**Machine Coding Round (120 minutes)**

**Problem Statement:**

**Design and implement a simple Food Delivery System in an object-oriented programming language of your choice.**

**The system should support the following operations:**

1. Register **CUSTOMERS** with name and address.
2. Register **RESTAURANTS** with a name, address, and **menu** (list of food items, each with name, price, and quantity).
3. Allow customers to view ***nearby restaurants and their menus.***
4. Let customers add food items to a **cart** and place an **order.**
5. Assign delivery agents to active orders on a ***first-come-first-serve basis.***
6. Track and update order status (ORDER\_PLACED → PREPARING → OUT\_FOR\_DELIVERY → DELIVERED).
7. Allow customers to view their **order history.**
8. Allow delivery agents to ***register and update the status of orders*** they are delivering.

**Implement this system with proper use of object-oriented principles, good code structure, and a menu-driven console interface.**

**Machine Coding Round: Food Delivery System**

**Duration**: 120 Minutes  
**Type**: Console-Based System (No UI)  
**Language**: Java / Python / Any OOP Language  
**Difficulty**: Medium-High

**🧾 Problem Statement**

Design and implement a simplified **Food Delivery System** that allows:

1. **Restaurants to register and offer food items.**
2. **Customers to place orders.**
3. **Delivery agents to deliver orders.**

Your system should support the following features:

**🔧 System Requirements**

**1. Restaurant Registration**

* Register a new restaurant with:
  + Name
  + Address
  + List of Food Items:
    - Item name
    - Price
    - Quantity available

**2. Customer Registration**

* Register a customer with:
  + Name
  + Address
  + Email / Phone (for search)

**3. Menu Browsing**

* Customers can browse nearby restaurants and see:
  + Menu items
  + Prices
  + Availability

**4. Place Order**

* Customer can:
  + Add items from one restaurant to cart
  + Place order
* Order status should progress:

rust

ORDER\_PLACED -> PREPARING -> OUT\_FOR\_DELIVERY -> DELIVERED

**5. Delivery Agent**

* Register delivery agents
* Assign agents to orders using first-come-first-serve
* Change order status as agent progresses

**6. Order History**

* View customer's previous orders with status and timestamps

**📦 Entities**

text

Customer

Restaurant

FoodItem

Order

DeliveryAgent

Cart

Address

OrderStatus (Enum)

**🔄 Operations (Console Menu)**

markdown

1. Register Restaurant

2. Register Customer

3. Register Delivery Agent

4. Show Restaurants

5. Show Menu (Restaurant ID)

6. Place Order (Customer)

7. Update Order Status (Agent)

8. Show Order History (Customer)

9. Exit

**💵 Optional: Payment (Bonus if time permits)**

* Support payment tracking (Paid/Unpaid)
* Calculate total cost
* Mark payment as done

**🧠 Bonus Features (Stretch Goals)**

* Sort restaurants by distance (hardcoded distance for simplicity)
* Search restaurants by name or cuisine
* Pagination in restaurant listing
* Multi-threading for order handling
* Save/load data to files

**✅ Evaluation Criteria**

| **Criteria** | **Weight** |
| --- | --- |
| OOP Design | 30% |
| Code Quality | 20% |
| Functional Coverage | 30% |
| Error Handling | 10% |
| Bonus Implementation | 10% |

**🧪 Sample Scenario**

text

Register Restaurant: "Biryani Express", "MG Road"

Add Item: Chicken Biryani ₹250 Qty: 10

Register Customer: "Ravi", "BTM Layout"

Customer Views Restaurant Menu

Customer Places Order: 1x Chicken Biryani

System assigns nearest available delivery agent

Agent updates status: PREPARING -> OUT\_FOR\_DELIVERY -> DELIVERED

Customer views order history.