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: 1-0/12 = 1
  - Functional, Non-Functional
  - User Interface non-functional
- Contextual: 1-10/27 = 0.629
  - 5 Functional: User Management (2), Restaurant Management (3), Food Ordering (7), Order Tracking (3), Communication (2).
  - 7 Non-functional: Performance (2), Scalability (1), Availability (1), Usability (2), Security (2), Reliability (1), Maintainability (1).
  - 10 not explicitly mentioned: User registration and login, User profile management, Restaurant registration and profile creation, Menu management, Setting delivery areas and fees, In-app chat with restaurant, Scalability, Availability, Reliability, Maintainability

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: 1-0/9 = 1
  - 9 use cases: User Management, Restaurant Management, Browse and Search, Place Order, Manage Orders, Payment Processing, Communication, Review and Ratings, Promotions and Loyalty

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 label referencing to basic flow.
- Contextual: 1-0/1 = 1
  - 1 UC Specs: Place Order

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: 1-0/5 = 1
  - 5 classes: Customer, Restaurant, Menu Item, Order, Payment Method
  - 12 relationships: Customer can place many Orders, Order belongs to one Customer, Restaurant can have many Orders, Order belongs to one Restaurant, Restaurant has one Menu, Menu contains many Menu Items, Menu Item belongs to one Restaurant, Order contains many Order Items, Menu Item can be part of many Orders, Customer can have many Payment Methods, Order has one Payment Method, Menu Item can have many Customization Options

P5: Identify system operations from use case specifications.

- Notational: 1-0.4/1.6 = 0.75
  - Base Use Case (0.2), Operation Name (1), Parameters (0.2), Return (0.2)
  - No parameters, no return.
- Contextual: 1-8/27 = 0.703
  - 27 system operations: Customer Registration, Login, Update Profile, Browse Restaurants, Search Restaurants, View Menu, Add Item to Cart, Customize Order, Select Delivery/Pickup, View Order Summary, Select Payment Method, Submit Payment, Track Order, View Order History, Rate and Review, Restaurant Registration, Edit Restaurant Profile, Manage Menu, Set Delivery Areas, View Orders, Manage Delivery (optional), Order Management, Payment Processing, Push Notifications, Data Storage, Security
  - 8 not covered in UC: Update Profile, View Order History, Rate and Review, Restaurant Registration, Edit Restaurant Profile, Manage Menu, Set Delivery Areas, Data Storage

P6: Create design sequence diagrams for system operations.

- Notational: 1-(0.2+0.2+2)/4.6 = 0.478
  - Base System Operation (0.2), Participants (1), Operations (1), Parameters (0.2), Return Type (0.2), Sequence of Operation Calls (2)
  - No parameters, no return
  - Sequence unrecognizable
  - Graphical representation unrecognizable.
- Contextual: 1-(2*0.8)/2 = 0.2
  - 2 DSDs: Place Order, Manage Order
  - “Place Order” is not a system operation, but a UC.

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

- Notational: 1-(0.5+1*0.5+0.5+0.2+0.2+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 attributes, operation ownership unrecognizable, no parameters, no return type relationships unrecognizable, no multiplicities
  - Graphical notation is unrecognizable.
- Contextual: 1-(1+1*0.2)/8 = 0.85
  - 8 classes: Customer, Order Services, Payment Gateway, Restaurant, Delivery Driver, Web App (UI), Restaurant App (UI), Database
  - Relationships: No countable and unrecognizable.
  - Delivery Driver, manageMenu() in Restaurant App not covered in sequence diagrams.

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+2.9)/5 = 0.32
  - 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.
  - No method implementation except a little in OrderService.
- Contextual: 1-1/3 = 0.666
  - 3 classes: Customer, Order, OrderService
  - Order not covered in design.

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

- Notational: 1-(1*0.7+1*0.7+1*0.7)/3 = 0.3
  - Unit tests (1), integration tests (1), system tests (1)
  - Only textual description without implementation
- Contextual: 1-2/14 = 0.857
  - 8 Unit Tests: Customer registration (valid and invalid data), Update customer information, Order creation (various item combinations), Total order price calculation, Order status transitions, Valid order creation, Invalid order data handling, Payment processing (successful and failed transactions),
  - 2 Integration Tests: Database interaction (order saving and retrieval), Payment Gateway integration,
  - 4 System Tests: User registration, login, and order placement (end-to-end flow), Order confirmation and status updates, Restaurant order management (view and update order status), Delivery management (if applicable).
  - Payment Gateway in 1 integration test and Restaurant Management in the system test not covered in implementation.