Scenario

* The system for a local food delivery startup that allows users to browse restaurants, select food items, place orders, and track delivery.
* The business wants an app to handle customer orders, restaurant menus, delivery tracking, and payment.

Goals

* Customer register, browse menus, and order food.
* Restaurants manage their menus.
* Delivery agent receive and update delivery status
* Admin monitor the overall system

Overall System Flow & Functionality

* **User Registration & Login User**
  + - * Customers, Restaurant Owners, Delivery Agent, Admins
      * User Class is the Base Class
      * Interact with system by logging in
      * Polymorphism each subclass will implement specific versions of the viewDasboard() or similar.
  + Functionality
    - * + Register new users (store in database).
        + Login checks credentials (via login() method).
        + Directs the user to their appropriate interface(Customer menu , Admin dashboard)
* **Restaurant Management Managed by Restaurant Owners or Admins.**
  + - * Uses the Restaurant class
      * Each restaurant has a menu, which is a list of FoodItems objects
  + Functionality
    - * + Add/update/delete food items (addFoodItem(), updateMenu()).
        + View orders placed to that restaurant.
        + Manage opening/closing hours, delivery areas.
* **Menu Browsing & Food Selection Performed by the Customer.**
  + Functionality
    - * + After login, customers can viewMenu() of various restaurants.
        + They can filter/search food items by cuisine, rating, price.
        + Selected items are added to a cart (a temporary list before ordering).
* **Order Placement**
  + - * Customer uses placeOrder() to create a new Order object.
      * The order links:

The Customer

The Restaurant

A list of FoodItems

Total price (calculated via calculateTotal())

* + Functionality
    - * + Check if items are in stock.
        + Confirm availability.
        + Generate unique order ID and timestamp.
* **Payment Processing**
  + - * Handled via interface abstraction (PaymentMethod).
      * Implementations: UPIPayment, CardPayment, CashOnDelivery
  + Functionality
    - * + Customer selects payment method.
        + System uses polymorphism to invoke processPayment() from the selected class.
        + Payment status updates the Order object.
* **Order Assignment & Delivery**
  + - * Managed by the DeliveryAgent class**.**
      * Once payment is done, a DeliveryAgent is assigned.
      * The order is added to their assignedOrders list.
  + Functionality
    - * + pickOrder() — changes order status to "Out for Delivery"
        + markAsDelivered() — final status update, marks order as complete.

📦 Customer Orders Food

Customer logs in (Customer.login()).

They browse a restaurant’s menu.

Adds 2 FoodItems to a cart.

Places the order: placeOrder(cartItems, restaurant).

Order is created → Order object instantiated.

Customer pays via UPI → UPIPayment.processPayment().

Order marked as "Confirmed".

🚗 Delivery Flow

System assigns a DeliveryAgent.

Agent calls pickOrder(orderId) → order status: "Out for Delivery".

After delivery, agent calls markAsDelivered() → order status: "Completed".