CECS 575 - Group 7
Food Ordering System

GROUP MEMBERS

Saloni Punjabi - 027983629

Tejas Kale - 027960801

Jay Patel - 029362656

Dhrumil Shah - 027972839

Priyanshi Shah - 028836234

Shubham Banekar - 029363319

Swapnil Kawade - 018811037

TABLE OF CONTENTS

Q1 In a document write the name and a description of your system (one or two paragraphs)

I. DESCRIPTION

The Food Ordering application helps make your life easy by getting your favorite cuisine from your restaurant of choice right from your couch. It allows you to browse restaurants available near your locale, explore various food choices and get them delivered at your doorstep.

User registration is the first step in the application's workflow. The user enters vital information such as his or her name, phone number, and home delivery address. Once the user is registered, they can **view the list of available restaurants** serving near their location. Upon selecting a restaurant, they can **view the menu** and **add desired items to their food cart**. Once the items are confirmed, users can select the payment method and **place an order**.

This application implements the core business model for the food ordering application. By leveraging the principles of **Object Oriented Programming, UML diagrams and Design Patterns**, the principle operations of a food ordering service model can be implemented in an elegant way.

Q2 Create a github repository name cecs575_group# and share the link in the document from step1

II. GITHUB REPOSITORY LINK

Github Link: Github_CECS575_Group7

Q3 Identify four main use case scenarios for your application. Write two of them in a detailed (fully dressed) use case scenario format. The other two can be brief/casual. Also mention if the system has any major non-functional requirements (FURPS+ model)

III. IDENTIFY FOUR MAIN USE CASES

Following are the 4 main use cases of the Food Ordering application:

1. View Restaurants
2. View Menu
3. Add Items to Cart
4. Place Order

1. View Restaurants:

Use Case Name	**View Restaurants**
Actor	Primary: User/Customer, Secondary: System
Goal/Description	User searches / explores the restaurants in the application
Related Use Case	SearchFoodItems SearchRestaurants
Pre conditions	User should be logged in to the system to view the restaurants in the application
Post conditions	The restaurants are displayed in the system
Scenarios	1. Enter the category of the items to be searched for. 2. Enter the location of the restaurant to be searched in 3. View the displayed restaurants from the filters applied. 4. Select the desired restaurant.
Exception	1. The system displays an error message if the requested restaurant is not in the list. 2. The system displays the error message if the desired restaurant is unable to be selected by the user. 3. The system displays the error message if there is any other technical glitch observed in the system.
Priority	Essential
Frequency of use	Everytime a user wants to place an order

2. View Menu:

Use Case Name	**View Menu**
Actor	Primary: User/Customer, Secondary: System
Goal/Description	User explores the menu for the selected restaurant in the application.
Related Use Case	SearchFoodItems SearchRestaurants
Pre conditions	1. Users should be logged in to the system. 2. User selects a restaurant
Post conditions	The menu for the selected restaurant is displayed in the application
Scenarios	1. Search for a particular food item in the menu 2. View the food items separated by different categories 3. Read the optional description of the food items 4. View the available options / sides as a part of the selected food item
Exception	1. The system displays the error message if there is any other technical glitch observed in the system. 2. The system displays an error message if the item is out of stock.
Priority	Essential
Frequency of use	Everytime a user wants to place an order
Open Issues	Gray out the food item if out of stock

3. Add items to cart

Use Case Name	**Add Items to Cart**
Actor	Primary: User/Customer, Secondary: System
Goal/Description	The user adds the items to the cart which he/she wishes to order
Related Use Case	Search Items

	View Menu View Restaurants Select Items Display message for successful addition of item In the cart.
Pre conditions	User should be logged in to the system
Post conditions	The items are added in the cart by the user
Scenario	1. User searches the item type by typing in the search bar 2. The system displays the results obtained for the entered search criteria 3. The system displays the restaurant names from where the food can be ordered 4. The user then selects the restaurant of choice 5. System allows the user to select the desired option 6. User adds/selects the number of items required to be added in the cart 7. The system adds the required number of food items from the selected restaurant in the cart
Exception	1. The system displays error message if the requested food item is not in the list 2. The system displays the error message if the desired restaurant is unable to be selected by the user 3. The system displays the error message it the required number of quantity is unable to be added in the cart 4. The system displays the error message if there is any other technical glitch observed in the system
Priority	Essential
Frequency of use	Everytime the user wants to add an item in the cart
Open Issues	How much time should the user wait to be notified about the search result to be displayed for the restaurants, food items and the items getting added in the cart

4. Place Order

Use Case Name	**Place Order**
Actor	Primary: User/Customer, Secondary: System
Goal/Description	The user is able to place an order on the application
Related Use Case	Search Items View Menu View Restaurants Select Items Display Message For Successful Addition In Cart
Pre conditions	User logged in the application
Post conditions	The items are added in the cart by the user
Scenario	1. User searches the items/item type by typing in the search bar 2. The system displays the results obtained for the entered search criteria 3. The user selects the food/category to be ordered 4. The system displays the restaurant names from where the food can be ordered 5. The user then selects the restaurant of choice 6. System allows the user to select the desired option 7. User adds/selects the number of items required to be added in the cart 8. The system adds the required number of food items from the selected restaurant in the cart 9. User Selects the Payment method 10. User Places the Order
Exception	1. The system displays error message if the requested food item is not in the list 2. The system displays the error message if the desired restaurant is unable to be selected by the user 3. The system displays the error message it the required number of quantity is unable to be added in the cart 4. The system displays the error message if there is any other technical glitch observed in the system
Priority	Essential

Frequency of use	Everytime the user wants to add an item in the cart
Open Issues	How much time should the user wait to be notified about the search result to be displayed for the restaurants, food items and the items getting added in the cart

Non Functional Requirements:

- **Functionality**
  The application should be able to fulfill the food ordering requirements of the user.

- **Usability**
  The application should be intuitive. Users should easily be able to search for restaurants, food items, add items to the cart and place orders.

- **Reliability**
  The application should reliably perform the required operations with minimum downtime. An acknowledgement should be displayed whenever the user performs an important action. The application should allow the user to view restaurants, select food items and order anytime.

- **Performance**
  System should be performant and process operations under an acceptable response time. There should be a scope for scalability in case of adding new feature sets / workflows.

Q4 Draw a UML use case diagram depicting the use cases from step 1 and their relationship

IV. UML USE CASE DIAGRAM

1. View Restaurants

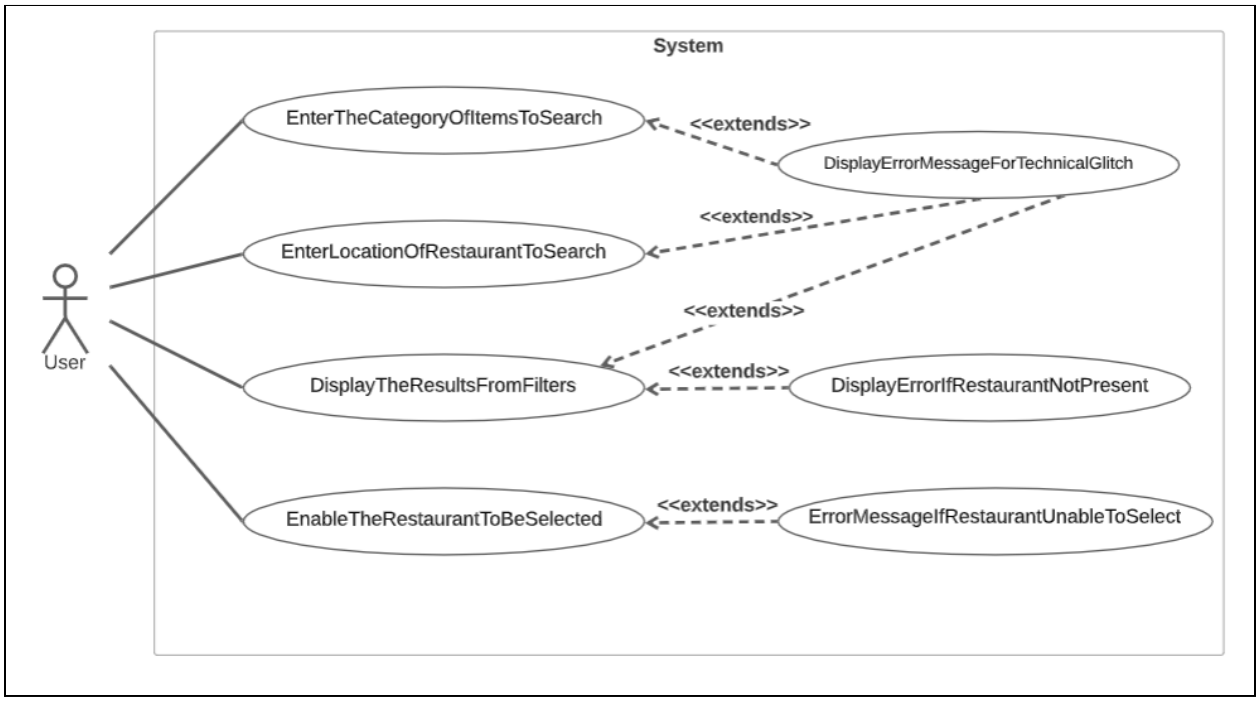

Fig 1.1: Use Case diagram for View Restaurants

2. View Menu

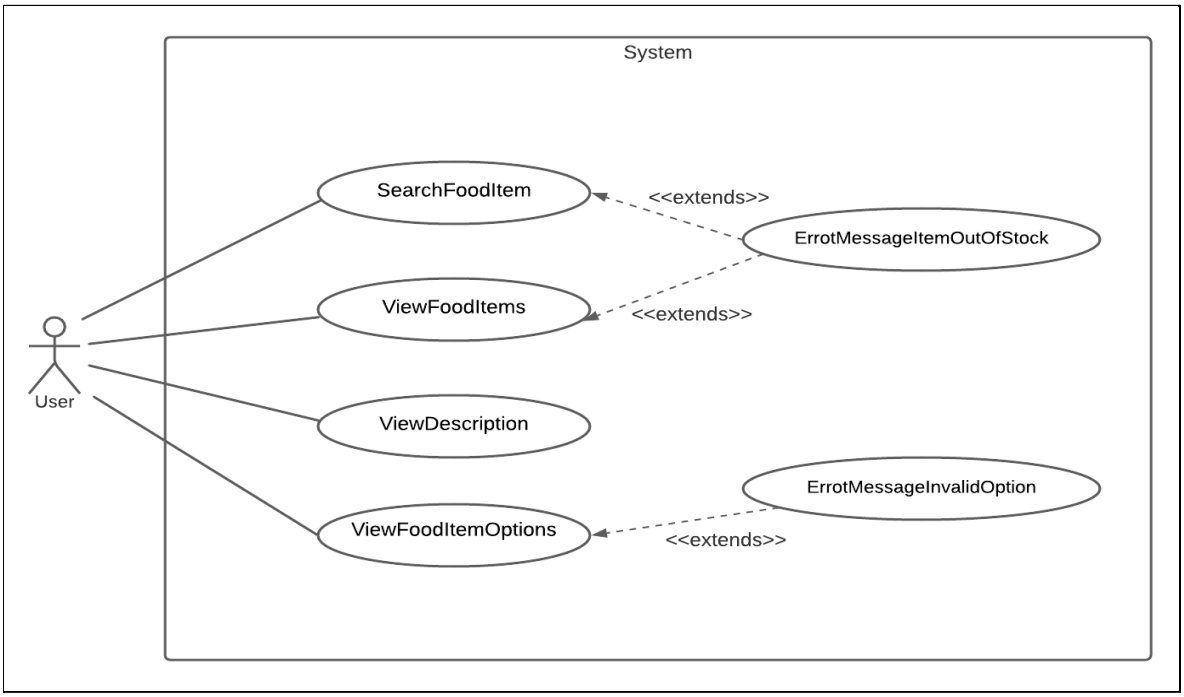

Fig 1.2: Use Case diagram for View Menu

3. Add items to Cart

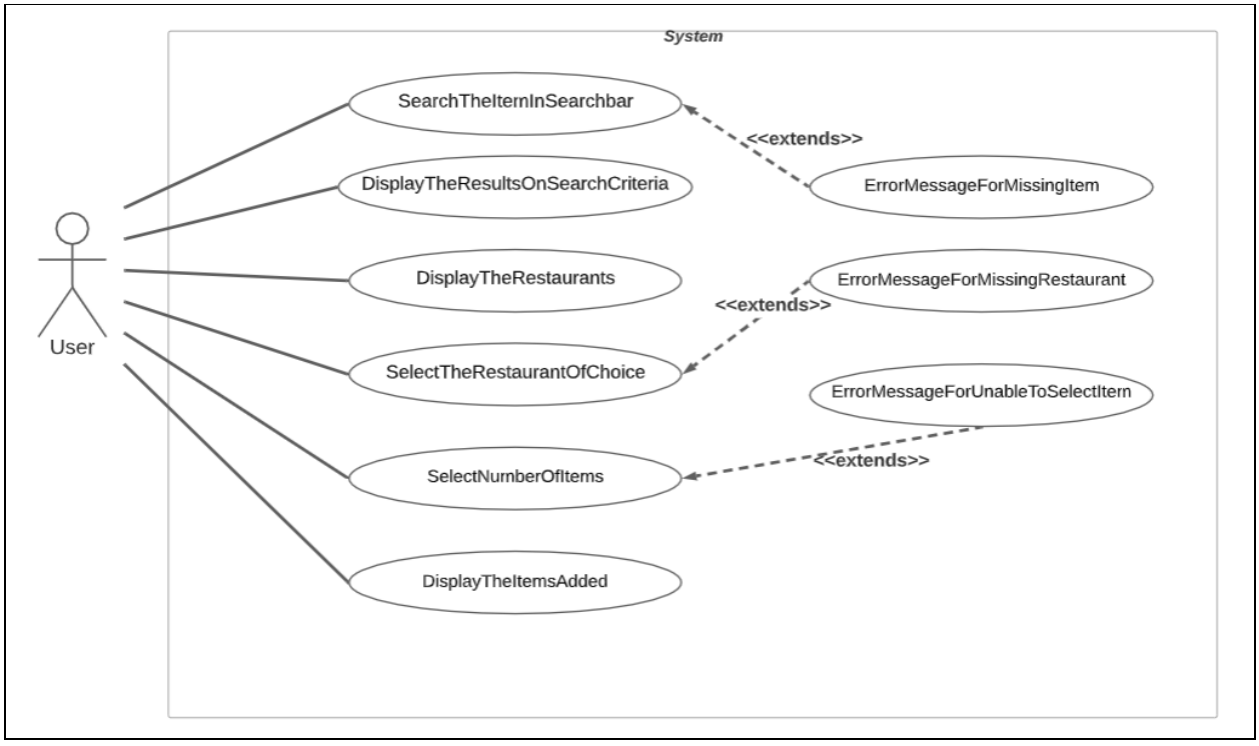

Fig 1.3: Use Case diagram for Add Items To Cart

4. Place Order

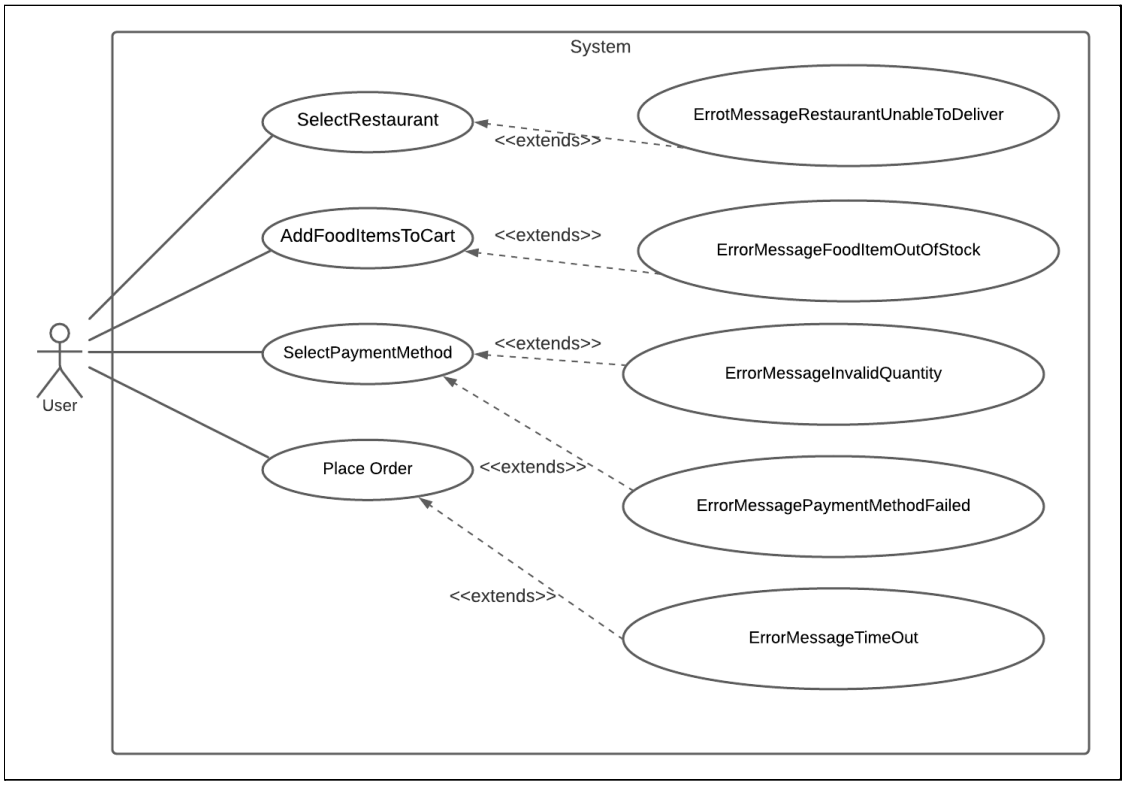

Fig 1.4: Use Case diagram for Place Order

Q5 Draw a UML system sequence diagram (SSD) for one use case (the main use case in your application)

V. UML SYSTEM SEQUENCE DIAGRAM (for Main Use Case)

1. Place Order

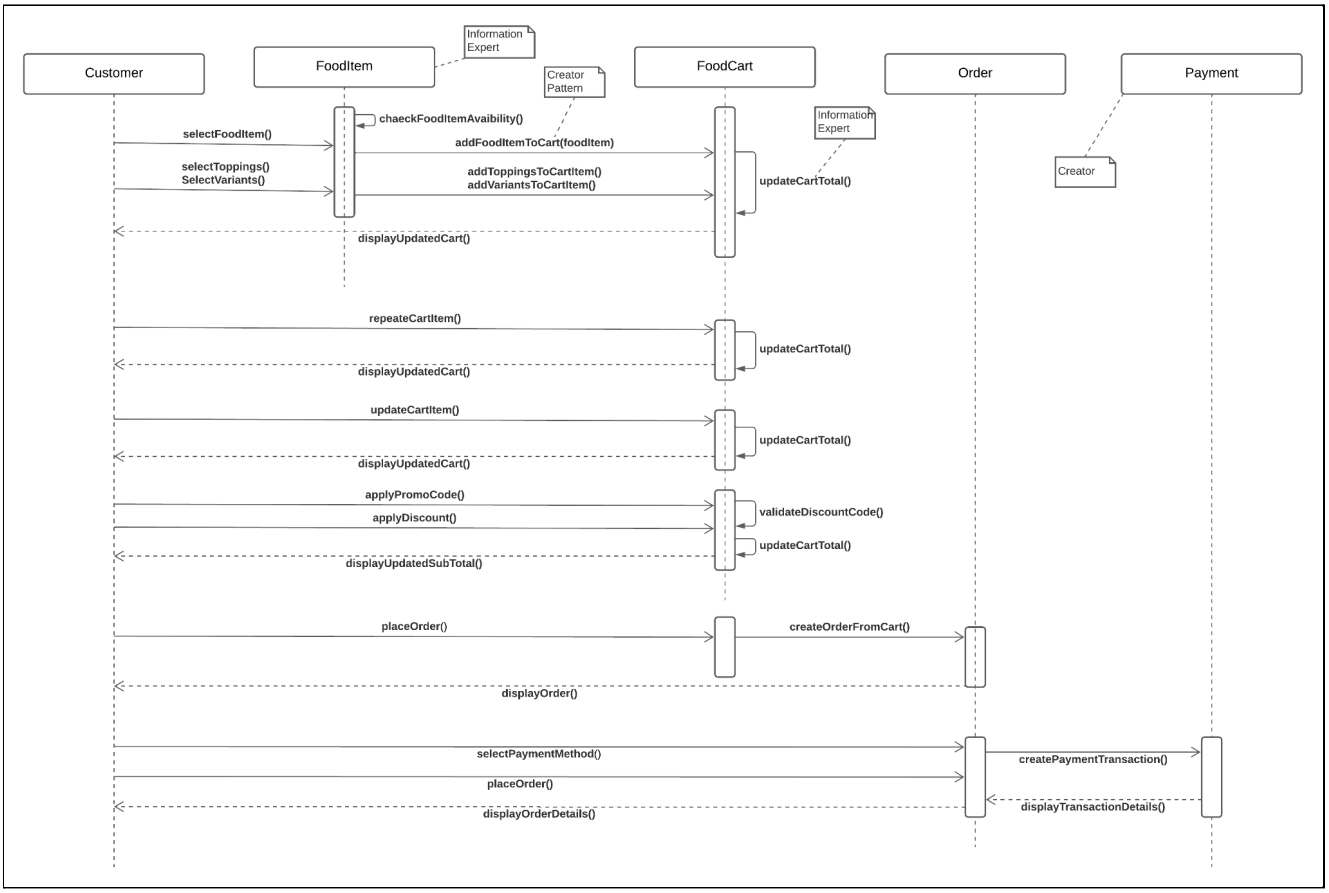

Fig 1.5: Sequence diagram for Place Order

Q6 Identify the domain objects in your system and draw a UML domain model

VI. DOMAIN OBJECTS AND UML DOMAIN MODEL

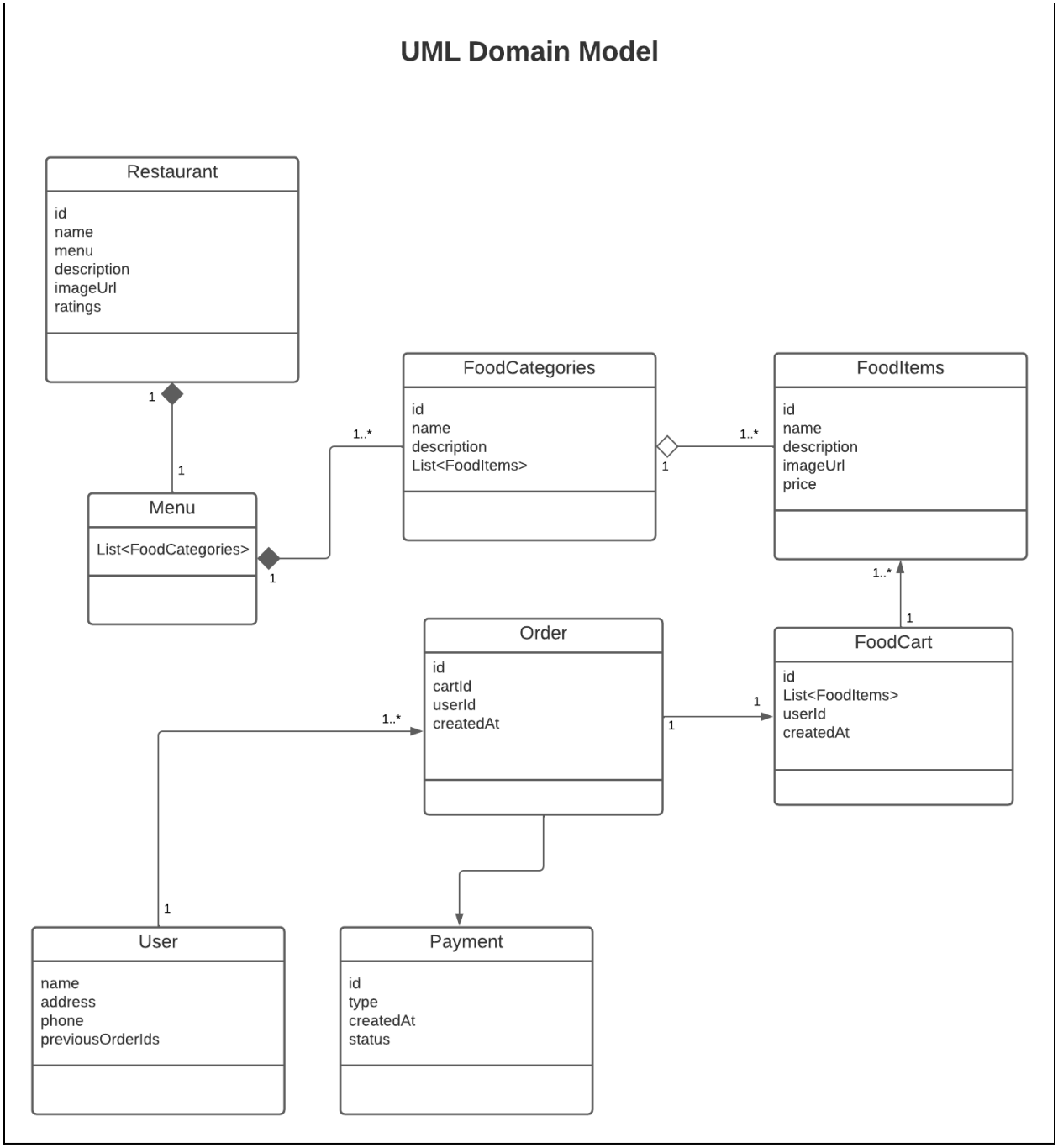

Fig 1.6: Domain Model for Food Ordering System

Q7 Using the 9 GRASP patterns, assign responsibilities to your class. Create the corresponding collaboration (three) or sequence diagrams (three). Annotate which GRASP pattern you applied in each scenario by UML comments (note symbol) in the diagrams. You need to apply at least four different GRASP patterns.

VII. GRASP PATTERNS

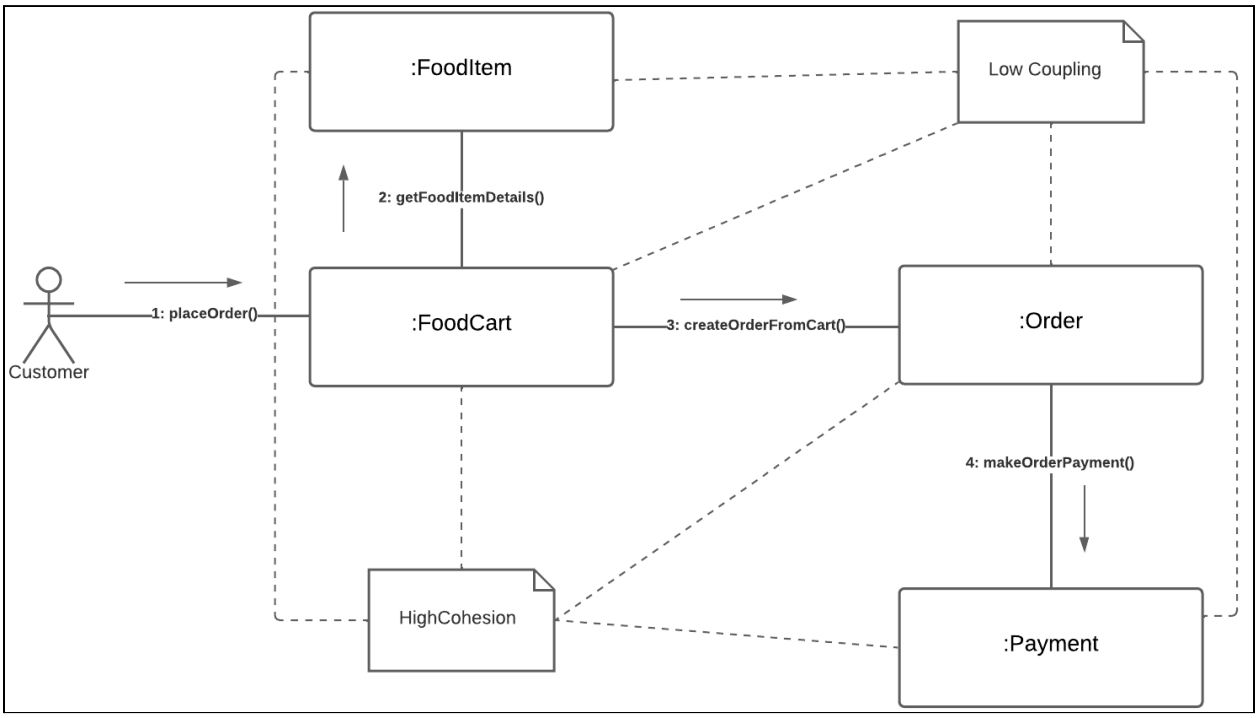

Fig 1.7: Collaboration Diagram for Place Order(High Cohesion and Low Coupling Patterns)

1. **High Cohesion**

   In order to reduce the complexities and minimize similar behaviors between the objects, we have created four different objects: FoodItem, FoodCart, Order and Payment. Thus, the objects will perform high cohesion by performing their individual responsibilities between them. For example, FoodCart will have a list of FoodItems and will be managed based on the customer's actions. Order object will be responsible for the subtotal and quantities of the items selected in FoodCart.

2. Low Coupling

Low coupling means our objects are more independent and isolated. If something is isolated we can change it without worrying that we have to change something else or whether we would break something. For example, the FoodItem object is dependent on the FoodCart object but it's independent from the order object. So any change made in the FoodItem object will not affect the past orders. Similarly, the Payment object is dependent on the Order object irrespective of the changes made in the FoodCart object.

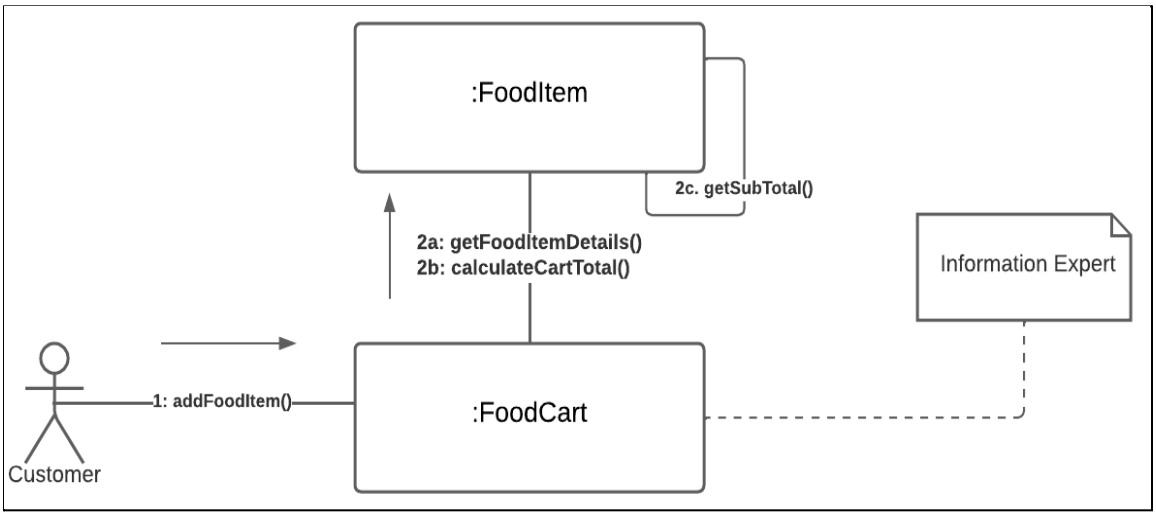

Fig 1.8: Collaboration diagram for Add Food Items (Information Expert Pattern)

3. Information Expert

In the System, the FoodCart class has references to all FoodItem so it is a natural candidate to take responsibility for calculating CartTotal. Here, the FoodCart object does not have the details of the FoodItem object. The Information Expert will assign the responsibilities to obtain the details of the FoodItem object, and thus the FoodCart will update the CartTotal based on prices and quantities of the food items.

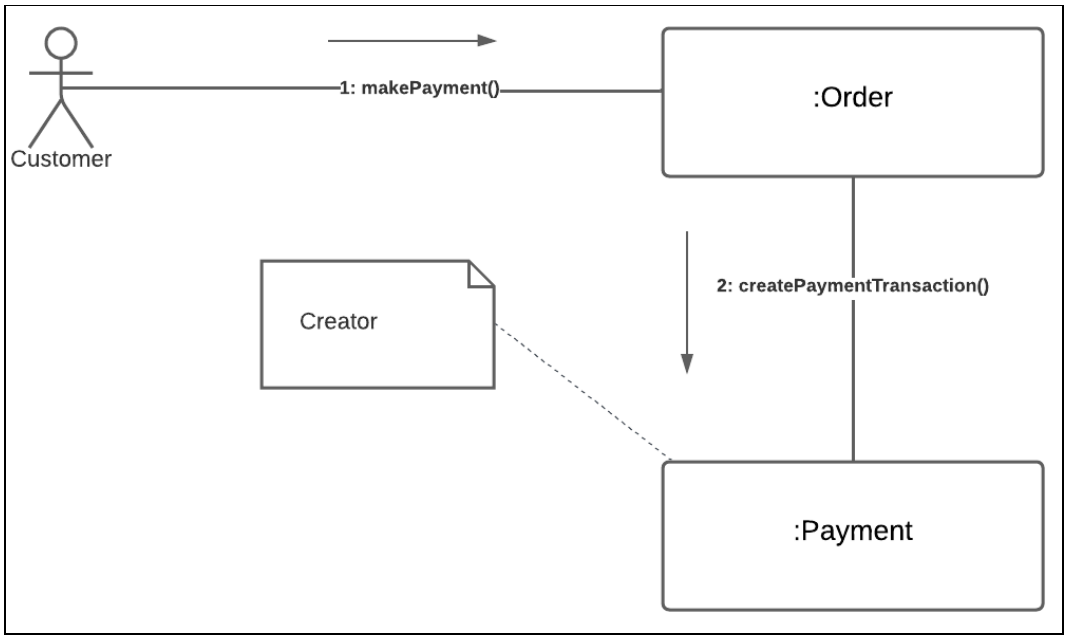

Fig 1.9: Collaboration Diagram for Make Payment (Creator Pattern)

4. Creator

As you can see above Order class compositely aggregates Payment (there is no Payment without Order), records Payment transactions, closely uses various payment methods and has initializing data passed by method parameters.

Q8 Create a Design class diagram for your system

VIII. DESIGN CLASS DIAGRAM FOR THE SYSTEM

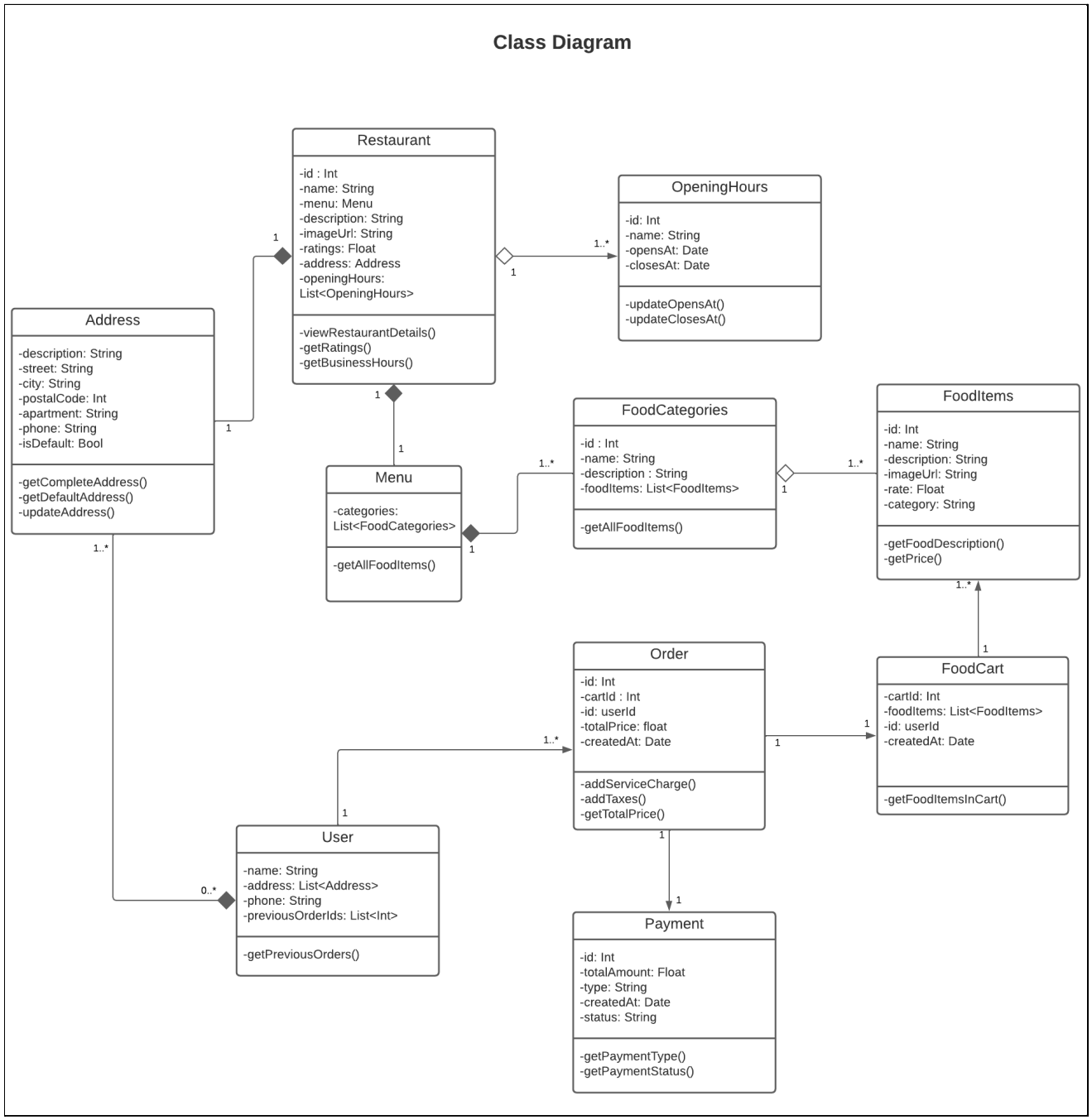

Fig 1.10: Class Diagram for Food Ordering System