Scenario: **Food Delivery System**

**Problem Statement**

Today's food delivery market demands fast, reliable service with a wide variety of restaurants.

Managing orders across restaurants, coordinating delivery partners, and ensuring food quality during transit creates significant operational complexity.

**Objective**

Allow customers to order food easily from nearby restaurants.

Enable real-time delivery tracking.

Match delivery partners efficiently to orders.

Ensure order accuracy and food quality.

# Strategic Design Principles.

1. **Collaboration between business experts and technical team**
2. **Domain Scope**

(Define the area / end users for the application)

1. **Domain description**

(Describing the complete domain briefly including code business domains and functionalities.)

1. Ubiquitous Language

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| Order | A customers food request placed |
| Cart | Temporary container for items before checkout |
| MenuItem | An item offered by a restaurant |
| Restaurant | A place where food is prepared and packed |
| Payment | Transaction for the order |
| Delivery Address | Location where food is delivered |
| Deliver Partner | Person delivering the order |
| Order Status | Current stage of the order |

1. Domain analysis
   1. Core domain

* Order Management
* Delivery Management
  1. Supporting subdomain
* Restaurant management
* Customer management
* Delivery partner management
* Notification system
* Payment Handling
  1. Generic subdomain
* Authentication and Authorization
* Location Services Integration (Google Maps)
* Audit Logging
* Promotion and Discount Management

1. Bounded Contexts

|  |  |  |
| --- | --- | --- |
| **Bounded Context** | **Responsibility** | **Interfaces with** |
| Order Management context | Handling order creation, validation and lifecycle | Restaurant, Delivery, Payment |
| Restaurant context | Manu management, availability of food, preparation time, | Order management, Delivery |
| Delivery assignment context | Assign order to delivery partner, optimize routes, track delivery status | Order management |
| Payment context | Handle payment, tips, and refunds | Order management |
| Notification context | Order status updates to customers, restaurants, delivery partners | All context |
| Promotion context | Manages discount coupons and promotions | Order management |

1. Context Mapping

* Order -> Restaurant, Delivery, Payment
* Restaurant <-> Delivery
* Notification <- Other contexts
* Order -> Payment Gateways, Location Services

# Tactical Design Principles

1. Key Entities

* Order -> orderId, customerId, restaurantId, deliveryId, status, totalAmount
* MenuItem -> itemId, name, description, picture, price
* Restaurant -> restaurantId, name, address, operatingHours
* Payment -> paymentId, amount, paymentMethod, status

1. Key Value Objects

* Address -> steet, city, postalCode, coordinates.
* Money -> amount, currency
* PaymentDetails -> cardNumber, expiry, cvv (encrypted)

1. Key Aggregates

* OrderAggregate -> Order, OrderItems, Payment, DeliveryStatus
* RestaurantAggregate -> Restaurant, MenuItem, Address
* DeliveryAggregate -> DeliveryPartner, Address, Restaurant, Customer

1. Key Repositories

* OrderRepository – Manage all order records
* RestaurantRepository -> Manage menu items and restaurant data
* DeliveryRepository -> Manage delivery partners, location updates.
* PaymentRepository -> Tract payments, tips, refunds.

1. Domain Events

* OrderPlaced -> Triggered when a customer places a new food order.
* OrderCancelled -> Triggered when a customer or restaurant cancels an order.
* OrderPrepared -> Triggered when the restaurant finishes preparing the order
* DeliveryPartnerAssigned -> Triggered when a delivery partner is assigned to pick up the order.
* OrderPickedUp -> Triggered when the delivery partner picks up the order from the restaurant.
* OrderDelivered -> Triggered when the delivery partner successfully delivers the foot to the customer.
* DeliveryFailed -> Triggered if the delivery could not be completed (customer unavailable, wrong address)
* PaymentReceived -> Triggered when payment from the customer is successfully completed.
* TipReceived -> Triggered when a customer gives a tip after delivery.
* RatingSubmitted -> Triggered when the customer submits a rating or feedback for the order, restaurant, or delivery partner.
* RefundInitiated -> Triggered if a refund id requested and initiated due to order issues.

1. Application Services

* OrderApplicationService -> Manage order placement, tracking and cancellation.
* RestaurantAppliationService -> Update menuitems, oprating hours, manage preparation time
* DeliveryApplicationService -> Delivery partner assignment, tracking live delivery
* PaymentApplicationService -> Payment processing, handle refund, and tips
* NotificationApplicationService -> Manage and send real-time notifications.