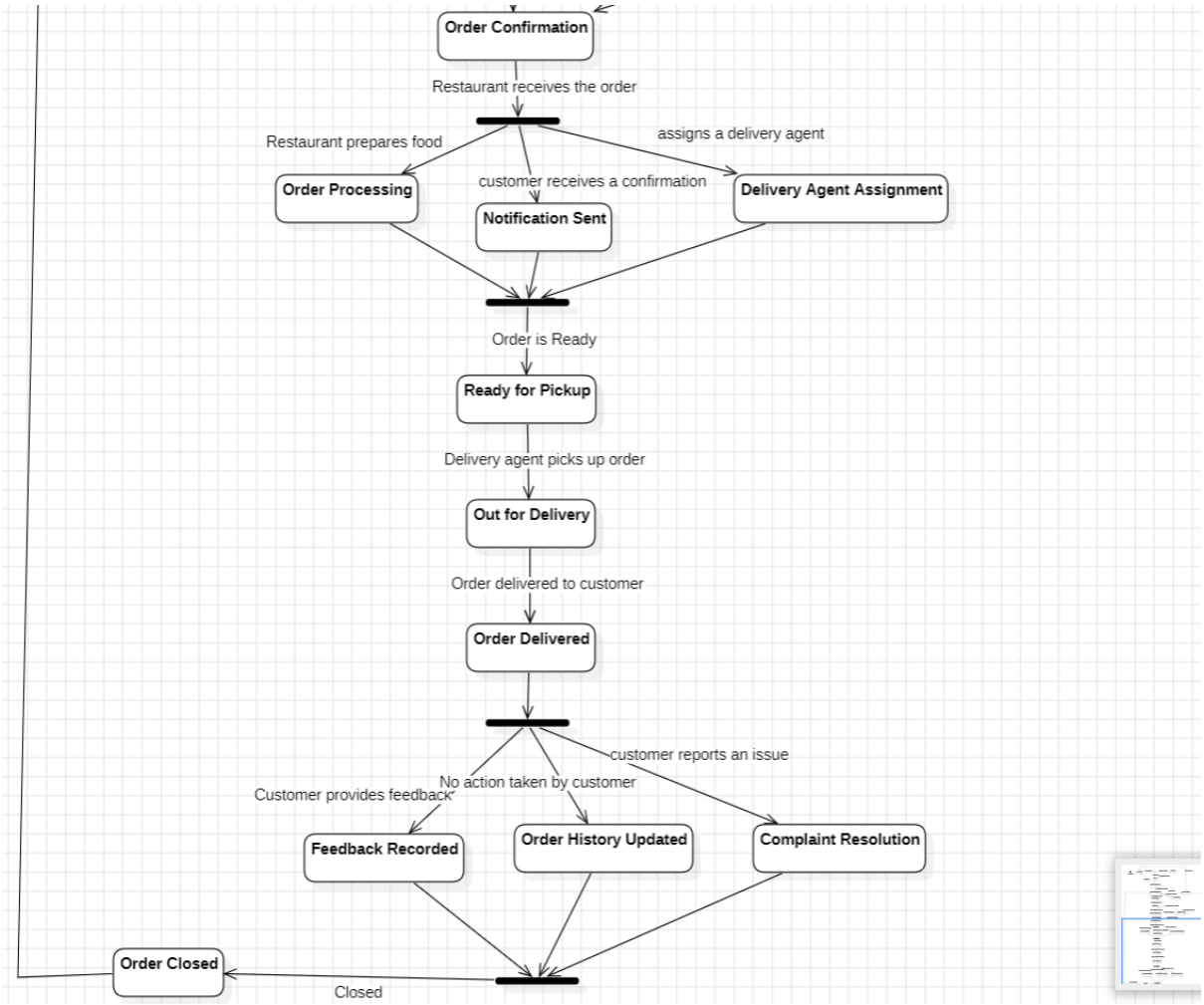
1. User Authentication Flow:
    - The user starts in the Idle State and can either register or log in.
    - Upon successful login, authentication is completed, transitioning to the Customer Dashboard.
2. Ordering Process:
    - The user selects food, adds it to the cart, and proceeds to checkout.
    - Payment processing happens, with an option to retry if it fails.
    - Upon successful payment, the order is confirmed and sent to the restaurant.
3. Forking Parallel Processes:
    - The restaurant starts order preparation.
    - A notification is sent to the customer.
    - A delivery agent is assigned.
    - These processes must all be completed before moving to Ready for Pickup.
4. Delivery Process:
    - Once the order is picked up, it transitions to Out for Delivery.
    - When delivered, it moves to Order Delivered.

5. Post-Delivery Actions:
    - Customers can provide feedback, which updates the order history.
    - If an issue is reported, a complaint resolution process starts.
    - If no action is taken, the order history is simply updated.
    - All these paths lead to Order Closed as the final state.
6. System Logout:
    - The user can log out at any stage, transitioning to System Logout.

This structured state transition ensures efficiency in the order processing, payment validation, and delivery tracking.

State Transition Diagram::


Initial State
Idle State
user interaction
Register New User
Register
Session Expires
Initiate Login
Registration Complete
Login
Unsuccessful
User Authentication
Authentication Successful
Customer Dashboard
User Logs Out
System Logout
Final State
User checks past orders
User selects food items
Order History
Food Selection
User adds items to cart
Cart Ready
user cancels retrying payment
User proceeds to checkout
multiple retries Allowed
Payment Unsuccessful
Payment Processing
Payment Retry
Payment Successful
User retries successfully
Order Confirmation


Order Confirmation
Restaurant receives the order
Restaurant prepares food
assigns a delivery agent
Order Processing
customer receives a confirmation
Delivery Agent Assignment
Notification Sent
Order is Ready
Ready for Pickup
Delivery agent picks up order
Out for Delivery
Order delivered to customer
Order Delivered
customer reports an issue
No action taken by customer
Customer provides feedback
Feedback Recorded
Order History Updated
Complaint Resolution
Order Closed
Closed

2) Scenario Model

Use Case Model for Food Ordering System

The **Use Case Model** for the **Food Ordering System** provides a high-level representation of the system’s functionality by identifying key actors, their interactions, and various processes involved in online food ordering. This model helps in understanding the system's workflow, ensuring clarity in requirements, and defining how different users interact with the platform.

The use case diagram visually represents these interactions, depicting core functionalities such as user registration, order placement, payment processing, food preparation, delivery, and complaint resolution. By establishing relationships between actors and use cases, this model ensures a structured approach to system design and development.

Step 1: Identify Actors

1. **Customer** - The end-user who places orders through the system.
2. **Admin** - Manages the restaurant listings, menu items, and handles complaints.
3. **Restaurant** - Prepares the ordered food.
4. **Delivery Agent** - Picks up and delivers the food to the customer.
5. **Payment System** - Handles transactions for order payments.

Step 2: Identify Use Cases

1. **User Registration & Login** - Allows customers to create an account and log in.
2. **Browse Menu** - Customers can explore restaurant menus.
3. **Add to Cart** - Customers can select items and add them to their cart.
4. **Checkout** - Customers finalize their order and proceed to payment.
5. **Make Payment** - Customers complete the transaction (includes Payment System).
6. **Confirm Order** - The system confirms the order after successful payment.
7. **Send Notification** - Sends updates about order status.

8. **Prepare Order** - The restaurant prepares the food.
9. **Assign Delivery Agent** - Assigns an agent to deliver the order.
10. **Pickup Order** - The delivery agent collects the order from the restaurant.
11. **Deliver Order** - The delivery agent delivers the food to the customer.
12. **Receive Order** - Customers receive the delivered order.
13. **Give Feedback** - Customers provide reviews and feedback.
14. **Resolve Complaint** - Customers raise complaints, and the admin handles them.
15. **View Order History** - Customers can view their past orders.
16. **Manage Restaurants** - The admin manages restaurant details.
17. **Manage Menu Items** - The admin updates restaurant menus.
18. **Handle Complaints** - The admin resolves customer complaints.

Step 3: Define Relationships

Associations:

- Customer interacts with **User Registration & Login**, **Browse Menu**, **Add to Cart**, **Checkout**, **Make Payment**, **Confirm Order**, **Receive Order**, **Give Feedback**, **Resolve Complaint**, and **View Order History**.
- Restaurant interacts with **Prepare Order** and **Assigning Delivery Agent**.
- Delivery Agent interacts with **Pickup Order** and **Deliver Order**.
- Admin interacts with **Manage Restaurants**, **Manage Menu Items**, and **Handle Complaints**.
- Payment System is included in **Make Payment** and **Send Notifications**.

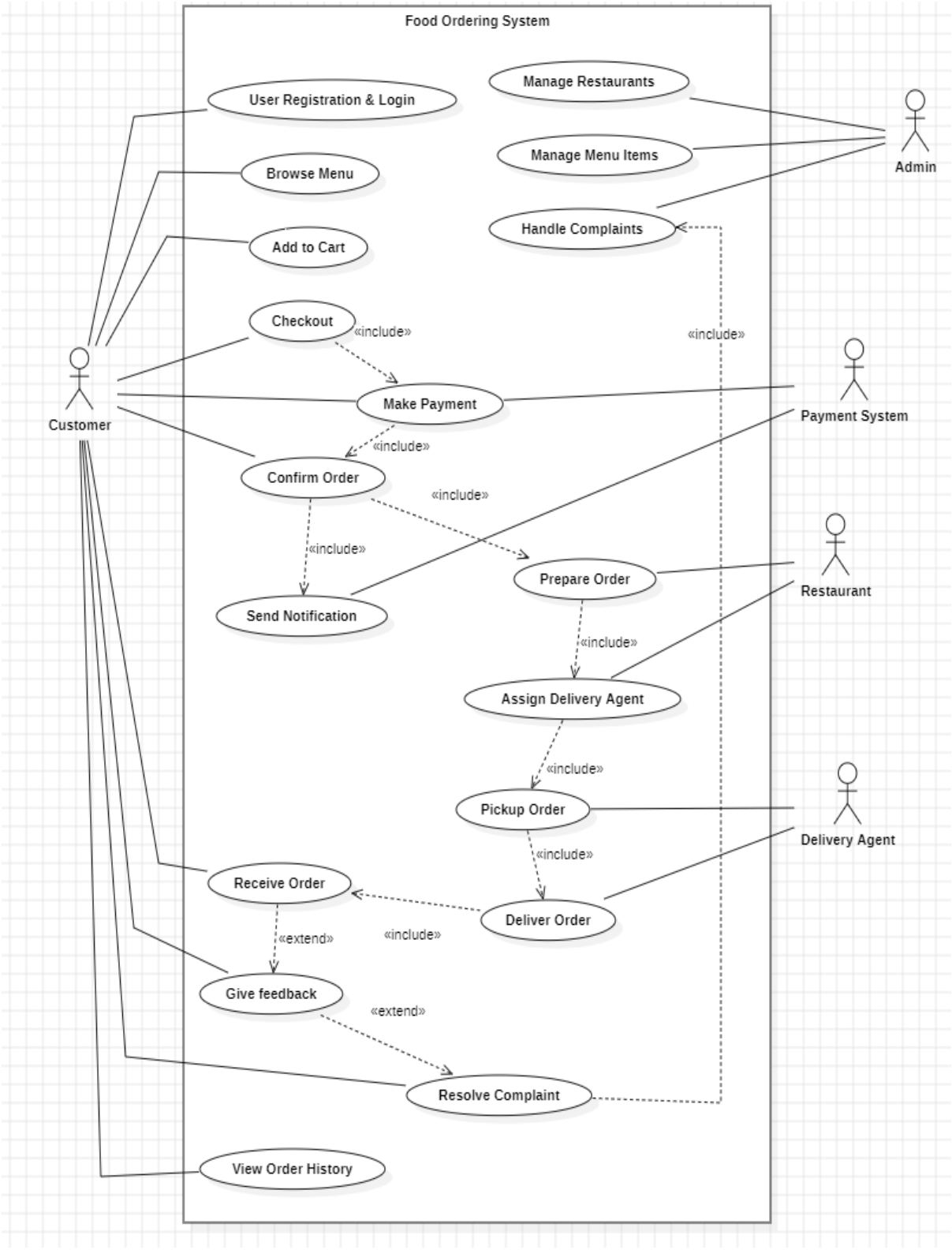
Includes:

- **Checkout** → *includes* → **Make Payment**
- **Make Payment** → *includes* → **Confirm Order**
- **Confirm Order** → *includes* → **Send Notification**
- **Confirm Order** → *includes* → **Prepare Order**

- **Prepare Order** → *includes* → **Assign Delivery Agent**
- **Assign Delivery Agent** → *includes* → **Pickup Order**
- **Pickup Order** → *includes* → **Deliver Order**
- **Receive Order** → *includes* → **Give Feedback**
- **Handle Complaints** → *includes* → **Resolve Complaint**

Extends:

- **Receive Order** → *extends* → **Give Feedback**
- **Give Feedback** → *extends* → **Resolve Complaint**


Food Ordering System
User Registration & Login
Browse Menu
Add to Cart
Checkout
Make Payment
Confirm Order
Send Notification
Prepare Order
Assign Delivery Agent
Pickup Order
Deliver Order
Receive Order
Give feedback
Resolve Complaint
View Order History
Manage Restaurants
Manage Menu Items
Handle Complaints
Customer
Admin
Payment System
Restaurant
Delivery Agent
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«extend»
«extend»

3) Class Model

Class Diagram for Food Ordering System

This UML class diagram provides a structured representation of the system's components, defining how different entities interact with each other.

Step 1: Identify Classes The following classes have been identified in the Food Ordering System UML Diagram:

1. **User**
2. **Customer** (inherits from User)
3. **Admin** (inherits from User)
4. **Order**
5. **Cart**
6. **Restaurant**
7. **MenuItem**
8. **Payment**
9. **DeliveryAgent**

Step 2: Define Attributes and Methods

1. **User**
    - Attributes:
        - id: int
        - name: string
        - email: string
        - password: string
    - Methods:
        - register()
        - login()
2. **Customer** (inherits User)

- Attributes:
  - address: string
  - phoneNumber: string
  - pinCode: string
  - offerLimit: float
  - pastOrderDate: date
  - multipleTransaction: boolean
- Methods:
  - placeOrder()
  - trackOrder()
  - browseMenu()
  - makePayment()
  - giveFeedback()

3. **Admin** (inherits User)
   - Methods:
     - manageRestaurants()
     - handleComplaints()
     - manageUsers()
4. **Order**
   - Attributes:
     - orderID: int
     - status: string
     - customerID: int
     - restaurantID: int
     - customerAddress: string
     - hotelAddress: string

        - phoneNo: string
        - pickupName: string
        - timeLimit: string
    - Methods:
        - updateStatus()
        - cancelOrder()
5. **Cart**
    - Attributes:
        - cartID: int
        - customerID: int
        - items: MenuItem[]
        - Total Amount: double
    - Methods:
        - addItem()
        - removeItem()
        - checkout()
6. **Restaurant**
    - Attributes:
        - restaurantID: int
        - name: string
        - location: string
        - menu: MenuItem[]
    - Methods:
        - updateMenu()
        - processOrder()
7. **MenuItem**

- Attributes:
  - itemID: int
  - name: string
  - price: float
  - description: string
- Methods:
  - updateDetails()
  - setAvailability()

8. **Payment**
   - Attributes:
     - id: int
     - amount: double
     - paymentMethod: string
     - status: string
     - bankAccountNo: string
     - transactionDate: date
     - balance: float
   - Methods:
     - processPayment()
     - refund()
9. **DeliveryAgent**
   - Attributes:
     - agentID: int
     - name: string
     - vehicleDetails: string
     - Status: string

- Methods:
  - pickupOrder()
  - deliverOrder()

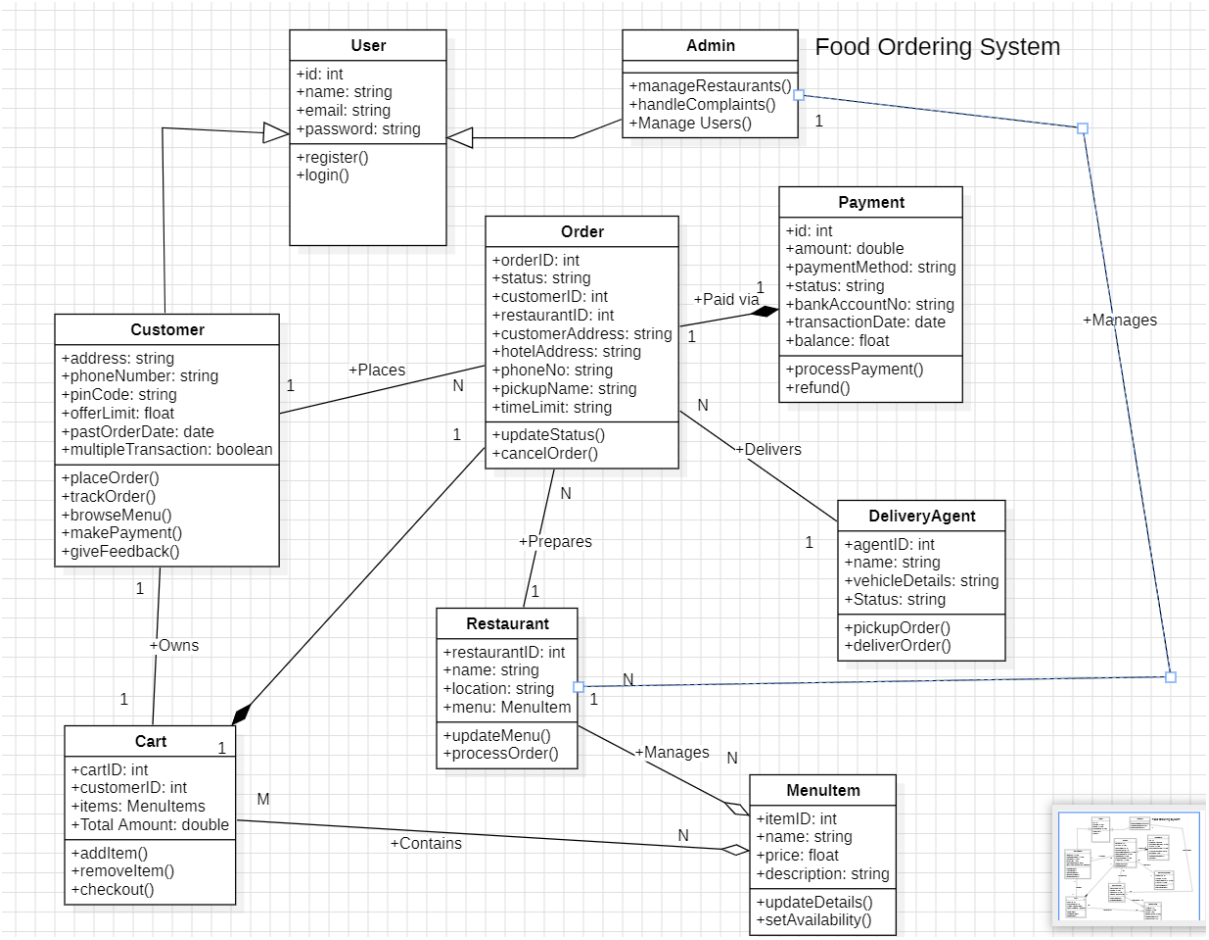
Step 3: Define Relationships

1. **Generalization (Inheritance)**
   - Customer and Admin inherit from User (shown with an arrow pointing to User)
   - Reason: Both customer and admin share common user attributes but have distinct responsibilities.
2. **Association**
   - Customer places an Order.
   - Order is prepared by a Restaurant.
   - Order is delivered by a DeliveryAgent.
   - Order is paid via Payment.
   - Admin manages Restaurants.
   - Reason: These relationships involve separate entities interacting without ownership.
3. **Aggregation**
   - Restaurant contains multiple MenuItems (hollow diamond near Restaurant).
   - Cart contains multiple MenuItems.
   - Reason: A restaurant and a cart can exist independently of their menu items.
4. **Composition**
   - Customer owns a Cart (filled diamond near Customer).
   - Reason: A cart is tightly bound to a customer and cannot exist without one.

Step 4: Define Cardinality and Reasoning

1. **Customer - Order (1:N)**
    - One customer can place multiple orders, but each order belongs to only one customer.
2. **Order - Payment (1:1)**
    - Each order has exactly one payment associated with it.
3. **Order - DeliveryAgent (N:1)**
    - One delivery agent can deliver multiple orders, but an order is assigned to only one agent.
4. **Order - Restaurant (N:1)**
    - One restaurant can handle multiple orders, but each order belongs to only one restaurant.
5. **Restaurant - MenuItem (1:N)**
    - A restaurant can have multiple menu items, but each menu item belongs to only one restaurant.
6. **Customer - Cart (1:1)**
    - Each customer has exactly one cart, and a cart belongs to only one customer.
7. **Cart - MenuItem (M:N)**
    - A cart can contain multiple menu items, but each menu item can be added to multiple carts.
8. **Admin - Restaurant (1:N)**
    - An admin can manage multiple restaurants, but each restaurant is managed by one admin.

Class Diagram:



4) Data Flow Model

Data Flow Diagrams for Food Ordering System

Level 0 (Context Flow Diagram)

Objective:

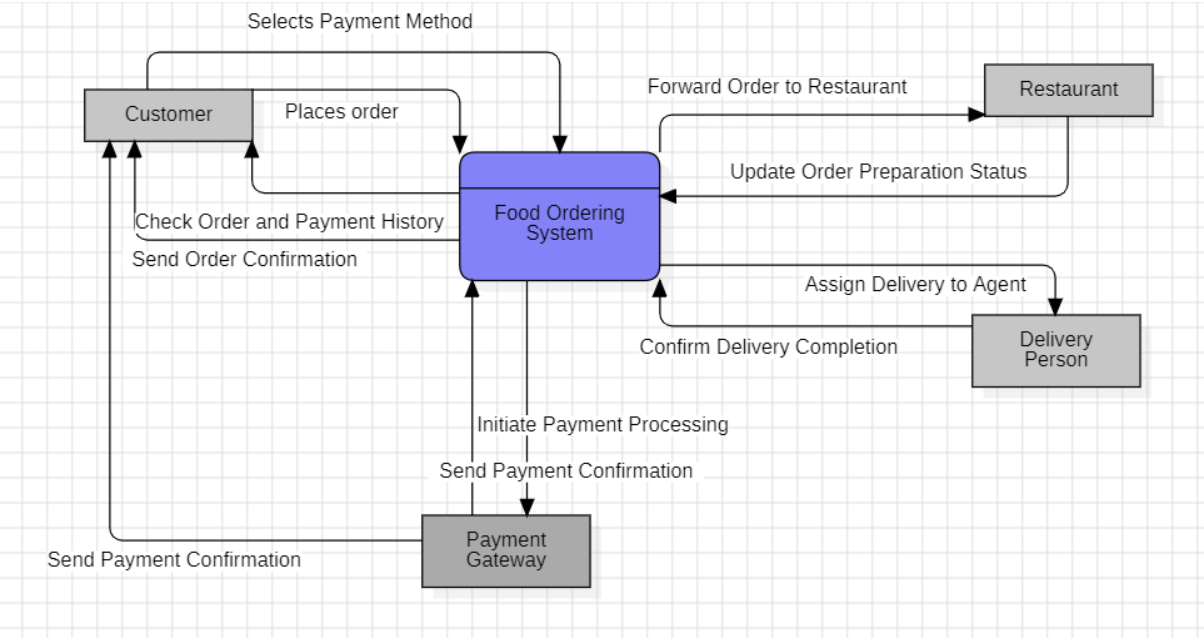
- Represents the entire Food Ordering System as a single process interacting with external entities.

Entities & Interactions:

- **Customer**: Places order, selects payment method, checks order and payment history, receives order confirmation.
- **Restaurant**: Receives order, updates preparation status.
- **Delivery Person**: Receives delivery assignment, confirms delivery completion.
- **Payment Gateway**: Processes payments, sends transaction confirmation.

Data Flows:

1. Customer selects payment method and places an order.
2. System forwards order to the restaurant.
3. Restaurant updates the order preparation status.
4. System assigns a delivery agent.
5. Delivery person confirms order delivery.
6. System initiates payment processing.
7. Payment gateway sends confirmation to the system and customer.
8. Customer checks order and payment history.



Level 1 DFD:

Overview

The Level 1 DFD expands upon the Level 0 DFD by breaking down the food ordering system into six subprocesses. These subprocesses interact with various external entities such as the customer, restaurant, delivery person, and payment gateway. The detailed data flows between these components ensure the seamless functioning of the system.

Subprocesses and Data Flows

1. Order Placement

- **Entities Involved:** Customer, Customer Management
- **Data Flows:**
    - The **Customer** selects items and places an order.
    - The system **validates the order** and processes **customer details**.
    - The system updates the **Customer Management module** with the order details and eligibility status.
    - Order confirmation is sent back to the **Customer**.

2. Order Processing

- **Entities Involved:** Order Placement, Customer Management, Order Preparation
- **Data Flows:**
    - The **validated order** from Order Placement is processed.
    - The **Customer Management module** provides **customer details**.
    - The order is forwarded to the **Order Preparation** module.

3. Order Preparation

- **Entities Involved:** Order Processing, Restaurant
- **Data Flows:**
    - The order is forwarded to the **Restaurant** for preparation.
    - The **Restaurant** updates the system with the **order preparation status**.

4. Payment Processing

- **Entities Involved:** Customer, Payment Gateway, Customer Management
- **Data Flows:**
    - The **Customer** selects a payment method.
    - The payment transaction is initiated through the **Payment Gateway**.
    - The **Payment Gateway** sends a **payment confirmation** back to the system.
    - The transaction history is updated in the **Customer Management** module.
    - The system ensures that only **paid orders** proceed to delivery.
    - The customer receives the **payment confirmation**.

5. Order Delivery

- **Entities Involved:** Customer Management, Delivery Person
- **Data Flows:**
    - The **Customer Management module** updates the **order history**.

    - The system assigns a **Delivery Agent** to fulfill the order.
    - The system sends a **delivery assignment** to the **Delivery Person**.
    - The **Delivery Person** confirms delivery completion.

6. Customer Management

- **Entities Involved:** Order Placement, Order Processing, Payment Processing, Order Delivery
- **Data Flows:**
    - This module maintains and updates **customer order history**.
    - It ensures that **transaction history** is properly recorded.
    - It processes **customer details** to verify eligibility for order processing.

External Entities and Their Relationships

Customer

- **Interacts with:** Order Placement, Payment Processing, Order Delivery
- **Data Flow:**
    - Places an order
    - Selects a payment method
    - Receives order and payment confirmations

Restaurant

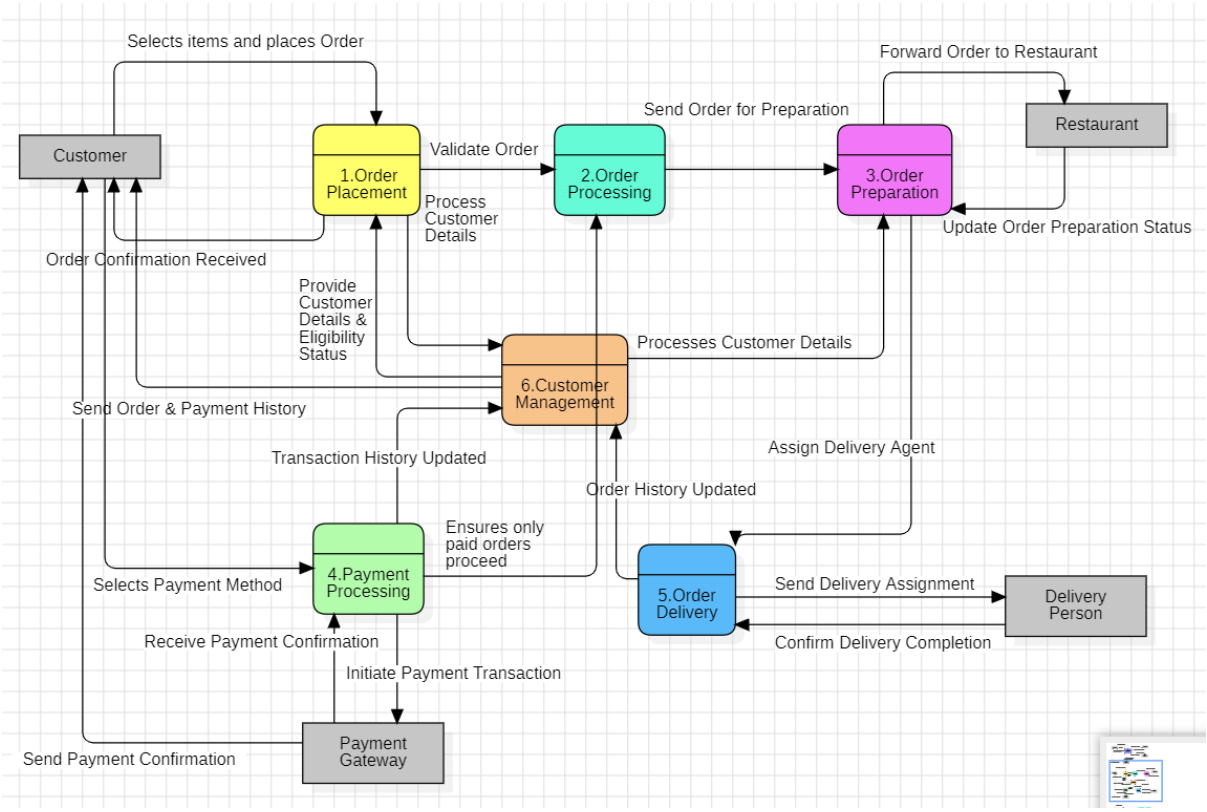
- **Interacts with:** Order Preparation
- **Data Flow:**
    - Receives orders for preparation
    - Updates order status

Payment Gateway

- **Interacts with:** Payment Processing
- **Data Flow:**
    - Initiates and confirms transactions

Delivery Person

- **Interacts with:** Order Delivery
- **Data Flow:**
    - Receives delivery assignment
    - Confirms order delivery



Level 2 DFD

Objective:

- Provides a more detailed breakdown of Level 1 processes into sub-processes while maintaining data flow balance.

Detailed Processes & Data Flows:

1. Order Placement:

- Customer browses menu.
- Selects items.
- Confirms order.
- Order stored in "Order Database".

2. Payment Processing:

- Customer selects payment method.
- System sends payment details to "Payment Gateway".
- Payment gateway verifies transaction.
- Payment confirmation received and stored in "Payment Transactions".

3. Order Preparation:

- System forwards order to restaurant.
- Restaurant processes order.
- Status updated as "preparing".
- Once ready, status updated as "ready for delivery".

4. Order Delivery:

- System assigns delivery person.
- Delivery person picks up order.
- Delivery status updated in "Delivery Records".
- Customer receives order and confirms delivery.

5. Customer Management:

- Stores customer details and past orders.
- Provides order history upon request.
- Validates customer details for order processing.

External Entities & Their Interconnections

The **external entities** involved in the system are:

1. **Customer** → Initiates the order, makes payments, and receives deliveries.
2. **Restaurant** → Prepares orders based on received requests.
3. **Delivery Person** → Picks up and delivers the order to the customer.
4. **Payment Gateway** → Processes payments and confirms transactions.

Each of these entities interacts with multiple subprocesses.

1. Order Placement (Customer → Order Processing)

- **1.1 Browse Menu → 1.2 Select Items** (Customer browses and adds items to the cart.)
- **1.2 Select Items → 1.3 Validate Order** (System checks item availability.)
- **1.3 Validate Order → 1.4 Process Customer Details** (Customer Management retrieves user details.)
- **1.4 Process Customer Details → 1.5 Confirm Order** (Checks discount eligibility, applies offers, and finalizes order.)
- **1.5 Confirm Order → 2.1 Fetch Order Details** (Sends validated order for further processing.)
- **1.5 Confirm Order → Customer** (Provides order confirmation message.)

External Entity: Customer

- Interacts with **Order Placement** to place an order.
- Receives order confirmation.

2. Order Processing (Internal System Handling Order Data)

- **2.1 Fetch Order Details → 2.2 Verify Order Information** (Ensures all order details are complete.)
- **2.2 Verify Order Information → 2.3 Assign Order ID** (Generates a unique tracking ID.)
- **2.3 Assign Order ID → 2.4 Send for Preparation** (Order is forwarded to the restaurant for cooking.)
- **2.4 Send for Preparation → 3.1 Receive Order Request** (Order details are passed to the restaurant.)

3. Order Preparation (Restaurant → Order Delivery)

- **3.1 Receive Order Request → 3.2 Start Preparation** (Restaurant starts cooking.)
- **3.2 Start Preparation → 3.3 Update Order Status** (Updates status: "Preparing" → "Ready".)
- **3.3 Update Order Status → 3.4 Notify Order Completion** (Sends notification that food is ready for pickup.)
- **3.4 Notify Order Completion → 5.1 Assign Delivery Agent** (Triggers order delivery process.)

External Entity: Restaurant

- Provides order preparation updates.
- Sends notification when the food is ready.

4. Payment Processing (Customer → Payment Gateway)

- **4.1 Select Payment Method → 4.2 Validate Payment Details** (Customer enters payment details.)
- **4.2 Validate Payment Details → 4.3 Initiate Payment Transaction** (Sends payment request to gateway.)
- **4.3 Initiate Payment Transaction → Payment Gateway** (External payment processor verifies transaction.)
- **Payment Gateway → 4.4 Receive Payment Confirmation** (Confirms payment success/failure.)

- **4.4 Receive Payment Confirmation → 4.5 Update Order Payment Status**
  (Marks order as "Paid".)
- **4.5 Update Order Payment Status → 6.3 Update Order & Payment Records**
  (Stores transaction history.)
- **4.5 Update Order Payment Status → 2.3 Assign Order ID**
  (Ensures only paid orders proceed.)

External Entity: Payment Gateway

- Processes transactions and sends payment confirmation.

External Entity: Customer

- Initiates the payment and receives payment confirmation.

5. Order Delivery (Restaurant & Delivery Person → Customer)

- **5.1 Assign Delivery Agent → 5.2 Send Delivery Assignment**
  (System selects an available delivery person.)
- **5.2 Send Delivery Assignment → 5.3 Pickup Order**
  (Delivery person collects the food.)
- **5.3 Pickup Order → 5.4 Deliver Order**
  (Delivery person transports the order.)
- **5.4 Deliver Order → 5.5 Confirm Delivery Completion**
  (System updates order status as "Delivered".)
- **5.5 Confirm Delivery Completion → 6.3 Update Order & Payment Records**
  (Logs order completion in customer history.)
- **5.5 Confirm Delivery Completion → Customer**
  (Sends delivery confirmation notification.)

External Entity: Delivery Person

- Receives order assignment.
- Picks up and delivers food.
- Confirms order delivery.

External Entity: Customer

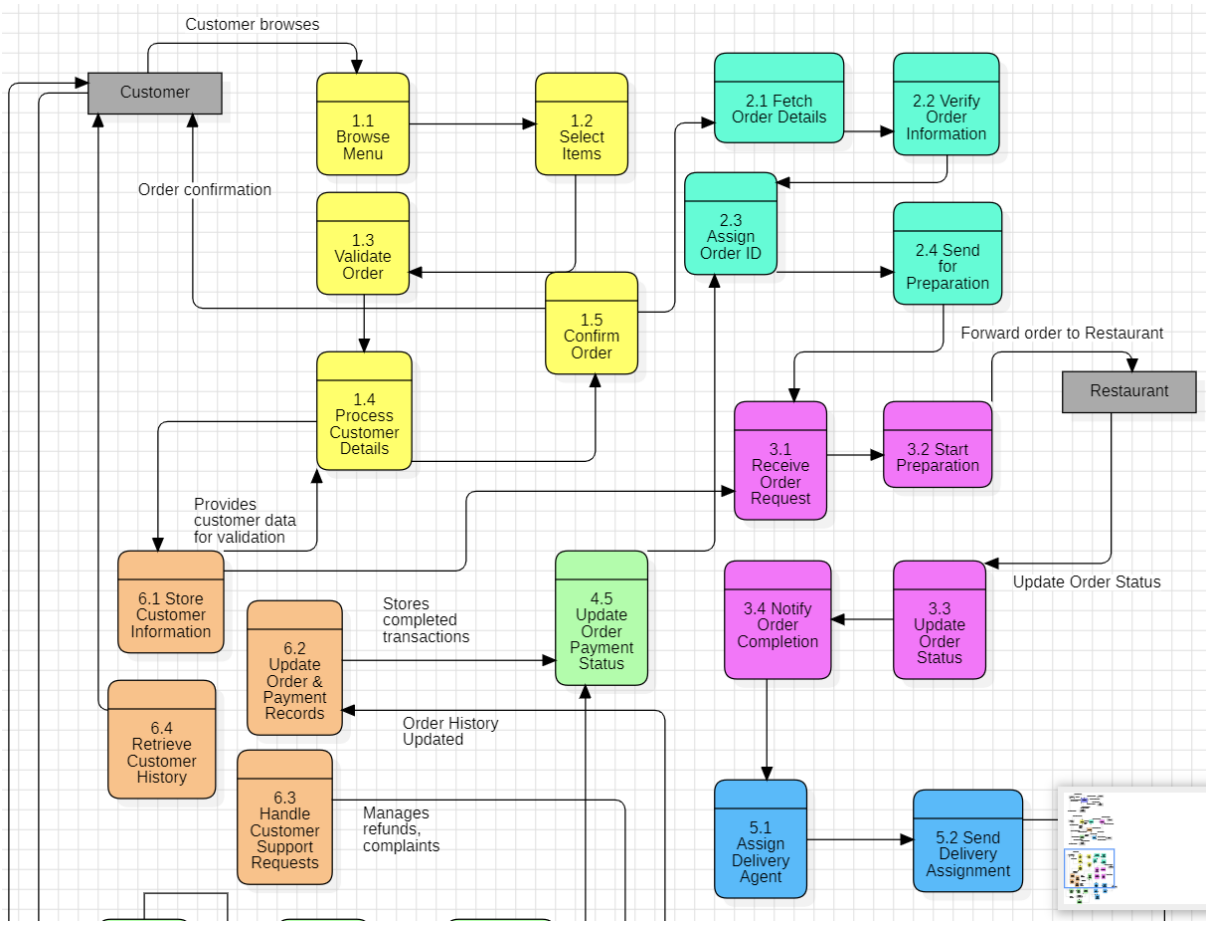
- Receives order delivery confirmation.
- Can provide feedback or raise issues.

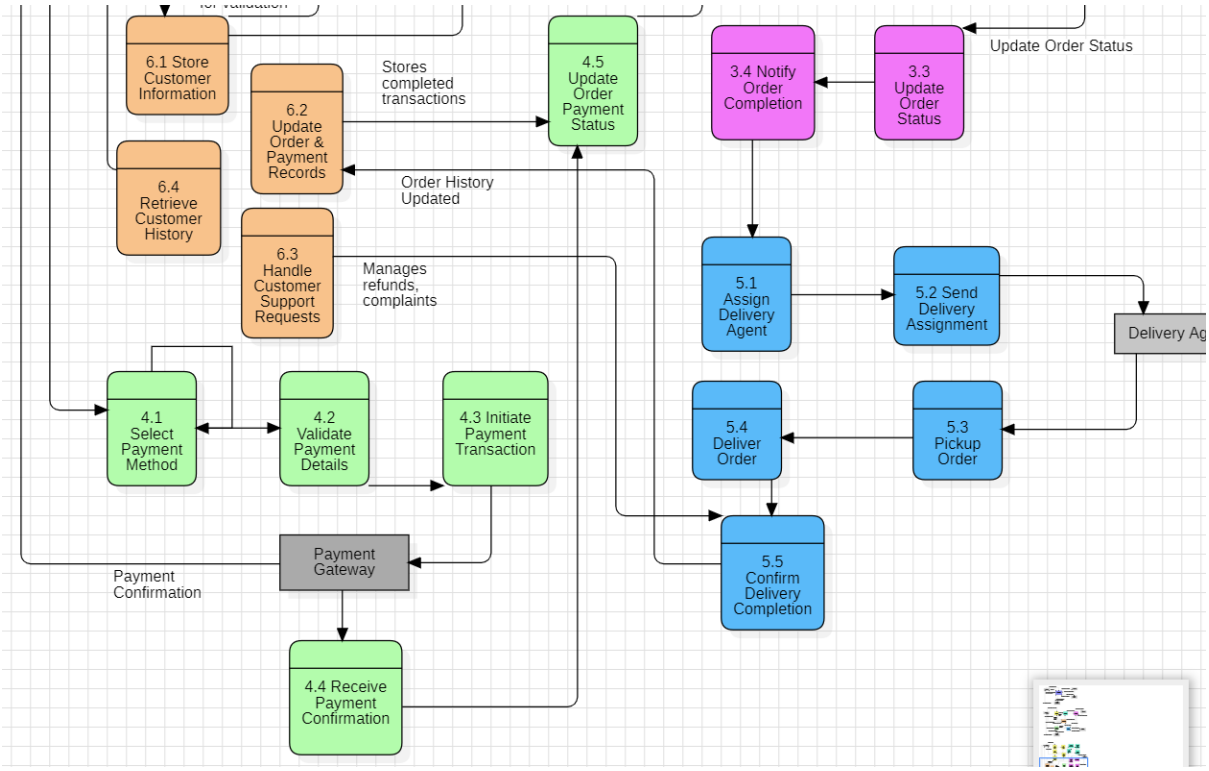
6. Customer Management (Tracking Order History & Issues)

- **6.1 Store Customer Information → 1.4 Process Customer Details**
  (Provides customer data for validation.)
- **6.2 Retrieve Customer History → 1.1 Browse Menu**
  (Suggests personalized recommendations.)
- **6.3 Update Order & Payment Records → 4.5 Update Order Payment Status**
  (Stores completed transactions.)
- **6.4 Check Discounts & Eligibility → 1.5 Confirm Order**
  (Applies offers before finalizing order.)
- **6.5 Handle Customer Support Requests → 5.5 Confirm Delivery Completion**
  (Manages refunds, complaints, or reassignment in case of failure.)

External Entity: Customer

- Can access past order history and request support.


Customer browses
Customer
1.1 Browse Menu
1.2 Select Items
2.1 Fetch Order Details
2.2 Verify Order Information
Order confirmation
1.3 Validate Order
2.3 Assign Order ID
2.4 Send for Preparation
1.5 Confirm Order
Forward order to Restaurant
Restaurant
1.4 Process Customer Details
3.1 Receive Order Request
3.2 Start Preparation
Provides customer data for validation
6.1 Store Customer Information
Stores completed transactions
4.5 Update Order Payment Status
3.4 Notify Order Completion
3.3 Update Order Status
Update Order Status
6.2 Update Order & Payment Records
6.4 Retrieve Customer History
Order History Updated
6.3 Handle Customer Support Requests
Manages refunds, complaints
5.1 Assign Delivery Agent
5.2 Send Delivery Assignment


6.1 Store Customer Information
Stores completed transactions
4.5 Update Order Payment Status
3.4 Notify Order Completion
3.3 Update Order Status
Update Order Status
6.2 Update Order & Payment Records
6.4 Retrieve Customer History
Order History Updated
6.3 Handle Customer Support Requests
Manages refunds, complaints
5.1 Assign Delivery Agent
5.2 Send Delivery Assignment
Delivery Ag
4.1 Select Payment Method
4.2 Validate Payment Details
4.3 Initiate Payment Transaction
5.4 Deliver Order
5.3 Pickup Order
Payment Gateway
Payment Confirmation
5.5 Confirm Delivery Completion
4.4 Receive Payment Confirmation

Conclusion:

The design and modelling of the **Food Ordering System** have been structured using a combination of **Unified Modelling Language (UML) diagrams** and **Data Flow Diagrams (DFDs)** to ensure a well-defined, scalable, and efficient architecture. Each model serves a distinct purpose in capturing different system aspects, providing clarity in functionality, data flow, and behavioural interactions

- The **Use Case Diagram** provides a **high-level view of system functionalities** from the perspective of different actors. It identifies the key users of the system, such as **Customers, Restaurants, Delivery Agents, Administrators, and Payment Gateways**, along with their interactions
- The **State Diagram** illustrates the **dynamic behavior of the Food Ordering System**, capturing different states an order goes through from initiation to completion.
- The **Class Diagram** represents the **object-oriented structure of the Food Ordering System**, defining the relationships between various entities in the system.
- The **Data Flow Diagrams (DFD)** map out the movement of data within the system, illustrating the transformation and interaction between different processes