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: 1-1/11 = 0.909
  - Functional, Non-Functional
  - User Interface is non-functional
- Contextual: 1-3/11 = 0.727
  - 4 Functional: User Interface, Order Management, Payment Processing, Promotional and Loyalty Features.
  - 7 Non-functional: Usability, Performance, Security, Reliability, Scalability, Maintainability, Compatibility.
  - Reliability, Scalability, Maintainability not explicated 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: 1-0/14 = 1
  - Browse Menus, Read Reviews, Customize Order, Place Order, Track Order, Receive Notifications, Make Payment, Manage Promotions, Participate in Loyalty Program, Receive Personalized Recommendations, Manage Account, Update Order Status, Manage Restaurant Menu, Administer System.
  - All covered by functional requirements.

P3: Create use case specifications for identified use cases.

- Notational: 1-(1*0.3)/6.5 = 0.953
  - Use Case Name (0.5), Actor (1), Precondition (1), Postcondition (1), Main Scenario (2), Alternative Scenario (1)
  - Alternative scenario having no reference to basic flow.

- Contextual: 1-0/14 = 1
  - Browse Menus, Read Reviews, Customize Order, Place Order, Track Order, Receive Notifications, Make Payment, Manage Promotions, Participate in Loyalty Program, Receive Personalized Recommendations, Manage Account, Update Order Status, Manage Restaurant Menu, Administer System.

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

- Notational: 1-(0.5+0.5+0.5+0.5)/5 = 0.6
  - Classes (2), Attributes (1), Relationships (1), Multiplicities (1)
  - Must not have Data Types (-0.5), Operations (-0.5), Navigabilities (-0.5), Visibilities (-0.5)
  - Data types, Operations, Navigabilities, and Visibilities included.
- Contextual: 1-0/15= 1
  - 15 classes: User, Restaurant, Menu, MenuItem, Order, OrderStatus, Review, Payment, Promotion, LoyaltyProgram, Recommendation, Account, Notification, DeliveryPerson, SystemAdministrator.
  - 17 relationships:User can browse Menu (1), User can read Review (1), User can customize Order (1), User places Order (1), User tracks Order (1), User receives Notification (1), User makes Payment (1), User participates in LoyaltyProgram (1:1), User receives Recommendation (1), User manages Account (1:1), Restaurant has Menu (1:1), Restaurant has MenuItem (1), Restaurant manages OrderStatus (1), Restaurant receives Review (1), Restaurant manages Promotion (1), DeliveryPerson updates OrderStatus (1), SystemAdministrator administers System (1:1).

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: 1-0/29 = 1
  - 29 system operations: displayAvailableMenus(), displayReviews(restaurantID, dishID), displayCustomizationOptions(itemID), saveCustomizedOrder(userID, orderDetails), createOrder(userID, orderDetails), confirmOrderPayment(orderID, paymentDetails), fetchOrderStatus(orderID), updateOrderStatus(orderID, status), sendOrderNotification(userID, orderID, message), processPayment(orderID, paymentDetails), createPromotion(promotionDetails), updatePromotion(promotionID, promotionDetails), deletePromotion(promotionID), enrollInLoyaltyProgram(userID), updateLoyaltyPoints(userID, points), generateRecommendations(userID), updateAccountDetails(userID, accountDetails), changeOrderStatus(orderID, status), addMenuItem(restaurantID, menuItemDetails), updateMenuItem(itemID, menuItemDetails), deleteMenuItem(itemID), manageUserAccounts(adminID, userAccountDetails), manageSecuritySettings(adminID, securitySettings), monitorSystemPerformance(adminID).

P6: Create design sequence diagrams for system operations.

- Notational: 1-0/4.6 = 1
  - Base System Operation (0.2), Participants (1), Operations (1), Parameters (0.2), Return Type (0.2), Sequence of Operation Calls (2)
- Contextual: 1-0/14 = 1
  - 14 DSDs: Browse Menus, Read Reviews, Customize Order, Place Order, Track Order, Receive Notifications, Make Payment, Manage Promotions, Participate in Loyalty Program, Receive Personalized Recommendations, Manage Account, Update Order Status, Manage Restaurant Menu, Administer System.

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

- Notational: 1-(0.2+0.2+1+0.5+0.5)/5.2 = 0.538
  - 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 parameters, no return type, no relationships, no multiplicities, no navigabilities
- Contextual: 1-0/17 = 1
  - 17 classes: User, Account, LoyaltyProgram, Recommendation, Notification, Restaurant, Menu, MenuItem, Promotion, Review, Order, OrderStatus, Payment, DeliveryPerson, SystemAdministrator, PerformanceMonitor, Security.

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+2.5)/5 = 0.4
  - 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.
  - Methods in all classes missing implementation.

- Contextual: 1-0/16 = 1
  - 16 classes: User, Restaurant, Menu, MenuItem, Order, OrderStatus, Payment, Notification, Promotion, Review, LoyaltyProgram, Recommendation, DeliveryPerson, SystemAdministrator, PerformanceMonitor, Security.

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

- Notational: 1-0/3 = 1
  - Unit tests (1), integration tests (1), system tests (1)
- Contextual: 1-0/6 = 1
  - 3 Unit Tests: UserTest, RestaurantTest, OrderTest
  - 2 Integration Tests: UserRestaurantIntegrationTest, PaymentIntegrationTest
  - 1 System Tests: SystemTest