A food order and delivery system should offer a convenient and efficient way for users to explore, order, and receive meals from a wide range of local and international cuisines. It should provide an accessible and easy-to-navigate interface, available on both web and mobile platforms, where users can browse menus, read reviews, and customize orders to their preferences. The system should feature real-time order tracking, estimated delivery times, and notifications to keep users updated from the moment an order is placed until it is delivered. Integration with various payment gateways should ensure secure and flexible payment options. Additionally, the system should support promotional features, loyalty programs, and personalized recommendations based on user preferences and order history, enhancing the overall user experience and encouraging repeat business.

P1: Identify functional and non-functional requirements from the system description.

- Notational: 1-0/18 = 1
  - Functional, Non-Functional
- Contextual: 1-2/18 = 0.888
  - 10 Functional: Menu browsing, Order placement, Order customization, Review system, Real-time order tracking, Notifications, Multiple payment options, Promotional features, Loyalty program, Personalized recommendations.
  - 8 Non-functional: Accessibility, Multi-platform support, User-friendly interface, Performance, Security, Scalability, Reliability, Efficiency.
  - Scalability and Reliability not explicitly mentioned.

P2: Create a use case model for the system.

- Notational: 1-0/5 = 1
  - System Boundary (0.5), Actors (1.5), Use Cases (2), Relationships (1)
- Contextual: 1-0/16 = 1
  - Register/Login, Browse Restaurants and Menus, Place Order, Process Payment, Track Order, Receive Notifications, Manage Restaurant Profile, Update Menu, Manage Orders, Accept/Reject Order, Update Order Status, Pick Up and Deliver Order, Rate and Review, Manage Promotions and Loyalty Program, Generate Reports, Provide Customer Support.
  - All covered by requirements.

P3: Create use case specifications for identified use cases.

- Notational: 1-0/6.5 = 1
  - Use Case Name (0.5), Actor (1), Precondition (1), Postcondition (1), Main Scenario (2), Alternative Scenario (1)
- Contextual: 1-0/3 = 1
  - 3 UC Specs: Place Order, Track Order, Manage Orders

P4: Create a domain model based on use case specifications.

- Notational: 1-0/5 = 1
  - Classes (2), Attributes (1), Relationships (1), Multiplicities (1)
  - Must not have Data Types (-0.5), Operations (-0.5), Navigabilities (-0.5), Visibilities (-0.5)
- Contextual: 1-0/15 = 1
  - 12 classes: User, Customer, RestaurantOwner, DeliveryDriver, Restaurant, Menu, MenuItem, Order, OrderItem, Payment, Review, Promotion
  - 12 relationships: Customer places Order, Order contains OrderItem, OrderItem references MenuItem, Restaurant has Menu, Menu contains MenuItem, Restaurant is managed by RestaurantOwner, Order is delivered by DeliveryDriver, Order has Payment, Customer writes Review for Restaurant, Promotion applies to Order, Restaurant offers Promotion

P5: Identify system operations from use case specifications.

- Notational: 1-0/1.6 = 1
  - Base Use Case (0.2), Operation Name (1), Parameters (0.2), Return (0.2)
- Contextual: 1-0/38 = 1
  - 38 system operations: browseMenu(), addItemToCart(), removeItemFromCart(), updateCartQuantity(), calculateTotalPrice(), setDeliveryAddress(), selectPaymentMethod(), validateOrder(), processPayment(), confirmOrder(), generateOrderNumber(), notifyRestaurant(), retrieveOrderStatus(), getEstimatedDeliveryTime(), getDeliveryPersonLocation(), displayTrackingInformation(), contactSupport(), contactDeliveryPerson(), displayOrderList(), getOrderDetails(), acceptOrder(), rejectOrder(), updateOrderStatus(), markOrderReady(), assignDeliveryDriver(), notifyCustomer(), initiateRefund(), modifyOrder(), registerUser(), loginUser(), updateUserProfile(), createRestaurantProfile(), updateRestaurantProfile(), addMenuItem(), removeMenuItem(), updateMenuItem(), submitReview(), getRestaurantRating(), displayReviews(), createPromotion(), applyPromotion(), calculateLoyaltyPoints(), redeemLoyaltyPoints(), generateSalesReport(), generateCustomerActivityReport(), generatePopularItemsReport()

P6: Create design sequence diagrams for system operations.

- Notational: 1-0.2/4.6 = 0.956
  - Base System Operation (0.2), Participants (1), Operations (1), Parameters (0.2), Return Type (0.2), Sequence of Operation Calls (2)
  - Sequence diagrams are by use cases not by system operations (-0.2)
- Contextual: 1-0/3 = 1
  - 3 DSDs: placeOrder, trackOrder, manageOrder

P7: Create design class diagrams based on the domain model and sequence diagrams.

- Notational: 1-0.3/5.2 = 0.942
  - Classes(1), Attributes with Types(0.5), Operations (1), Parameters with Types(0.2), Return Type (0.2), Relationship(1), Multiplicities(0.5), Navigabilities(0.5), Visibilities(0.3)
  - No navigabilities
- Contextual: 1-(3*0.3)/15 = 0.94
  - 15 classes: User, Customer, RestaurantOwner, DeliveryDriver, Restaurant, Menu, MenuItem, Order, OrderItem, Payment, Review, Promotion, OrderController, RestaurantController, PaymentGateway
  - 11 relationships: Customer "places" Order, Order "contains" OrderItem, OrderItem "references" MenuItem, Restaurant "has" Menu, Menu "contains" MenuItem, Restaurant "is managed by" RestaurantOwner, Order "is delivered by" DeliveryDriver, Order "has" Payment, Customer "writes" Review for Restaurant, Promotion "applies to" Order, Restaurant "offers" Promotion
  - Missing ShopingCart, DeliveryService classes.
  - DeliveryDriver defined in domain class diagram, but differently as DeliveryService in DSD.
  - OrderController, Order, Restaurant, RestuarantController missing some methods.

P8: Develop a Java implementation for the system as specified in the class diagram and sequence diagrams.

- Notational: 1-(0.1+0.2+0.2+3*((3+7*0.4)/14))/7 = 0.751
  - Classes (0.5), Data types (0.5), Visibilities (0.5), Constructor (0.1), Getters (0.2), Setters (0.2), Methods (3)
  - Missing constructors, getters, setters.
  - 3 classes with no implementation: UserDetails, RestaurantDetails, MenuItemDetails,
  - 7 classes missing implementation for some methods: User, Customer, Restaurant, Menu, MenuItem, OrderController, PaymentGateway, DeliverService

- Contextual: 1-1/15 = 0.933
  - 14 classes: User, Customer, Restaurant, Menu, MenuItem, Order, OrderItem, OrderController, PaymentGateway, UserDetails, RestaurantDetails, MenuItemDetails, TrackingInfo, DeliverService
  - 1 enumeration: OrderStatus
  - DeliverService not defined in class diagram.

P9: Develop tests including unit tests, integration tests, and system tests for the implementation.

- Notational: 1-0/4 = 1
  - Unit tests (1), integration tests (1), system tests (1)
- Contextual: 1-0/5 = 1
  - 3 Unit Tests: testPlaceOrder_Success, testPlaceOrder_PaymentFailure, testTrackOrder
  - 1 Integration Tests: testOrderFlow
  - 1 System Tests: testCompleteOrderFlow