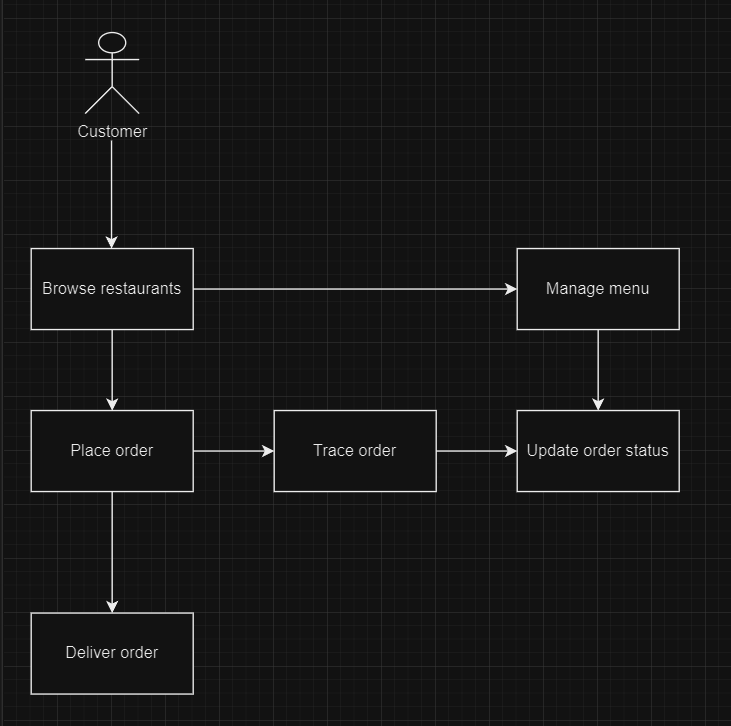
**Assignment 4   
SE-2217**

**Shaimardanov Yerkebulan**

**Food delivery service**

**TASK 1. USE CASE DIAGRAM TESTING**



The diagram visually represents:

* **Customer** can browse restaurants, place orders, and track orders.
* **Restaurant Owner** can manage the menu and update the order status.
* **Delivery Person** can update the delivery status

**TASK 2. ARCHITECTURE CHOICE & ITS JUSTIFICATION**

**Clean Architecture Diagram**

This diagram represents the layers of Clean Architecture in the context of the Online Food Delivery Service microservice.

**Explanation:**

1. **Entities**:
   * Order: Represents a customer's order.
   * Restaurant: Represents a restaurant.
   * MenuItem: Represents an item on a restaurant's menu.
2. **Use Cases**:
   * PlaceOrder: Use case for placing an order.
   * BrowseRestaurants: Use case for browsing available restaurants.
   * TrackOrder: Use case for tracking an order.
   * ManageMenu: Use case for restaurant owners to manage their menu.
   * UpdateOrderStatus: Use case for updating the status of an order.
   * DeliverOrder: Use case for delivery persons to update the delivery status.
3. **Interface Adapters**:
   * HTTP Handlers: Adapters that handle HTTP requests and map them to use cases.
   * Database Repositories: Adapters that interface with the database to perform CRUD operations.
4. **Frameworks and Drivers**:
   * Router: Sets up HTTP routes.
   * Database: Connects to the database (e.g., SQL, NoSQL).

**Clean Architecture Code Structure**

**cmd/**

**└── food-delivery-service/**

**└── main.go**

**internal/**

**├── entities/**

**│ ├── order.go**

**│ └── menu\_item.go**

**│**

**├── usecases/**

**│ ├── place\_order.go**

**│ ├── browse\_restaurants.go**

**│ ├── track\_order.go**

**│ ├── manage\_menu.go**

**│ ├── update\_order\_status.go**

**│ └── deliver\_order.go**

**│**

**├── adapters/**

**│ ├── handlers.go**

**│ └── repository.go**

**│**

**└── frameworks/**

**├── router.go**

**└── database.go**