
Software Requirements and Design Document

For

Food Delivery Management System

Prepared By:

Abdullah Saqib (20i-0458)

Rayed M. Saeed (20i-1822)

Fast NUCES Islamabad

27th November 2022



MAVERICKTECH

Table of Contents

1. Introduction.....	3
1.1 Purpose	3
1.2 Scope.....	3
1.3 Title	3
1.4 Objectives	3
1.5 Problem Statement	4
2. Overall Description.....	4
2.1 Product perspective	4
2.2 Product Functions	4
2.3 List of Use Cases.....	5
2.4 Extended Use Cases	5
2.5 Use Case Diagram:	32
3. Other Non-Functional Requirements	33
3.1 Performance Requirements.....	33
3.2 Safety Requirements	33
3.3 Security Requirements	33
3.4 Software Quality Attributes	33
3.5 Business Rules.....	33
3.6 Operating Environment.....	33
3.7 User Interfaces	33
4. Domain Model	53
5. System Sequence Diagrams	54
6. Sequence Diagrams	65
7. Class Diagram.....	86
8. Package Diagram.....	87
9. Deployment Diagram.....	87

1. Introduction

1.1 Purpose

The purpose of this product is to provide a reliable food delivery management system. It will have an attractive interface for the customer, restaurant, and rider alike. Using this product, we are tackling one of the major issues of the food industry which is a reliable and easy to use interface.

1.2 Scope

Domain: Local restaurants and consumers within Islamabad

Some of the already available food delivery systems are:

1. Cheetay
2. FoodPanda
3. careem food
4. uber eats
5. Jovi

The food delivery system will help firstly, the consumers that can benefit by having an easy and fast interface that will make ordering food easier than ever, secondly it will help the riders as they will be able to manage their rides and deliveries much easily, and lastly it will help the restaurants manage their orders and have an efficient and smooth ordering system.

- Customer interface where customers can order food and manage their cart
- Rider interface that will be showing the rider information and will help rider manage their deliveries.
- Restaurant information and menu
- Payment options
- Customer account information
- Adding new customers and riders
- Adding new food items in the menu

1.3 Title

Food Delivery System

Order from anywhere and we will deliver, Quick and reliable.

1.4 Objectives

An efficient system to help the restaurants to be able to accept orders quickly and easily.

Provide a system to the end users that can provide service to any area within Islamabad and Rawalpindi.

Provide the riders a better interface and better bonuses based on their hard work.

1.5 Problem Statement

Currently there is Food Panda that is the leading food delivery provider but the problem with it is that it has high delivery charges, and it does not deliver in all the sectors. For instance, if someone living in G-13 wants to order from a restaurant that is only available in let's say blue area then it would not even appear on their food panda app.

It would provide an efficient and better interface than the competition, and with the increasing traffic on such platforms due to ease that they provide to the people it is a much better opportunity to step into a growing market, with a better idea and implementation.

2. Overall Description

2.1 Product perspective

Context:

This product was introduced to provide ease to the end users who had to first call the restaurants and then place order and the restaurants use to charge a delivery fee as well for that feasibility.

Furthermore, the restaurants had to hire delivery staff that used to be an added cost on their tab. By the introduction of this product many restaurants have been able to keep the cost of hiring down. The platform creates job opportunities for delivery riders and ease of maintenance to the end users and restaurants.

The food delivery management system that we made is a follow on and improvement of the food panda app that is already on its heights. Providing the users with added choices as the cancellation of order options.

2.2 Product Functions

Below is the definition of the functions that this management system will provide:

Place Order:

The customer logs in into the system and selects their choice of restaurant, after which a menu is displayed to them from which they select the items they want and add those items to their cart, after this they proceed to payment and then place the order.

Cancel Order:

The customer has ordered and wants to cancel that order for some reason, they select the order they want to cancel and are displayed with a small form from which they select the reason why they want to cancel that order and then submit the form after which the order is deleted.

Manage Rider:

This function is for the admin where he can authorize new riders that want to work for the system, only authorized riders can be allowed to deliver the orders.

Manage Restaurant:

This function is for the admin where he can authorize new restaurants that want to register themselves on the system, only authorized restaurants can be allowed to receive orders from customers.

Register Users:

This function is for different users like rider, restaurant, and customer. In this function they will sign up to the system making a new account and then becoming part of the system.

Accept Order:

This function is for the rider, when the customer has ordered for that specific restaurant, the restaurant gets to accept the order and start preparing the customer's order. When the order is prepared the riders are notified and they can select what order to deliver seeing whichever restaurant is close to them.

Add Food Item:

In this function the restaurant can add some new food items to their existing menu.

Delete Food Items:

In this function the restaurant can delete some food items from their menu.

Modify Food Items:

In this function the restaurant can update their menu, changing the price or any other entity of their menu.

Processing Order:

This function is for the restaurants, when the customer has ordered for that specific restaurant, the restaurant gets to accept the order and start preparing the customer's order which will then be delivered.

2.3 List of Use Cases

1. Place Order
2. Cancel Order
3. Manage Rider
4. Manage Restaurant
5. Register Users (Customer, Restaurant, Rider)
6. Accept Order
7. Add Food Item
8. Delete Food Item
9. Modify Food Item
10. Process Order

2.4 Extended Use Cases

UC01

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Place Order

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Customer

Stakeholders and Interests:

The customer wants to easily view all the restaurants and search by the food item that he wants, and the system should display the food items of his liking. He also wants that whatever he orders has a smooth process up till the delivery.

The restaurant owner will be preparing the order and want a fast system using which it can manage the orders.

The rider wants to have the completed orders so that they can start making the delivery.

Pre-conditions:

- 1) The customer should be registered and logged into the system
- 2) The restaurant he wants to order from should also be registered on the system

Post-conditions: The order is placed, the system will be updated, the customer will be given a waiting time, the restaurant will start to prepare the food.

Main success scenario:

Actor Action	System Response
1) The use case begins when the customer chooses a restaurant.	2) The system will display the menu of selected Restaurant.
3) The Customer Selects a dish from the menu of the Restaurant.	4) System will add the dish in the cart.
5) The customer selects the checkout option	6) System will Generate the Bill accordingly.

7) Customer proceeds to payment and enters the card details.	8) System will verify the payment credentials and will notify the customer.
9) The customer then selects Place order option, and the order is placed.	10) The system updates the order status and gives the customer a specific waiting time.

Extensions:

***A. At any time, the system fails:**

1. The customer quits the current session
2. The customer logs in again into the system.
3. If any items were added in the cart they remain there
4. The customer can begin from where the system crashed.

1a. The restaurant the customer wants to order from is not registered on the system:

1. The customer is given suggestions of restaurants that have the similar menu as the customer wanted restaurant.
2. The customer can then select from that list any restaurant and order from it.

1b. Customer cannot login into their account:

1. The customer is taken to an account recovery step
2. The customer enters the details like their email id
3. The system verifies the information
4. The account is retrieved.

4a. The cart already has items from another restaurant:

1. The system generates an error message asking the customer if they want to remove the other items.
2. The customer selects the option to remove the items.
- 2b. The customer selects to not remove the old items

1. The customer is then taken to the checkout option
3. The process of placing the order is then continued

8a. The customer wants to do payment by cash on delivery method:

1. The system updates the order with a COD on it.

2. The system prompts the restaurant the customer ordered from that the customer will be paying by cash.
3. The rider that picks up the order is notified that the customer will be paying by cash.

8b. The payment details are not correct:

1. The system prompts an error message
2. The customer is asked to enter the details again
3. If the error persists, they are given the option of doing payment by cash.

UC02

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Cancel Order

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Customer

Stakeholders and Interests:

The customer wants to have an efficient system that removes the already made order by the customer due to any reason.

The restaurant owner wants a system through which they can keep track of the cancelled orders and can make things work efficiently for their kitchen.

Pre-conditions:

- 1) The customer should be registered onto the system
- 2) The customer should have ordered from a restaurant.
- 3) The restaurant has not yet dispatched the order.

Post-conditions: The order is cancelled, the system updates the order status, if the customer ordered via card payment the amount is refunded.

Main success scenario:

Actor Action	System Response
1) The use case begins when the customer has ordered from a restaurant and wants to cancel the order. 2) The customer opens the order menu and selects the cancel order option. 4) The customer fills the form and submits it. 7) The customer selects the option best suited to them.	3) System will display the customer with a small form asking the reason for the cancellation. 5) The system approves the request if the order has not yet been delivered or dispatched. 6) The system then shows an option to the user to receive a voucher for the amount or to receive the amount back in bank. 8) The system returns the amount in whatever method the customer selected.

Extensions:

***A. At any time, the system fails:**

1. The customer quits the current session
2. The customer logs in again into the system.
3. The system starts from where the cancellation process was left.
4. The customer can then continue with the rest of the process.

5a. The order has been picked up by the rider:

1. The system gives the customer two options either take the order and make the payment or give a small fine fee for the order and rider cost.
2. The customer selects the option suited to them.
 - a. The customer accepts the order.
 - b. The customer gives the small fee and cancels the order.

UC03

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Manage Rider

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Admin

Stakeholders and Interests:

The admin wants a system through which he can easily and quickly verify the newly registered riders, delete riders that have resigned or update their records.

The restaurant owner wants a system that gives them the updated records of the riders so that they can request the riders for their orders accordingly.

The riders want to be verified so that they can accept new orders.

Pre-conditions:

- 1) The rider should be registered onto the system
- 2) The admin should be logged into the system.
- 3) The admin should have the rider details.

Post-conditions: The rider is verified and now can deliver orders from different restaurants. The admin has updated the system regarding the newly verified rider.

Main success scenario:

Actor Action	System Response
1) The use case begins when the admin logs into the system. 3) The admin receives the request from registered riders to get verified.	2) System will display the home screen and displays options.

<p>5) The admin chooses the option to authorize the rider.</p> <p>7) The admin picks a rider which they want to verify.</p> <p>9) The admin sees if the rider meets the requirements and then clicks confirm to authorize the rider.</p>	<p>4) The system notifies the admin and gives option to the admin to verify the new riders.</p> <p>6) The system will then show a detailed list of all the riders that want to be verified.</p> <p>8) The details of that rider are displayed to the admin.</p> <p>10) The system authorizes the rider and updates in the database records with the rider details and status as verified.</p>
--	---

Extensions:

***A. The system crashes at any point:**

1. The admin quits the current session.
2. The admin restarts the system.
3. The admin then continues from where the system crashed.

1a. The admin enters invalid account details:

1. The admin is prompted by the system that they entered invalid details.
2. The admin is asked to enter the details again
3. If the admin still cannot enter the correct details the admin is asked to verify the email address
4. The password of the account is then sent on that email address
5. The account is retrieved.

2a. The admin wants to delete the rider account:

1. Admin selects the option to delete rider account from the options shown by the system.
2. The system then asks the admin to enter the details of the rider that they want to remove.
3. The admin enters the details of the rider and the reason for deletion of account of that rider.
4. The admin then clicks delete and deletes that rider account.

5. The system checks the account information entered by the admin in the database.
6. The system then removes the deleted rider from the database records.

2b. The admin wants to modify the rider information:

1. The admin selects the option of updating information from the options shown to him.
2. The system then asks the admin to enter the details of the rider that they want to update.
3. The admin enters the details of the rider and the information they want to update.
4. The admin then clicks update.
5. The system checks the account information entered by the admin in the database.
6. The system updates the database records with the newly updated information of the rider.

6a. There are no riders that are registered and need verification:

1. The admin can select other options from step 2.

9a. The rider does not fulfill the requirements for verification:

1. The admin generates a per-written email that tells the rider that they are not fulfilling the requirements for verification.
2. The system sends the email to the rider email address.
3. The system then removes the request of that rider from the requests list
4. The system then again executes step 2.

UC04

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Manage Restaurant

Scope: Food Delivery Management

System Level: User Goal

Primary Actor: Admin

Stakeholders and Interests:

The admin wants a system through which he can easily and quickly verify the newly registered restaurants, delete restaurants that have completed the contract or update their records.

The restaurant owner wants a system that can easily and quickly guide them through the verification process and make it easier for them to start working with the system.

The riders want an updated list of restaurants that they can choose from to deliver their orders.

The customers want to have an updated list of restaurants that they can select from and order from.

Pre-conditions:

- 1) The restaurant should be registered onto the system
- 2) The admin should be logged into the system.
- 3) The admin should have the restaurant details.

Post-conditions: The restaurant is verified and now can take orders from customers. The admin has updated the system regarding the newly verified restaurant. The customers can now see the new restaurant on their system and can order from there. The riders are notified about any orders being sent from this restaurant.

Main Success Scenario:

Actor Action	System Response
1) The use case begins when the admin logs into the system.	
3) The admin receives the request from registered restaurants to get verified.	2) System will display the home screen and displays options. 4) The system notifies the admin and gives option to the admin to verify the new restaurants.
5) The admin chooses the option to authorize the restaurants.	6) The system will then show a detailed list of all the restaurants that want to be verified.
7) The admin picks a restaurant which they want to verify.	

9) The admin sees if the restaurant meets the requirements and then clicks confirm to authorize the restaurant.	8) The details of that restaurant are displayed to the admin. 10) The system authorizes the restaurant and updates in the database records with the restaurant details and status as verified.
---	---

Extensions:

***A. The system crashes at any point:**

1. The admin quits the current session.
2. The admin restarts the system.
3. The admin then continues from where the system crashed.

1a. The admin enters invalid account details:

1. The admin is prompted by the system that they entered invalid details.
2. The admin is asked to enter the details again
3. If the admin still cannot enter the correct details the admin is asked to verify the email address
4. The password of the account is then sent on that email address
5. The account is retrieved.

2a. The admin wants to delete the restaurant account:

1. Admin selects the option to delete restaurant account from the options shown by the system.
2. The system then asks the admin to enter the details of the restaurant that they want to remove.
3. The admin enters the details of the restaurant and the reason for deletion of account of that restaurant.
4. The admin then clicks delete and deletes that rider account.
5. The system checks the account information entered by the admin in the database.
6. The system then removes the deleted restaurant from the database records.

2b. The admin wants to modify/update the restaurant information:

1. The admin selects the option of updating information from the options shown to him.

2. The system then asks the admin to enter the details of the restaurant that they want to update.
3. The admin enters the details of the restaurant and the information they want to update.
4. The admin then clicks update.
5. The system checks the account information entered by the admin in the database.
6. The system updates the database records with the newly updated information of the restaurant.

6a. There are no restaurants that are registered and need verification:

1. The admin can select other options from step 2.

9a. The restaurant does not fulfill the requirements for verification:

1. The admin generates a per-written email that tells the restaurant that they are not fulfilling the requirements for verification.
2. The system sends the email to the restaurant email address.
3. The system then removes the request of that restaurant from the requests list.
4. The system then again executes step 2.

UC05

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Customer

Stakeholders and Interests:

The customers will want to register quickly, so that they can order the food.

Pre-Conditions: The customer should have a phone number for OTP verification.

Post-Conditions: The customer will be able to login their account in the system and perform the actions of placing orders.

Main Success Scenario:

Actor Action	System Response
1) The use case begins when an unregistered user wants to register as a customer. The customer enters the 'sign up' button.	
3) The customer fills the required fields in the form.	2) System directs the customer to a page, where the customer is promoted to fill in a form.
4) The user clicks the 'sign up' button.	
6) The customer enters the correct code.	5) The system notifies the customer that an OTP code has been sent to their mobile number. The system prompts the customer to enter the code.
9) The customer can now login.	7) On correct code, the system shows a message indicating that the user has been registered. 8) The system takes the customer to the login page

Extensions:

***A. At any time, the system fails:**

1. The customer quits the current session
2. The customer reloads the system
3. The system starts from where the registration process was left

***B. At any time, the customer discards the sign up and goes back:**

1. The customer is provided with a cancel sign up option.
2. If clicked, the registration is discarded, and the system takes the customer back.

4a. The Customer does not fill all the required fields:

1. After the customer clicks the sign-up button, the system checks if all the required fields are filled.

2. If not, the system reloads the same page, prompting the user to enter all the required fields of the form. The system also informs which field was empty.
3. The Customer fills the form again.
4. This scenario repeats until the customer fills all the required fields of the form.

4b. The customer incorrectly fills certain fields:

1. The system checks all the fields, to see if all of them are correctly entered,
2. If not, the system displays an error message, along with the incorrect fields entered.
3. The customer re-enters those fields.
4. This is repeated until all the fields are correctly entered.

5a. The customer does not receive the OTP code after clicking sign-up:

1. After the system prompts user to enter the OTP code, the system displays an option to re send the code if not received.
2. Customer clicks on this option, until they receive the OTP code.

6a. The Customer enters wrong code:

1. The system informs the customer that they entered wrong OTP code and asks to enter the code again.
2. The Customer re-enters the code.
3. This repeats until the customer enters the correct code.

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Rider

Stakeholders and Interests:

The Rider will want to register quickly, so that they can deliver the food, and do their jobs and earn money.

The restaurant owners will want to have more and more riders available for the increase in number of orders from customers.

The customers will want to have their food delivered quickly, which will be fulfilled by having more registered riders.

Pre-Conditions: The rider should have a phone number for OTP verification. The rider should have a bike driving license.

Post-Conditions: The rider will be able to login their account in the system. To perform deliveries, the system will have to prompt the admin to verify the driver. This is done by the 'manage riders' use case.

Main Success Scenario:

Actor Action	System Response
1) The use case begins when an unregistered user wants to register as a rider. The rider enters the 'sign up' button.	
3) The rider fills the required fields in the form.	2) System directs the rider to a page, where the rider is prompted to fill in a form.
4) The user clicks the 'sign up' button.	
6) The rider enters the correct code.	5) The system notifies the rider that an OTP code has been sent to their mobile number. The system prompts the rider to enter the code.
	7) On correct code, the system shows a message indicating that the rider has been registered, as an unverified rider.
	8) The system takes the rider to the login page
9) The rider can now login.	

Extensions:

*A. At any time, the system fails:

4. The rider quits the current session
5. The rider reloads the system
6. The system starts from where the registration process was left

***B. At any time, the rider discards the sign up and goes back:**

3. The rider is provided with a cancel sign up option.
4. If clicked, the registration is discarded, and the system takes the rider back.

4a. The rider does not fill all the required fields:

1. After the rider clicks the sign-up button, the system checks if all the required fields are filled.
2. If not, the system reloads the same page, prompting the user to enter all the required fields of the form. The system also informs which field was empty.
3. The rider fills the form again.
4. This scenario repeats until the rider fills all the required fields of the form.

4b. The rider incorrectly fills certain fields:

1. The system checks all the fields, to see if all of them are correctly entered,
2. If not, the system displays an error message, along with the incorrect fields entered.
3. The rider re-enters those fields.
4. This is repeated until all the fields are correctly entered.

5a. The rider does not receive the OTP code after clicking sign-up:

1. After the system prompts user to enter the OTP code, the system displays an option to re send the code if not received.
2. rider clicks on this option, until they receive the OTP code.

6a. The rider enters wrong code:

1. The system informs the rider that they entered wrong OTP code, and asks to enter the code again.
2. The rider re-enters the code.
3. This repeats until the rider enters the correct code.

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Restaurant

Stakeholders and Interests:

The Restaurant owner will want to register quickly, so that they can acquire the food delivery services and get more customers.

The customers will want to have more restaurant options, which will be fulfilled by having more registered restaurants.

Pre-Conditions: The restaurant should have a phone number for OTP verification. The restaurant should have necessary documents.

Post-Conditions: The restaurant will be able to login their account in the system. To allow customers to access their restaurant through the food delivery system, the system will have to prompt the admin to verify the restaurant. This is done by the 'Manage restaurant' use case.

Main Success Scenario:

Actor Action	System Response
1) The use case begins when an unregistered user wants to register as a restaurant. The restaurant enters the 'sign up' button.	
3) The restaurant fills the required fields in the form.	2) System directs the restaurant to a page, where the restaurant is promoted to fill in a form.
4) The user clicks the 'sign up' button.	
6) The restaurant enters the correct code.	5) The system notifies the restaurant that an OTP code has been sent to their mobile number. The system prompts the restaurant to enter the code.

9) The restaurant can now login.	7) On correct code, the system shows a message indicating that the restaurant has been registered, as an unverified restaurant. 8) The system takes the restaurant to the login page
----------------------------------	---

Extensions:

***A. At any time, the system fails:**

7. The restaurant quits the current session
8. The restaurant reloads the system
9. The system starts from where the registration process was left

***B. At any time, the restaurant discards the sign up and goes back:**

5. The restaurant is provided with a cancel sign up option.
6. If clicked, the registration is discarded, and the system takes the restaurant back.

4a. The restaurant does not fill all the required fields:

1. After the restaurant clicks the sign-up button, the system checks if all the required fields are filled.
2. If not, the system reloads the same page, prompting the user to enter all the required fields of the form. The system also informs which field was empty.
3. The restaurant fills the form again.
4. This scenario repeats until the restaurant fills all the required fields of the form.

4b. The restaurant incorrectly fills certain fields:

1. The system checks all the fields, to see if all of them are correctly entered,
2. If not, the system displays an error message, along with the incorrect fields entered.
3. The restaurant re-enters those fields.
4. This is repeated until all the fields are correctly entered.

5a. The restaurant does not receive the OTP code after clicking sign-up:

1. After the system prompts user to enter the OTP code, the system displays an option to re send the code if not received.
2. Restaurant clicks on this option, until they receive the OTP code.

6b. The restaurant enters wrong code:

1. The system informs the restaurant that they entered wrong OTP code and asks to enter the code again.
2. The restaurant re-enters the code.
3. This repeats until the rider enters the correct code.

UC06

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Accept Order

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Rider

Stakeholders and Interests:

The rider wants a system from which they can easily accept orders that are nearest to their location.

The restaurant owner wants to keep prepared orders in their restaurant to the minimum and hence want a system that is fast and efficient and ensures that their orders are delivered as soon as possible.

The customer wants a fast delivery system that ensures that the ordered items are delivered to them quickly.

Pre-conditions:

- 1) The rider should be registered onto the system
- 2) The customer should have ordered something, and the restaurant must have prepared it.

- 3) The restaurant must have put out a rider request.
- 4) The rider must be in proximity of the delivery address.

Post-conditions: The rider picks up the order from the restaurant and delivers to the designated address, the system updates the order status with completed.

Main success scenario:

Actor Action	System Response
1) The use case begins when the restaurant has prepared the order and sends a request out for a rider.	2) System will display the rider the restaurants near the rider's location and the addresses where the orders must be delivered.
3) The rider picks up any order most suited to them.	4) The system notifies the restaurant about the rider coming to pick up that order.
5) The rider reaches the restaurant and picks the order. 6) The rider then updates that he has picked the order.	7) The system then updates the order information to picked by rider and gives the rider the customer information. It also shows the customer that their order has been picked.
8) The rider reaches the specified address and calls the customer and delivers the order.	
9) the rider then updates the order as delivered.	10) The system updates and marks the order as completed.

Extensions:

***A. The system crashes at any point:**

1. The rider quits the current session.
2. The rider starts the system again

3. The rider then selects from the orders that are being shown.
4. The rider then goes to the restaurant and picks up the order to be delivered.
5. The rider delivers the order at the specified address.

1a. There are no orders to be delivered around the rider:

1. The rider checks again and refreshes to see if any orders appear on the system.
2. The rider then moves to a different sector and then refreshes their app.
3. The rider then picks order from this new location and then delivers to the location most suited to him.

8a. The restaurant has prepared the wrong order:

1. The rider calls the restaurant and asks them to confirm the order.
2. The restaurant then confirms the order and tells them they had made a mistake.
3. The rider then apologizes and goes to the restaurant again.
4. The rider picks the right order from the restaurant.
5. The system is updated regarding the issue and the rider is compensated for the extra trip.

UC07

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Add food item

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Restaurant

Stakeholders and Interests:

The Restaurant owner will want to add items to their restaurant menu, so that the customers can see their restaurants latest food items.

The customers will want to have more food options, which will be fulfilled by adding food items.

Pre-Conditions: The restaurant should be logged into their account.

Post-Conditions: The restaurant menu will now have more food options.

Main Success Scenario:

Actor Action	System Response
<p>1) The use case begins when a restaurant chooses the 'add food item' option, to add food items to their menu.</p> <p>3) The restaurant fills all these fields.</p> <p>4) After correctly filling all the fields, the restaurant clicks the 'add' button.</p>	<p>2) System directs the user to a page, where there is a form with fields: food item name, food item ingredients and food item price.</p> <p>5) The system checks if all the fields are filled, with the accepted inputs.</p> <p>6) On correct inputs, the system adds the food item to the restaurant menu and the system database.</p> <p>7) The system displays a message indicating that the food item has been successfully added.</p>

Extensions:

***A. At any time, the system fails:**

1. The restaurant quits the current session
2. The restaurant logs in again into the system
3. The system starts from where the add item process was left

***B. The restaurant can stay on the same page to add another food item:**

1. After one food item is added, the system reloads the 'add food item' page again.
2. Steps from step 2 onwards are repeated.
3. This repeats, until the restaurant wishes to go back, for which it is provided with the 'go back' option

***C. At any time, the restaurant discards the 'add food item' and goes back:**

1. The restaurant is provided with a cancel action option.
2. If clicked, the action is discarded, and the system takes the restaurant back.

5a. The restaurant does not fill all the required fields:

1. After the restaurant clicks the 'add' button, the system checks if all the required fields are filled.

2. If not, the system reloads the same page, prompting the user to enter all the required fields of the form. The system also informs which field was empty.
3. The restaurant fills the form again.
4. This scenario repeats until the restaurant fills all the required fields of the form.

5b. The restaurant incorrectly fills certain fields:

1. The system checks all the fields, to see if all of them are correctly entered,
2. If not, the system displays an error message, along with the incorrect fields entered.
3. The restaurant re-enters those fields.
4. This is repeated until all the fields are correctly entered.

UC08

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Delete food item

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Restaurant

Stakeholders and Interests:

The Restaurant owner will want to delete items in their restaurant menu, so that their menu is up to date.

Pre-Conditions: The restaurant should be logged into their account.

Post-Conditions: The selected food item will be removed from the restaurant menu.

Main Success Scenario:

Actor Action	System Response
<p>1) The use case begins when a restaurant chooses the 'delete food item' option, to delete food items from their menu.</p> <p>3) The restaurant selects item or items from this list.</p> <p>4) After selecting food item or items, the restaurant clicks on the 'delete' option.</p>	<p>2) System directs the user to a page, where there is the complete list of food items for that restaurant.</p> <p>5) The system checks if at least one item is selected.</p> <p>6) The system removes the selected item or items from the restaurant menu and the system database.</p> <p>7) The system displays a message indicating that the selected food items have been successfully deleted.</p>

Extensions:

***A. At any time, the system fails:**

1. The restaurant quits the current session
2. The restaurant logs in again into the system
3. The system starts from where the delete item process was left

***B. The restaurant can stay on the same page to delete another food item:**

1. After one food item is deleted, the system reloads the 'delete food item' page again.
2. Steps from step 2 onwards are repeated.
3. This repeats, until the restaurant wishes to go back, for which it is provided with the 'go back' option

***C. At any time, the restaurant discards the ‘delete food item’ and goes back:**

1. The restaurant is provided with a cancel action option.
2. If clicked, the action is discarded, and the system takes the restaurant back.

5a. The restaurant does not select any food item and clicks on the delete option:

1. The system displays an error message and prompts the restaurant to select at least one food item.
2. This repeats until the restaurant selects one or more items.

UC09

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Modify food item

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Restaurant

Stakeholders and Interests:

The Restaurant owner will want to modify food items in their restaurant menu, so that their menu is up to date.

The customers will want to have up to date information on each food item.

Pre-Conditions:

The restaurant should be logged into their account.

Some food items must exist on the restaurant menu.

Post-Conditions: The modified food item will be updated on the restaurant menu.

Main Success Scenario:

Actor Action	System Response
1) The use case begins when a restaurant chooses the ‘modify food item’ option, to delete food items from their menu.	

<p>3) The restaurant selects an item from this list.</p> <p>5) The restaurant changes the fields where they want to make the changes for.</p> <p>6) After updating the required fields, the restaurant clicks on the 'modify details' option.</p>	<p>2) System directs the user to a page, where there is the complete list of food items for that restaurant.</p> <p>4) The system takes the restaurant to a page, much like the 'add food item' page, where the selected food item details are already entered in the fields.</p> <p>6) The system responds by removing the old item and replacing it with the food item currently modified, with the modified details. Thus, modifying the item in the restaurant menu and the system database.</p> <p>7) The system displays a message indicating that the selected food items have been successfully modified.</p>
---	---

Extensions:

***A. At any time, the system fails:**

4. The restaurant quits the current session
5. The restaurant logs in again into the system
6. The system starts from where the modify item process was left

***B. The restaurant can stay on the same page to modify another food item:**

4. After one modification, the system reloads the 'modify food item' page again.
5. Steps from step 2 onwards are repeated.
6. This repeats, until the restaurant wishes to go back, for which it is provided with the 'go back' option

***C. At any time, the restaurant discards the 'modify food item' and goes back:**

3. The restaurant is provided with a cancel action option.

4. If clicked, the action is discarded, and the system takes the restaurant back.

2a. There are no food items in the menu

1. The system displays an error message indicating the restaurant that there are no food items to modify.
2. The system takes the user back.

UC10

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Processing Order

Scope: Food Delivery Management System

Level: User Goal

Primary Actor: Restaurants

Stakeholders and Interests:

The restaurant owner wants to easily access all the orders that are coming their way, and quickly accept them so that the restaurant start preparing the orders.

The customer wants fast delivery, which can be made sure by correctly and quickly processing the orders, without any delay.

The riders want to have the completed orders so that they can start making the delivery.

Pre-Conditions: The restaurant must be registered into the system and logged in.

Post-Conditions: The order is accepted, order status is updated, order can now be delivered by the rider.

Main Success Scenario:

Actor Action	System Response
1) The Restaurant receives an order, whenever a customer place orders. The use case begins when the Restaurant selects the 'process order' option.	2) System will display the orders list, which have not been processed yet. The list is in the order of

3) Now the restaurant accepts the first order in the list.	orders received in the system by the customers, at first come basis.
5) The Kitchen now makes the order, and when it's done, the kitchen changes the order status to 'order prepared'.	4) When the order has been accepted, the system changes the order status to 'order being prepared'. At this point, the order is sent to the kitchen.
	6) The system notifies all the available riders, to pick the prepared order.

Extensions:

***A. At any time, the system fails:**

1. The restaurant quits the current session
2. The restaurant logs in again into the system
3. The system starts from where the process order was left

***B. At any time, the restaurant discards the 'add food item' and goes back:**

1. The restaurant is provided with a cancel action option.
2. If clicked, the action is discarded, and the system takes the restaurant back.

3a. The restaurant denies a particular order:

1. The restaurant has the option to deny orders (because of any reason), which is done by clicking the 'deny order' option.
2. The system prompts to write the reason.
3. The restaurant writes the reason for cancelling order and select 'send'.
4. The system sends this error message to the customer, along with the reason.

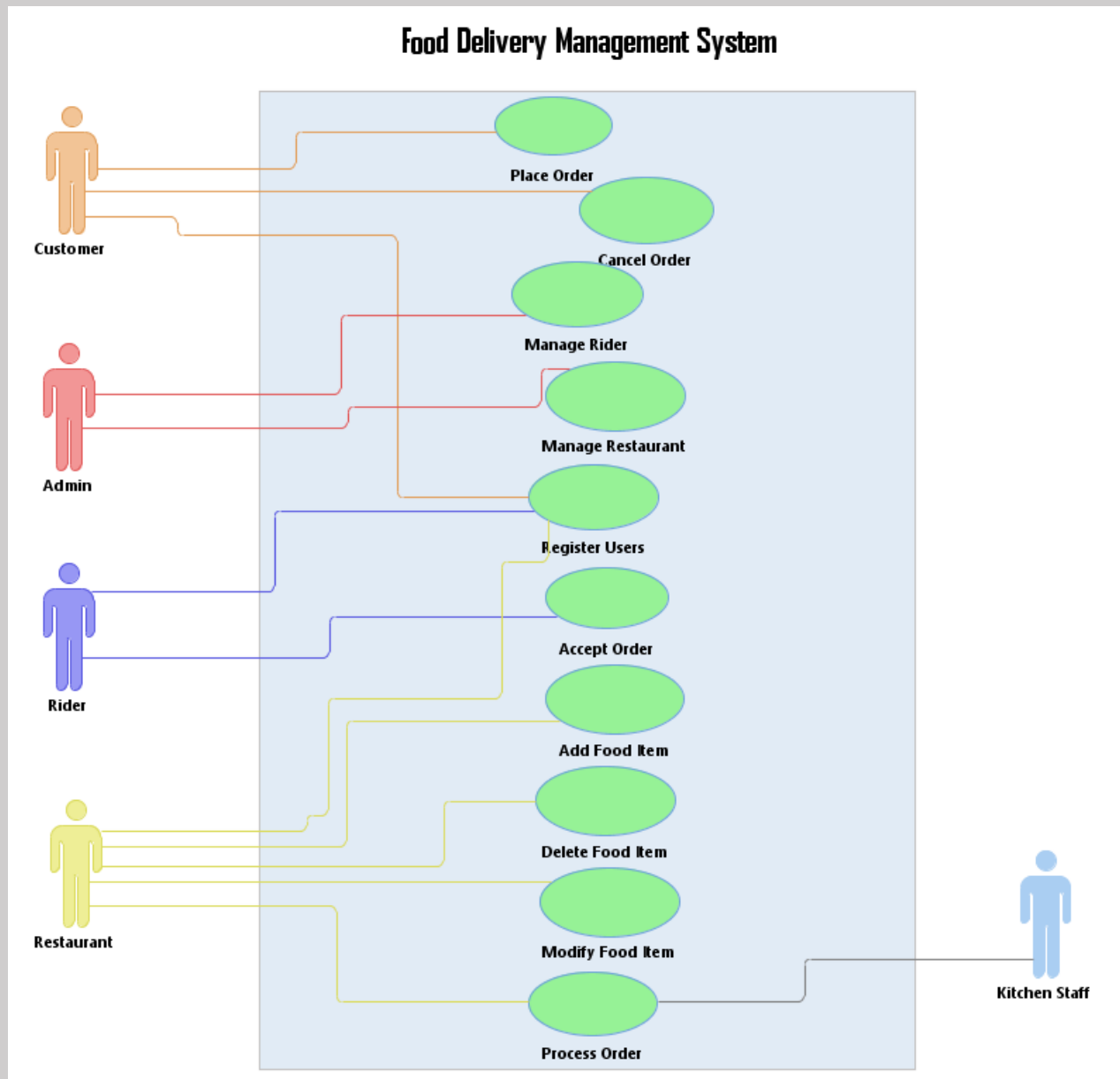
2a. All the orders for the restaurant have been processed:

1. The system will respond by displaying a message indicating that there are no orders to process.
2. The restaurant can now go back.

6a. There are no free riders to notify to pick up the prepared order:

1. The system informs the restaurant that there are no free riders now
2. The system checks again after 5 minutes
3. The system repeats this until there is a free rider available

2.5 Use Case Diagram:



3. Other Non-Functional Requirements

3.1 Performance Requirements

The requirement for this product is that the users know and remember their login credentials and log into the system via them.

The system does not have any requirements regarding the devices of use and can work on all devices.

The average time to login till the customer places the order is 3mins to 6mins. Orders are shown on the restaurant screen as soon as they are placed.

3.2 Safety Requirements

The product is safe to use for users of all ages, having no risk factor.

3.3 Security Requirements

We hereby attest to the fact that our system will not violate anyone's personal information. Any information shared on the platform will remain strictly confidential. The users are advised to not share their credentials with others, as this may cause a breach of their privacy, in which case we will not be responsible. If such a case occurs this must be reported to us immediately, to prevent further mishap.

3.4 Software Quality Attributes

Our system is very easy and efficient to use. It is adaptable for various platforms. The interface navigation is easy and attractive for the users. The design patterns behind this system make it easily maintainable and reusable.

3.5 Business Rules

There are separate accounts for the different users on this system that are: admin, customer, restaurants and riders. All the users have different functions that they can perform on the system.

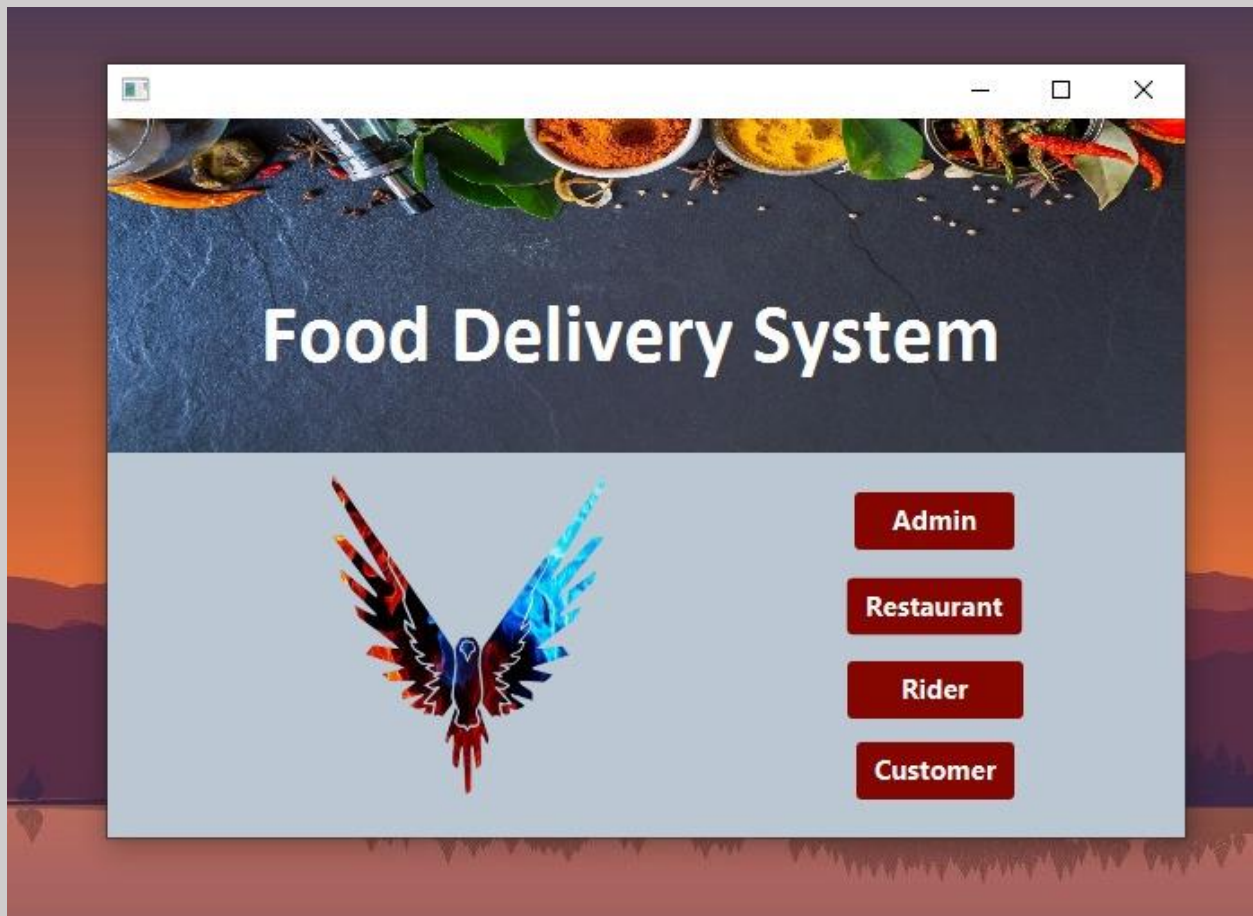
3.6 Operating Environment

This system runs on any device that supports java language and has scene builder application installed on it. IDE's such as eclipse or intellij are best to operate this product.

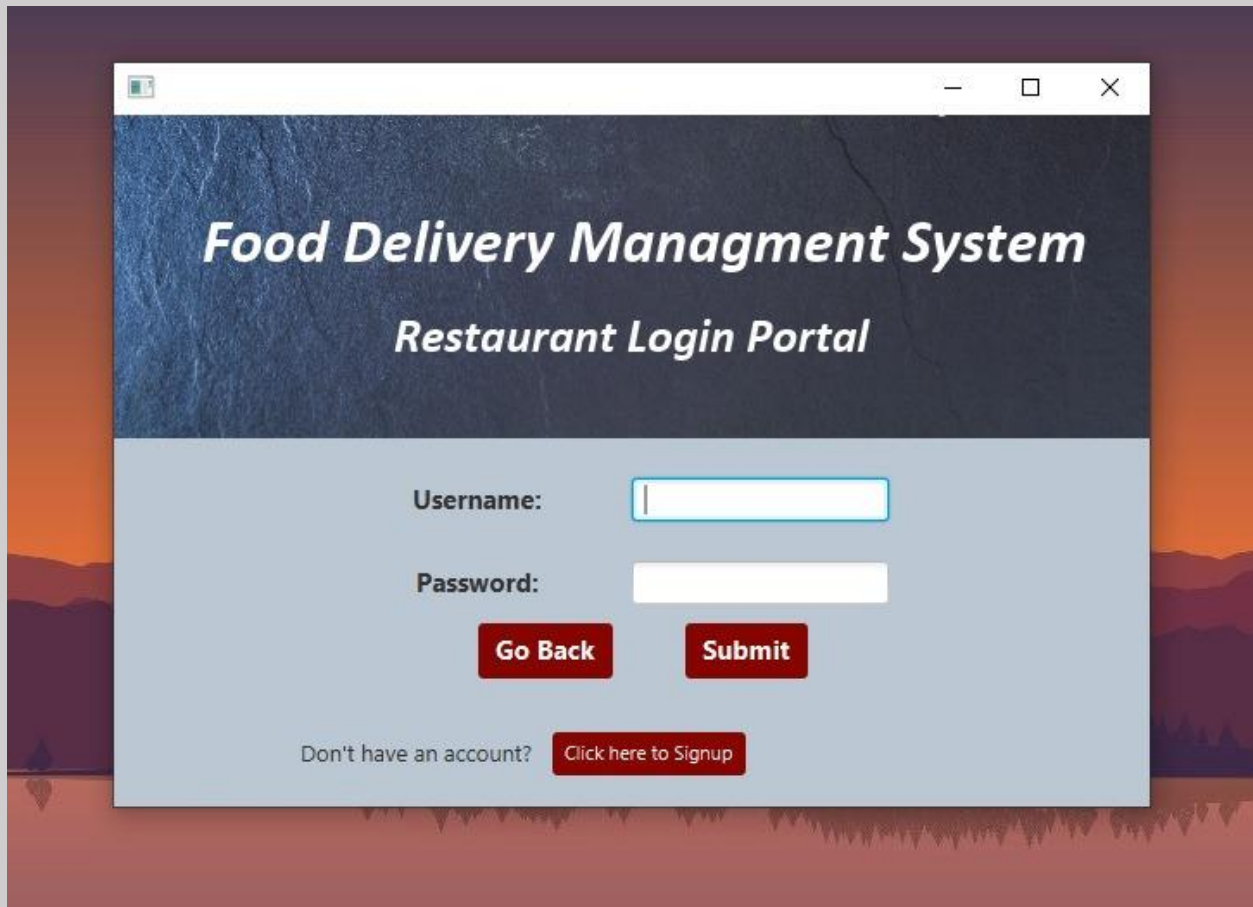
3.7 User Interfaces

This system is very easy to use. There are clear instructions on every page and the buttons and pages can be easily inferred by the user as to what their functionality is. Below are some snippets attached to show how the above point is proven:

The main page is easy to access and navigate through having different buttons for the different users that there are in the system. Below is the snippet of the page:

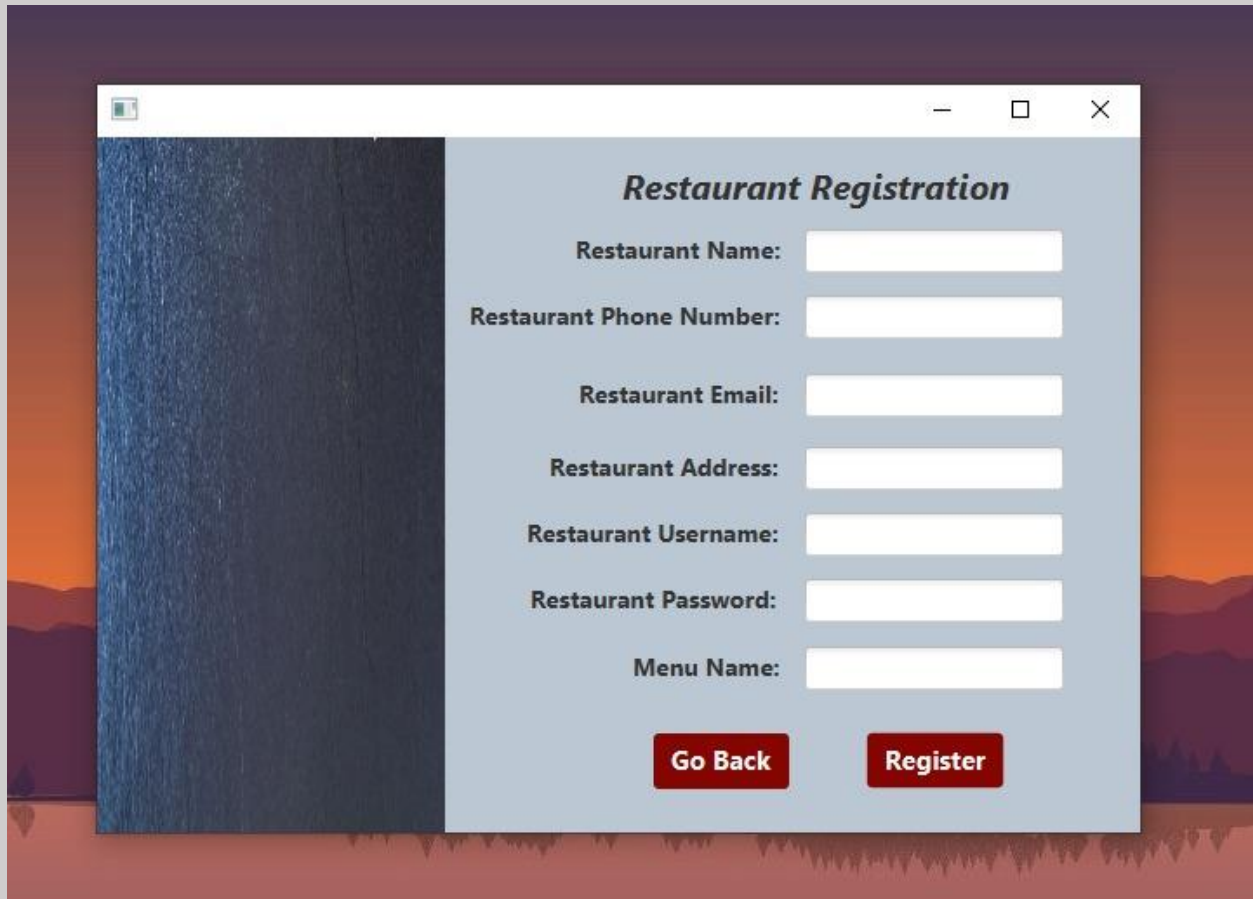


After selecting any of the option the user is asked to log into their account and a login page is displayed. The login page has the option to go back to make it easy for the user if they select the wrong option they can go back and select the write one. Below is the snippet for this page:



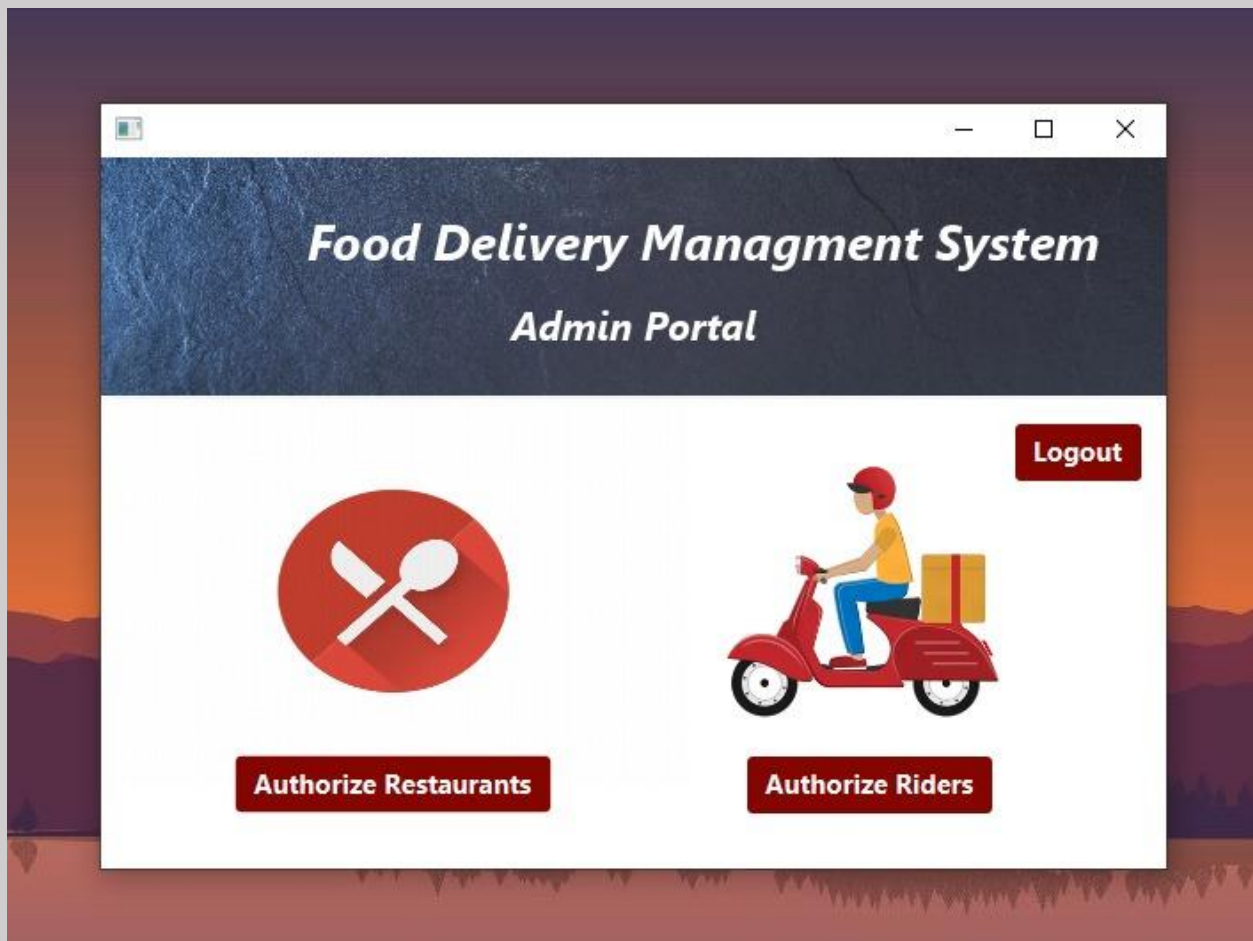
The login page also shows the signup button for those users that are not currently in their system and want to join, it is also shown that there are clear indications as to where to go to do what operation.

Upon selecting the signup button the user is showed a form that they must fill and then submit, this registers the user to the system and makes their new account on the system.

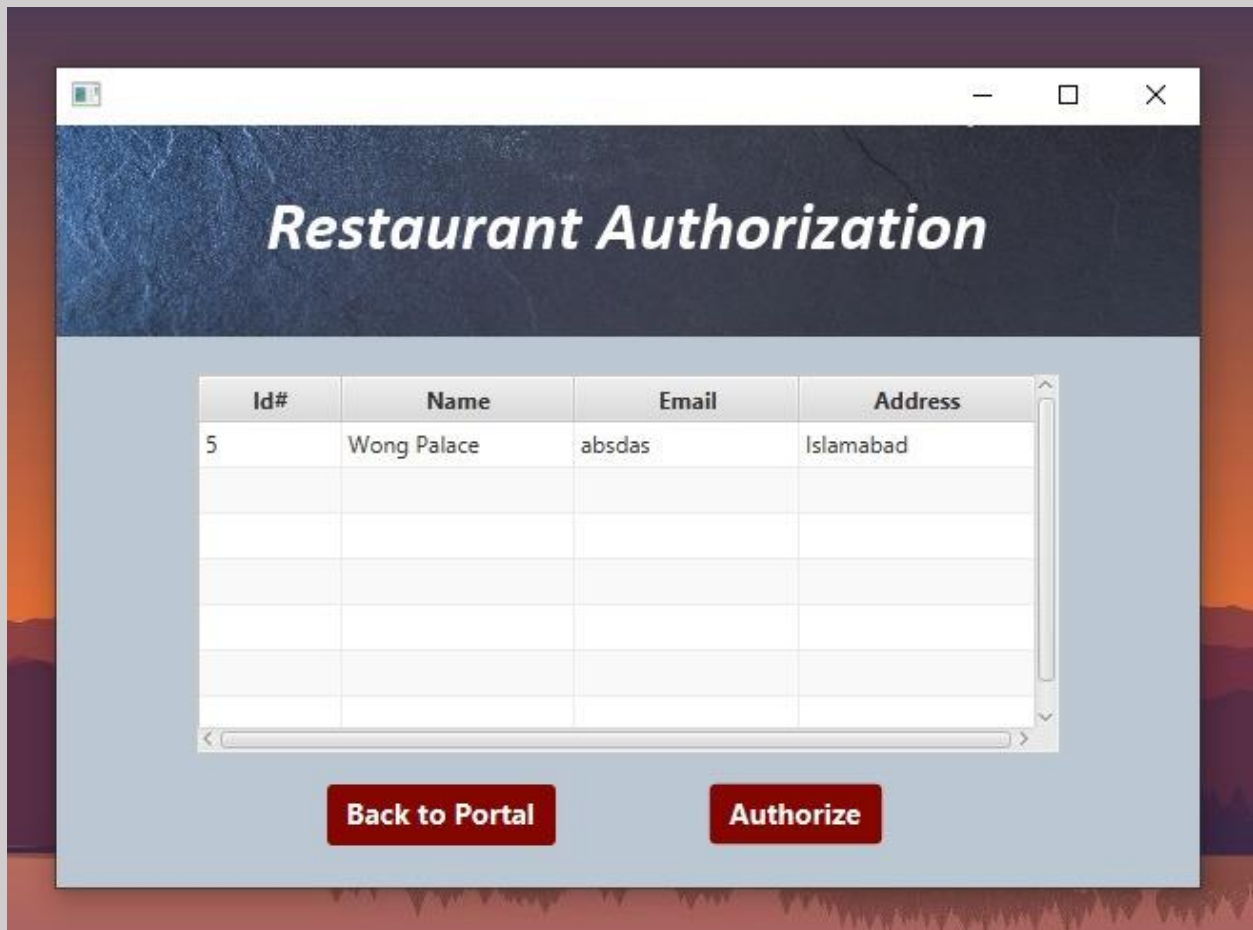
A screenshot of a web browser window displaying a 'Restaurant Registration' form. The form is set against a light blue background with a dark blue textured sidebar on the left. The title 'Restaurant Registration' is centered at the top in a bold, italicized font. Below the title, there are seven input fields, each preceded by a label: 'Restaurant Name:', 'Restaurant Phone Number:', 'Restaurant Email:', 'Restaurant Address:', 'Restaurant Username:', 'Restaurant Password:', and 'Menu Name:'. At the bottom of the form, there are two red buttons with white text: 'Go Back' and 'Register'. The browser window has a standard title bar with minimize, maximize, and close buttons.

Upon logging in to the system the user portal is displayed to the user where they are shown all the operations that they can perform.

Below is the portal for the user admin and it shows all the operations that they can perform using their account. The portal page also clearly indicates the operations with clear text on the buttons, there is also a logout button on all the portal pages to allow the user to logout of the system any time they want. The interface is made more attractive by placing related images and gifs.

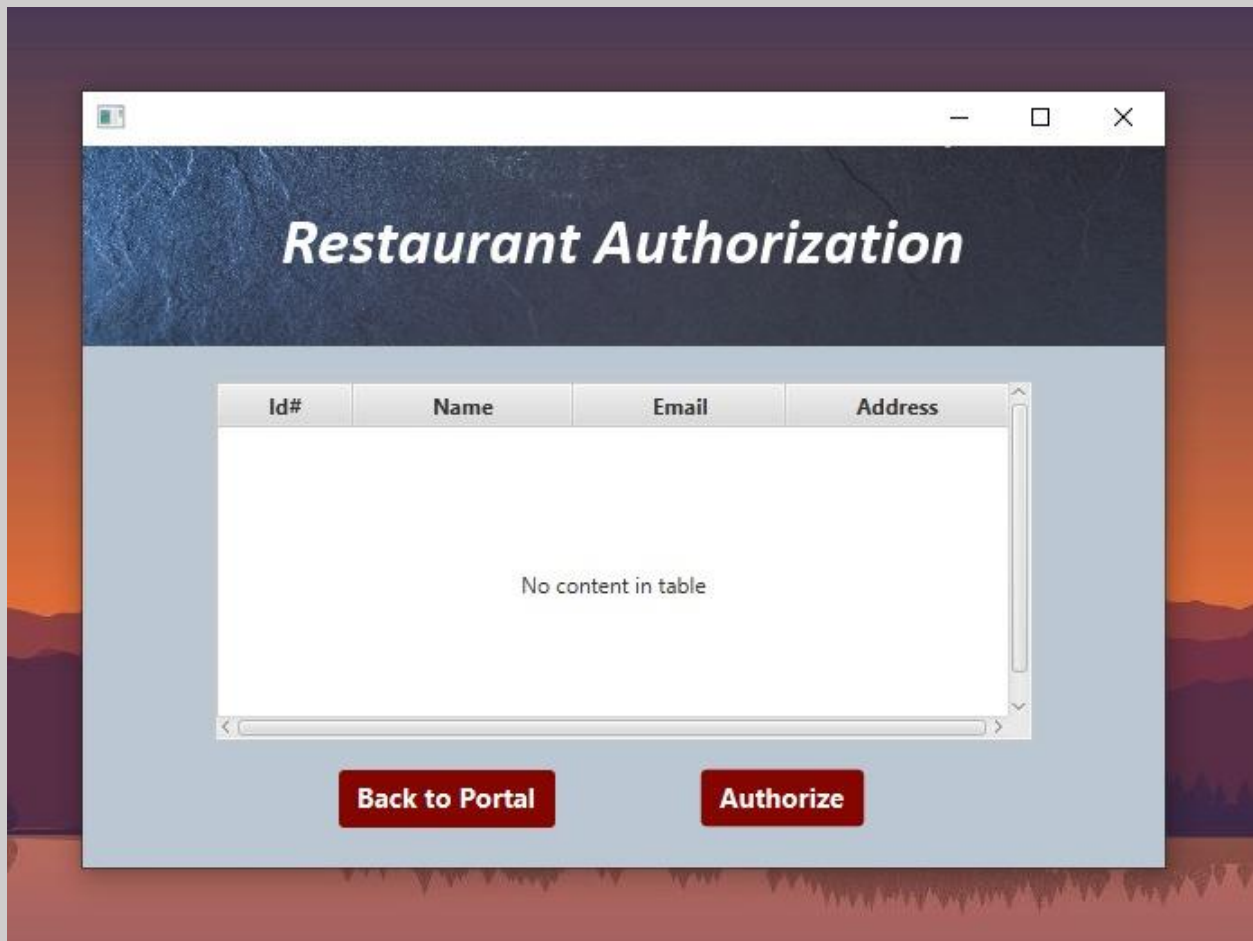


The admin can perform authorization operation on riders as well as restaurants and this can be shown on the authorization page displayed after the admin clicks on the button.



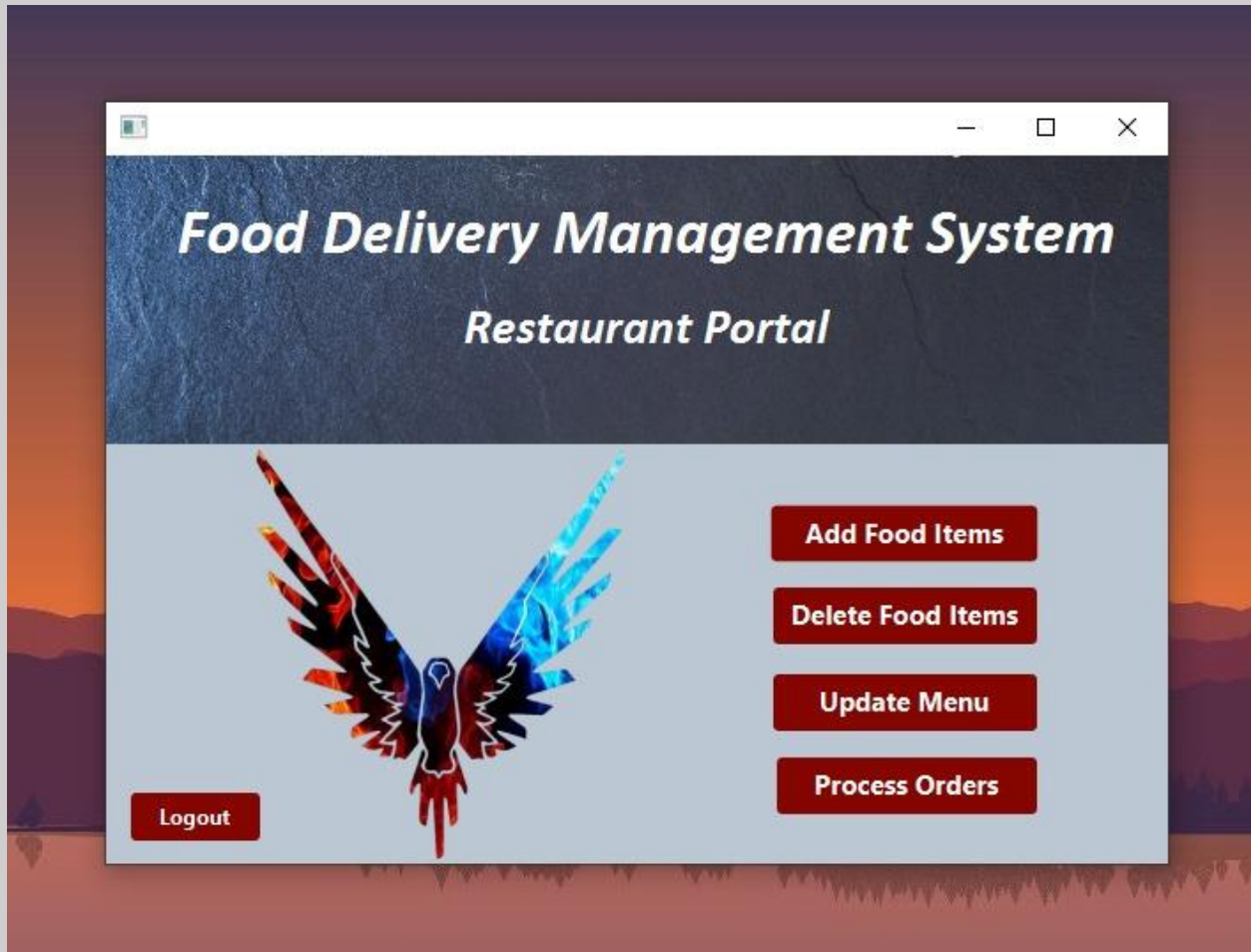
The table will show all the restaurant that have to be authorized and the admin can pick any restaurant of their choice. Upon selecting the restaurant and authorizing the restaurant will be removed from the table.

This is shown below:



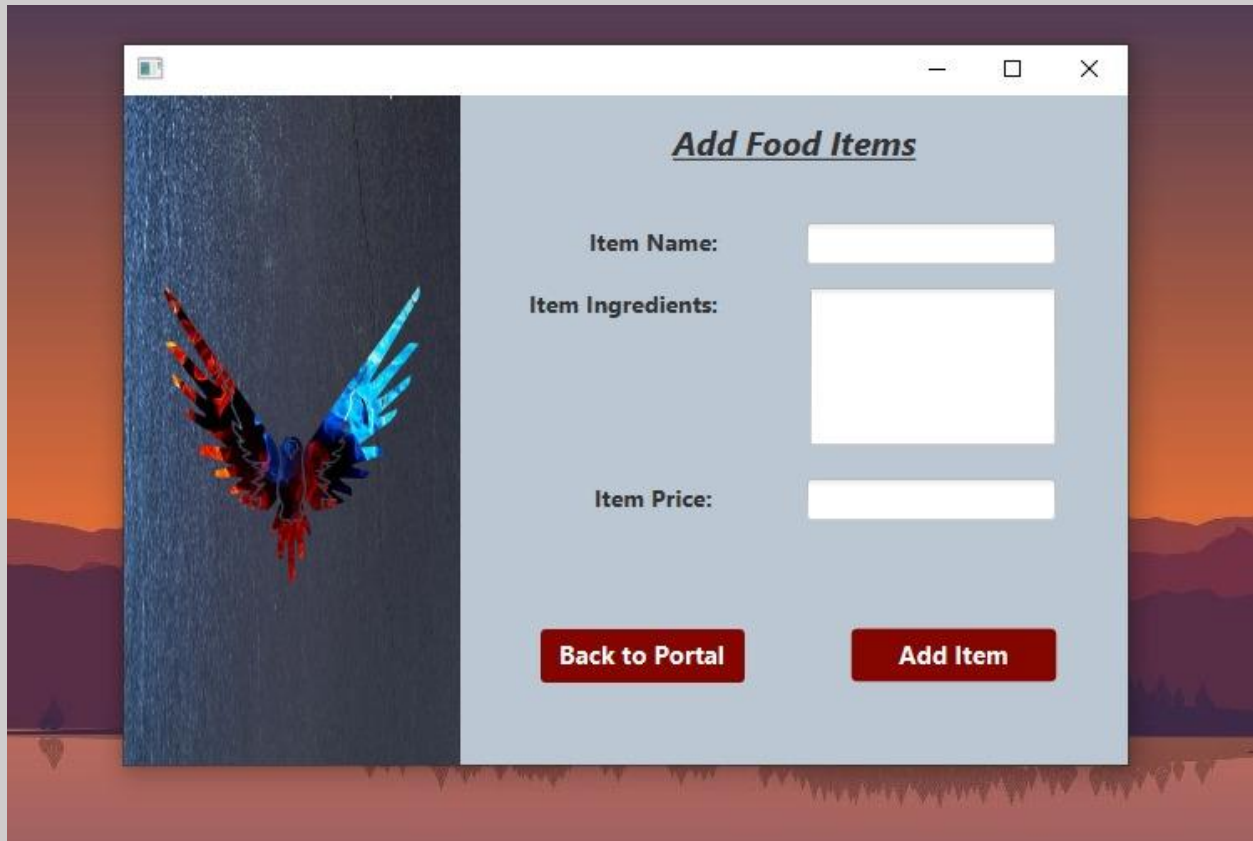
This was all for one of the users which was admin now we go towards the other user of our system restaurant.

Below is the restaurant portal that is shown when the restaurant owner logs into the system, here they are shown all the operations that they can perform.



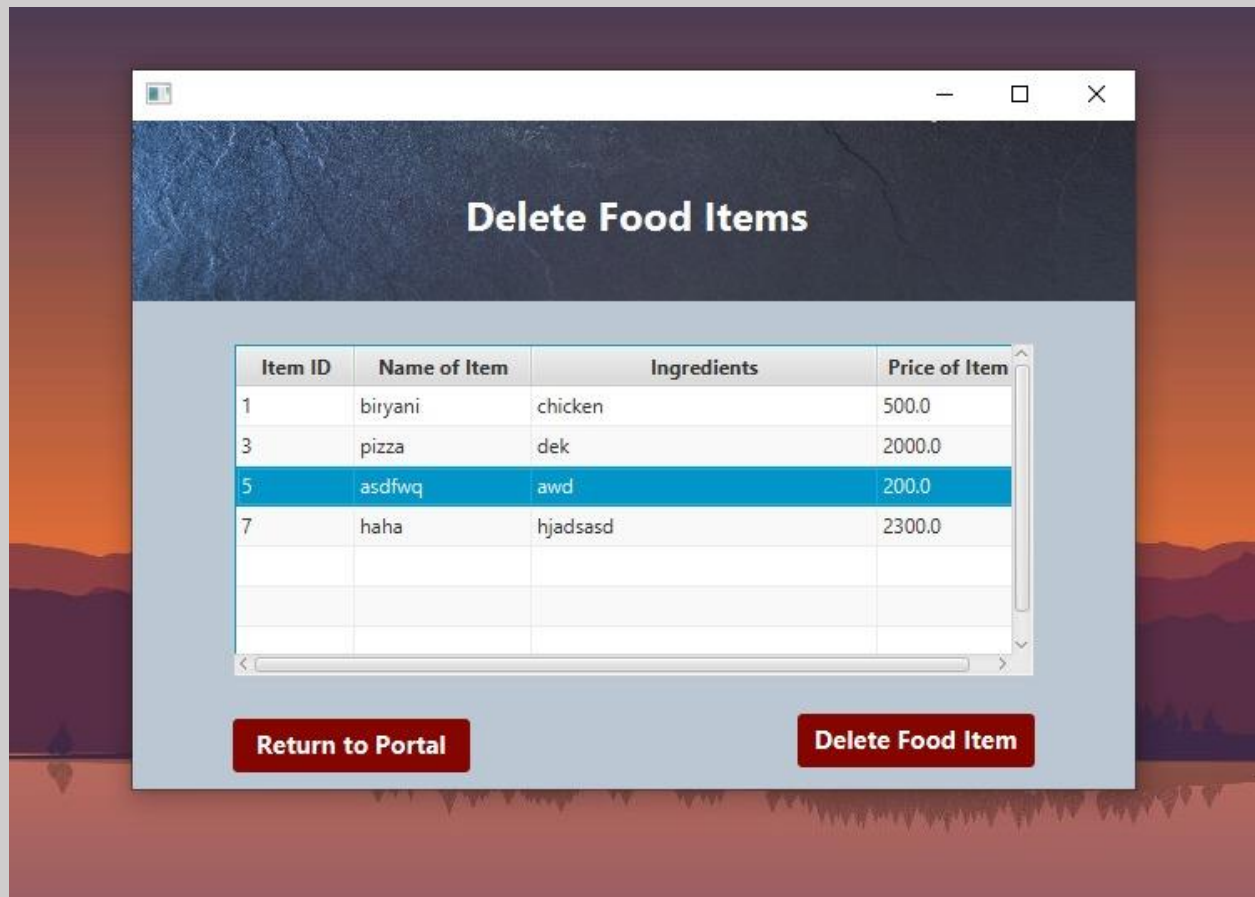
The restaurant staff can select from any of the following operations and it will take them to a new interface of that specific operation.

Below is the add food items page that is one of the primary operation of restaurant, where the restaurant can add new items to their menu.



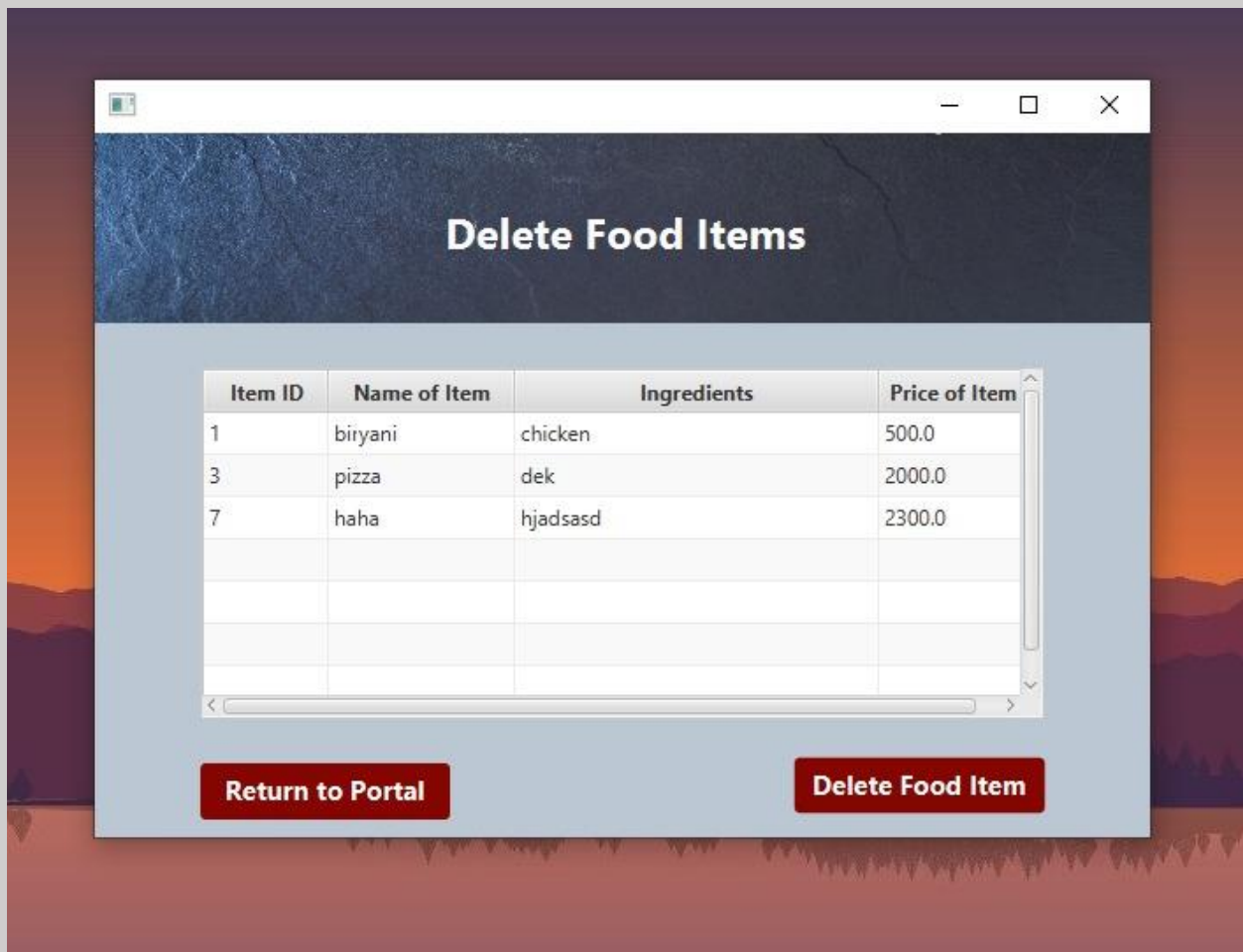
The restaurant fills in all the details and clicks on add item to successfully add it to their menu.

Next is the restaurant delete operation shown where the restaurant staff can delete a selected item from their menu and the new and updated menu will be shown to them right after they delete the item.



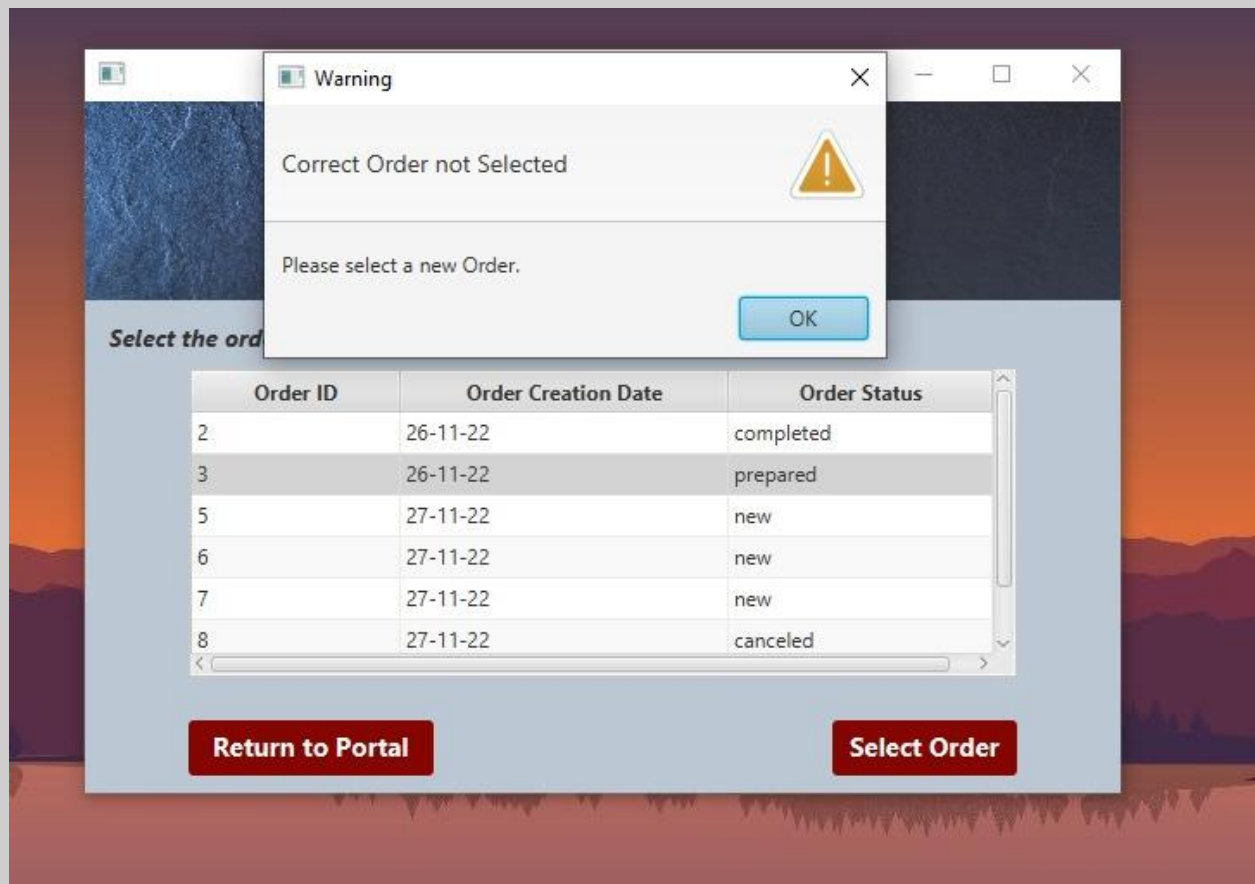
The selected item will be deleted upon clicking the delete food item button.

Below is shown that the selected item is now not in the menu anymore and has been deleted. This can be done for any and all items in the menu.



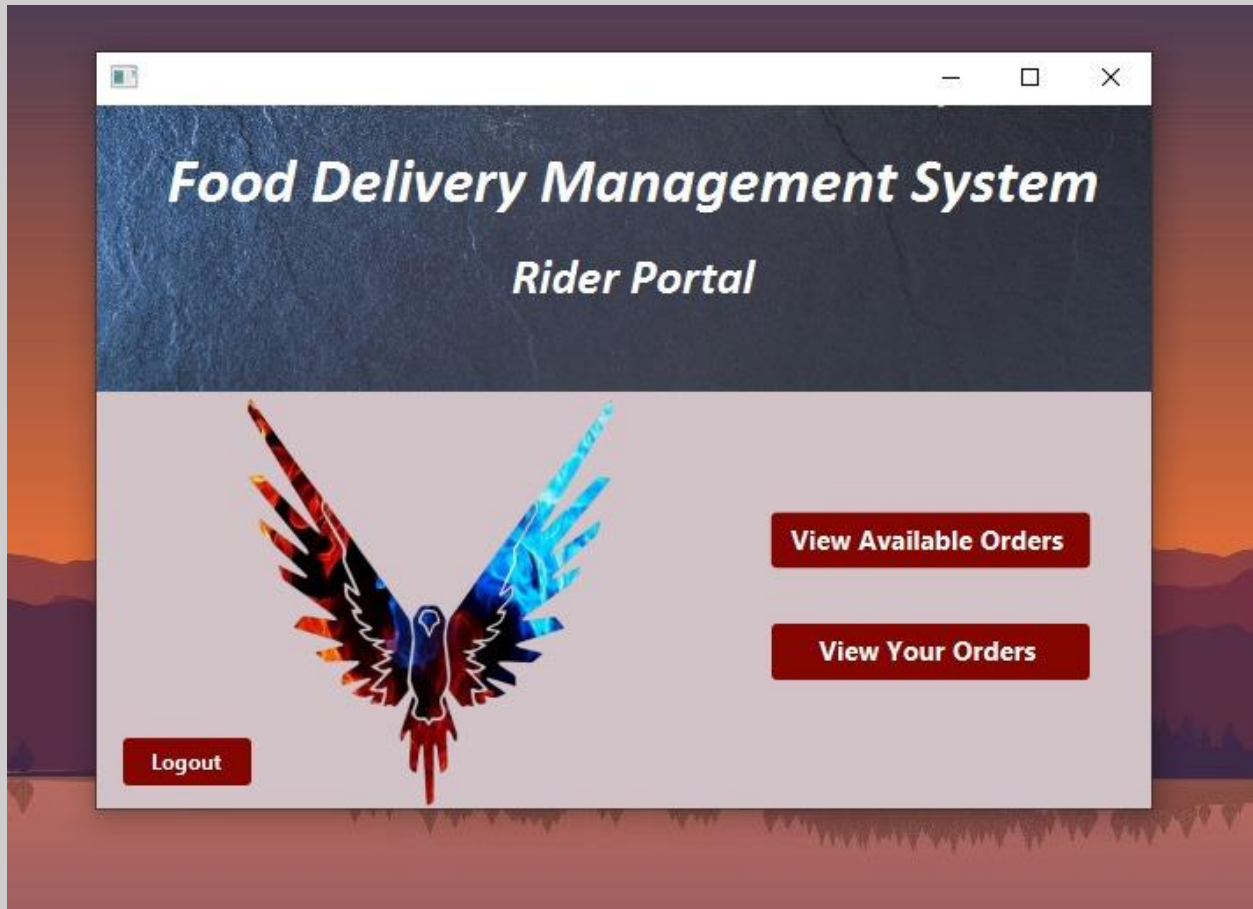
The system is also designed to display warning messages and warning text when the user do not select the only acceptable option, this will repeat for all the times that the user do not select the correct option and upon correct option selection there will be no warning shown, this is also down for the forms where the text data is needed and is necessary and upon missing any field empty the user is indicated to fill in that field to proceed further.

Below is shown the example of such a warning the system generates:



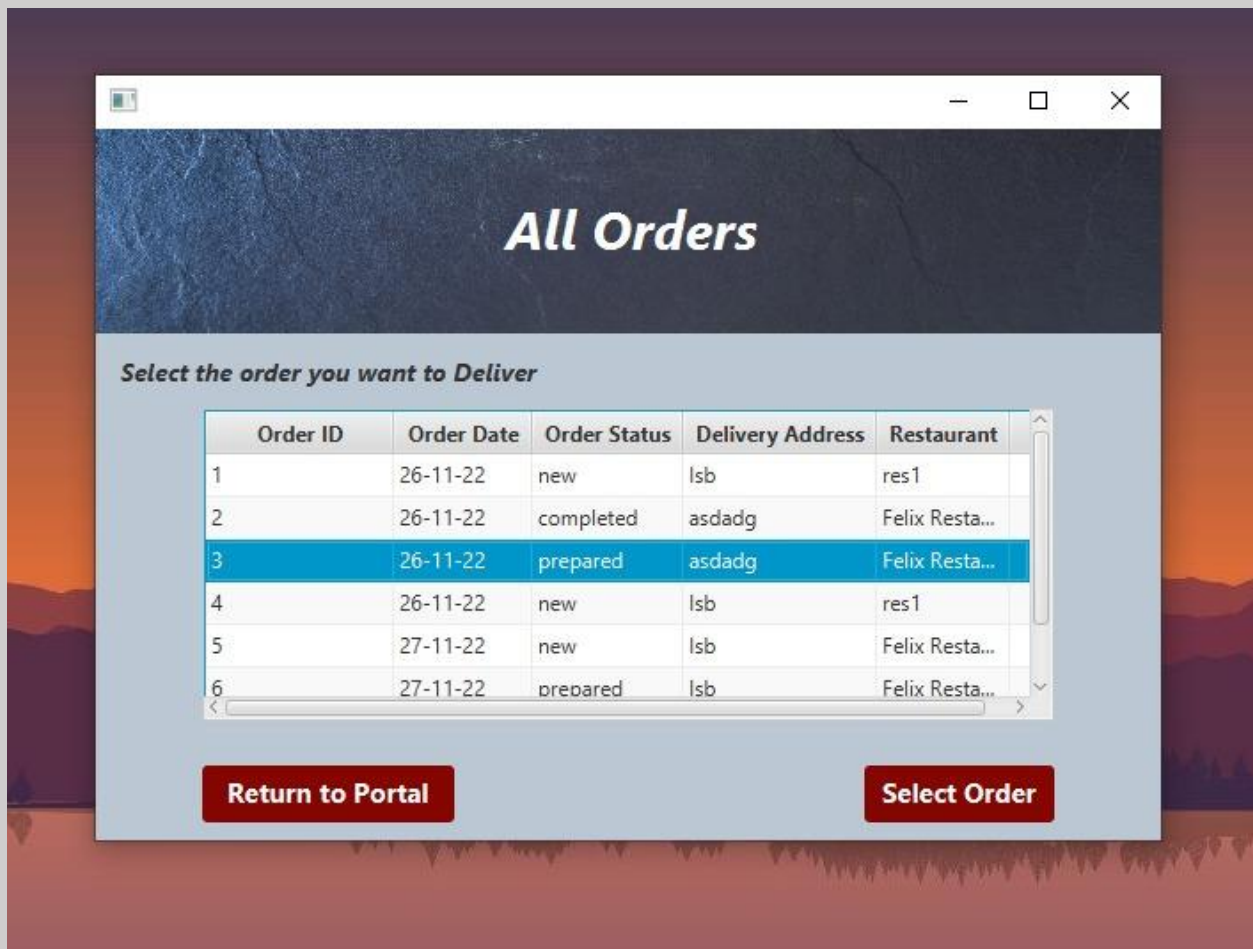
This was all for one of the users which was restaurant now we go towards the other user of our system rider.

Below is the rider portal that is shown when the rider logs into the system, here they are shown all the operations that they can perform.



The operations of view available orders and view your orders is available to the riders.

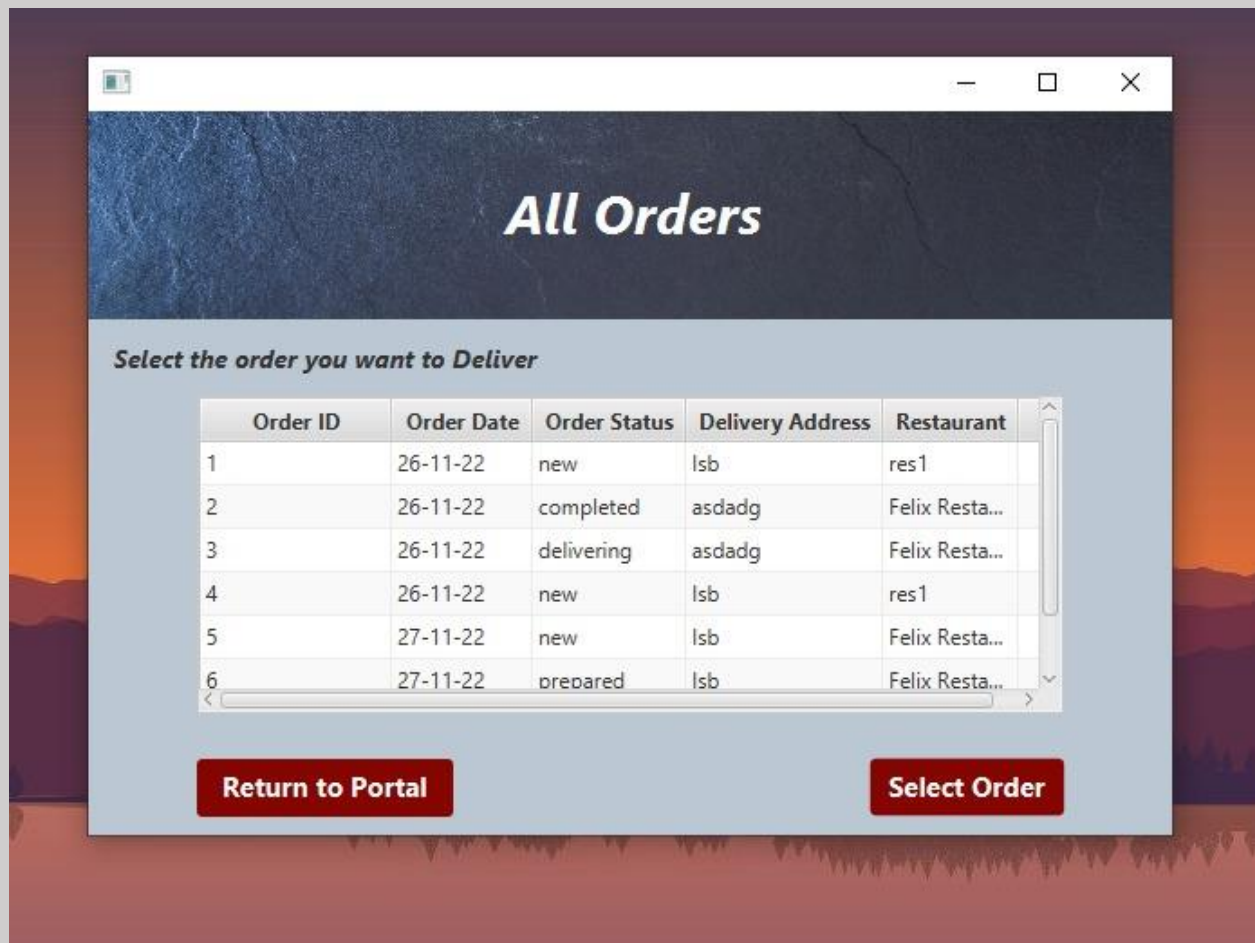
The view available orders operation opens to the page where the rider can select a order to deliver. Below is the page shown:



The selected ordered will now be selected and its status will then be changed.

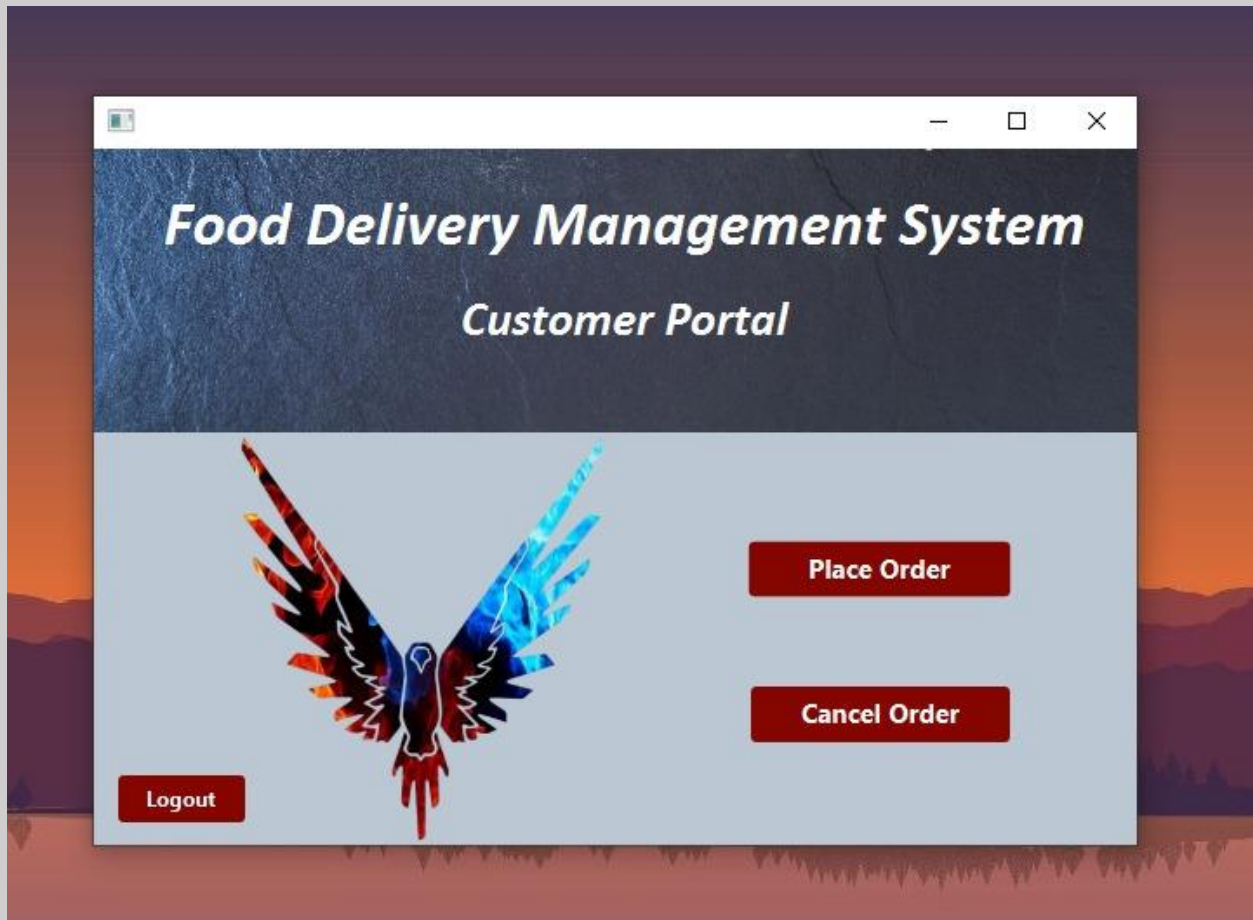
Below is the page after the order has been selected and the rider has clicked select order button.

And it can be seen that the order status has been changed to delivering.



This was all for one of the users which was customer now we go towards the other user of our system Customer.

Below is the customer portal that is shown when the customer logs into the system, here they are shown all the operations that they can perform.



The customer has the option to select from placing a new order or cancel an existing order.

When the customer clicks on the cancel order button he is shown a list of his orders that are not yet prepared or delivered and from those orders they can select which order to cancel. Upon which they fill a small form and then cancel the order.

The screenshot shows a web application titled "Cancellation Order Portal". Below the title, there is a instruction: "Select the order you want to cancel from the order list below." A table lists orders with columns for Order ID, Order Creation Date, and Order Status. Order ID 4 is highlighted. To the right of the table, there are input fields for "Order ID:" (containing 4), "Restaurant Name:" (containing res1), and "Reason for Cancellation:". At the bottom, there are three buttons: "Return to Portal", "Select Order", and "Cancel Order".

Order ID	Order Creation Date	Order Status
1	26-11-22	new
4	26-11-22	new
5	27-11-22	new
6	27-11-22	prepared
7	27-11-22	new
8	27-11-22	canceled

Order ID: 4

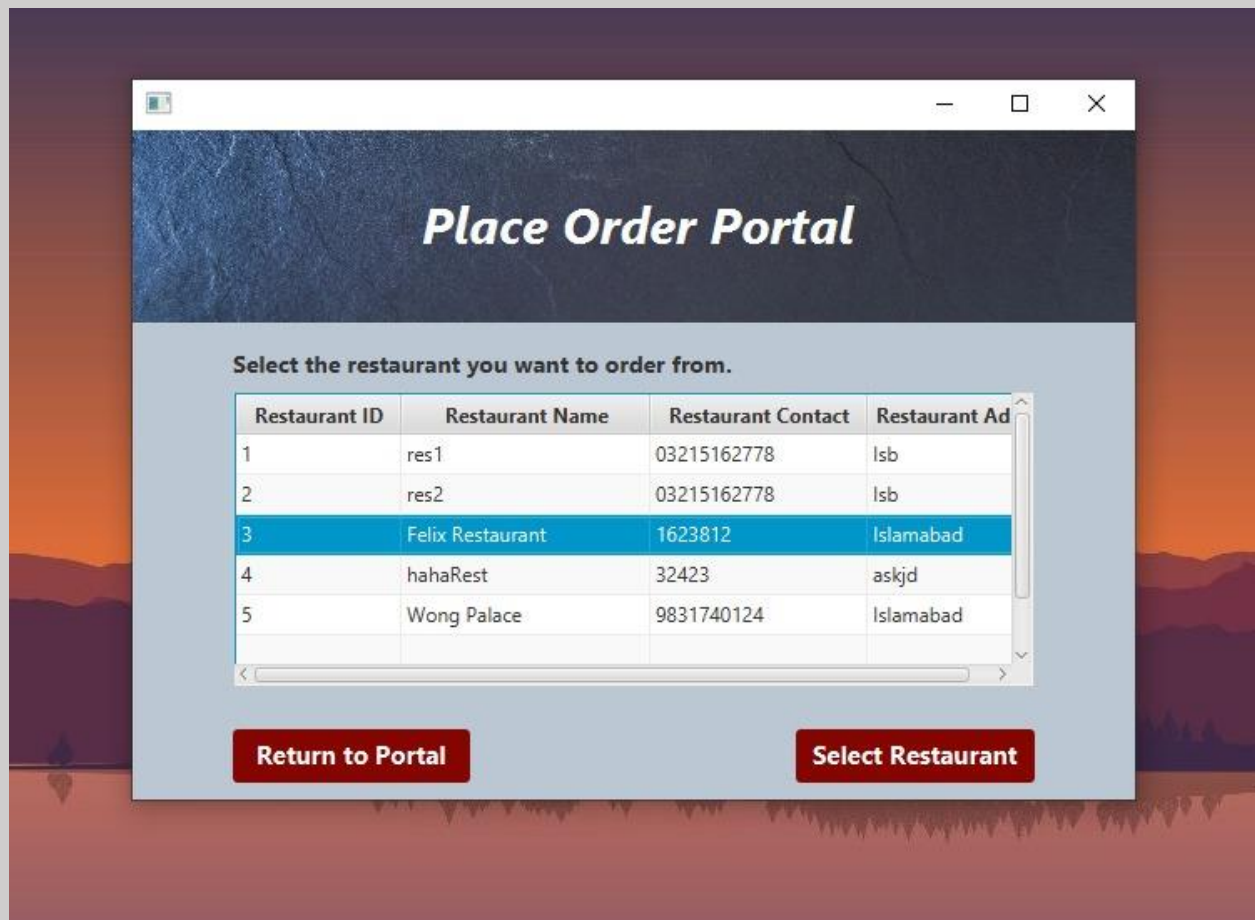
Restaurant Name: res1

Reason for Cancellation:

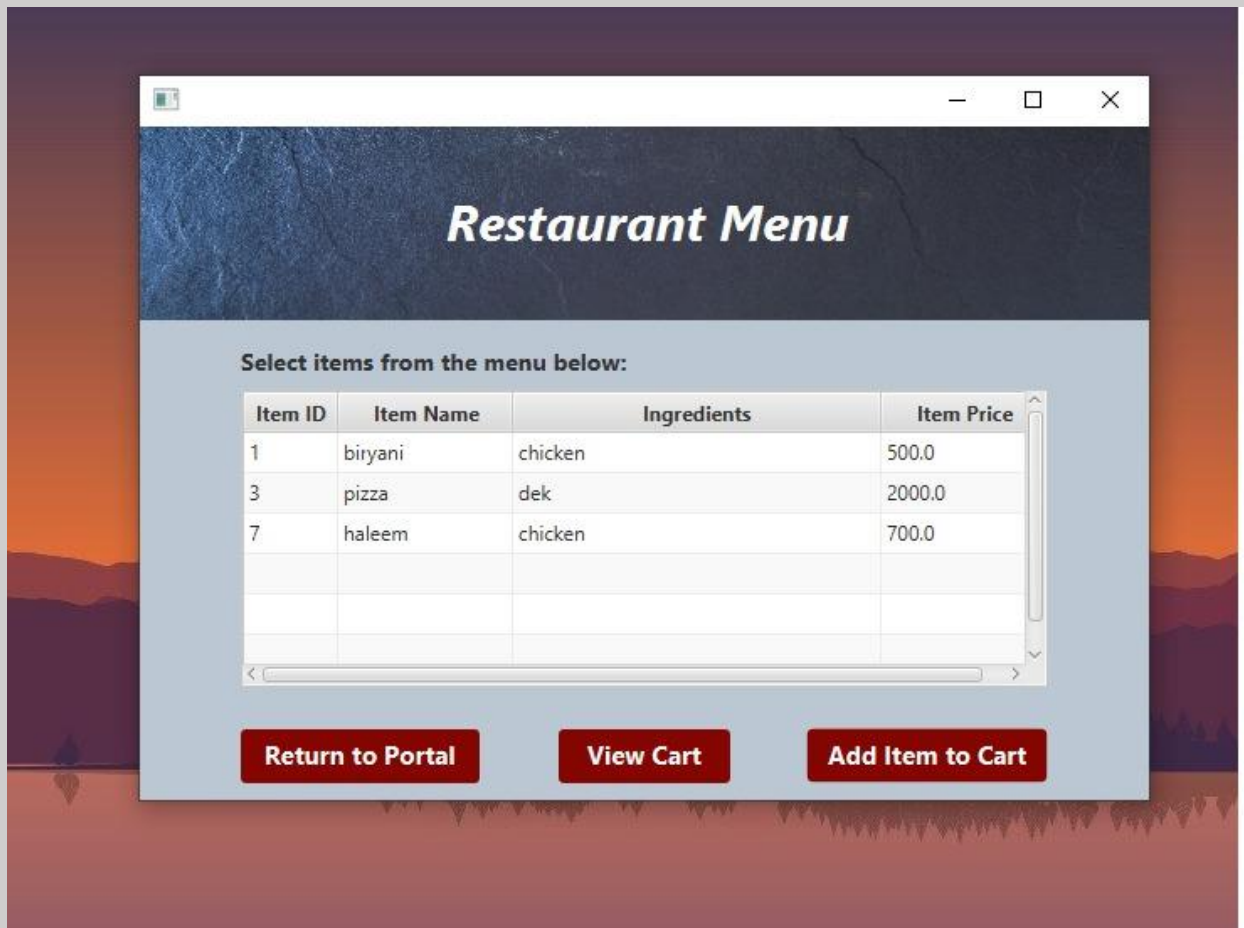
Return to Portal Select Order Cancel Order

Upon selecting the order they want to delete and then clicking the select order button the name and order id of that order are shown in the display boxes and then the customer fills in the reason for the cancellation which is not a compulsory text box.

The other operation that the customer can perform is the place order operation.



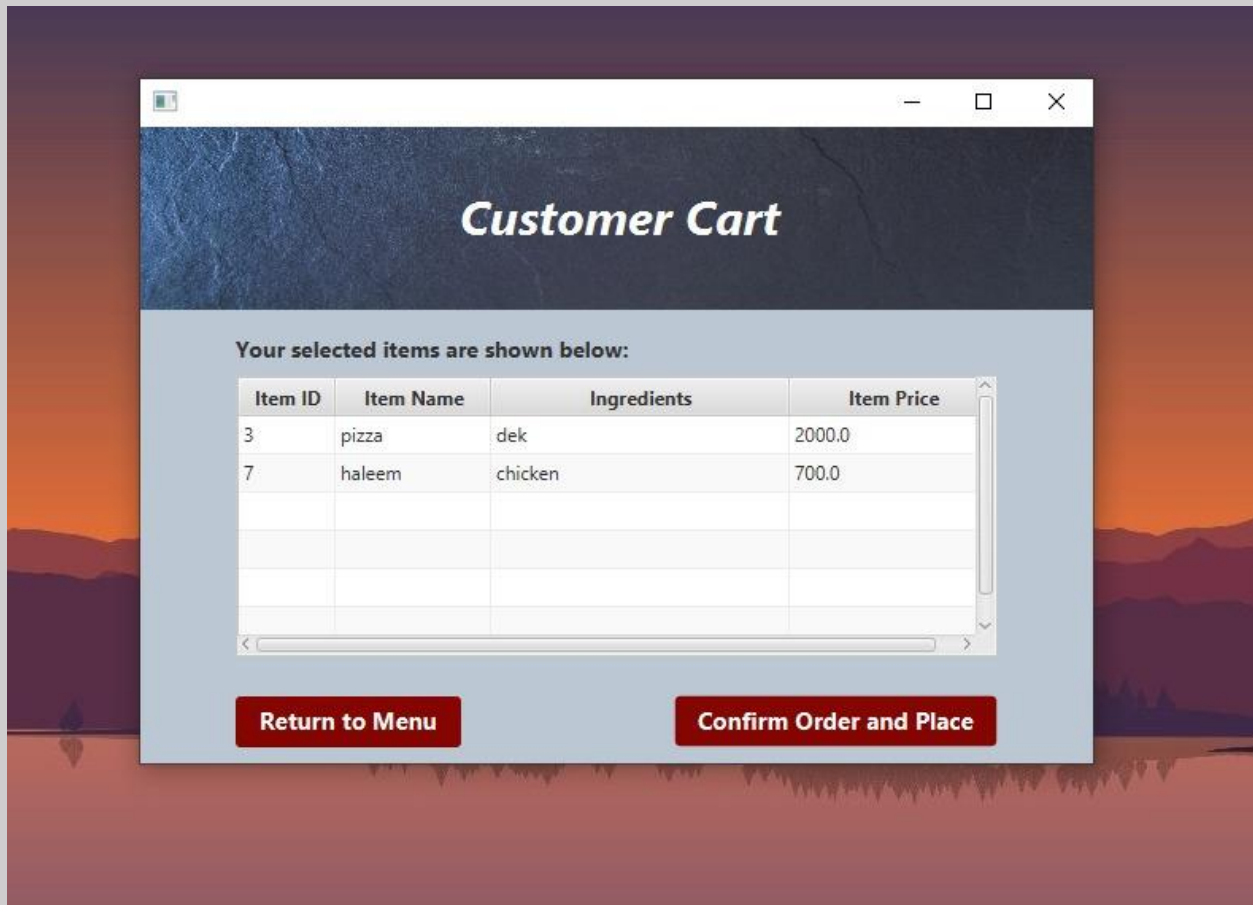
Here the customer is shown the list of restaurants from where he can select the restaurant he wants to order from. Then the customer clicks select restaurant button and is taken to the restaurant menu page where the menu of the selected restaurant is displayed to the user.



On this page the customer selects the food items that he wants to order. The price of those items is mentioned along with the item names and item ids.

Upon selection of the items the customer can click the add item to cart button and that adds that item in the customers cart and a prompt message is shown to the user that tells them that the item has been successfully added into the cart.

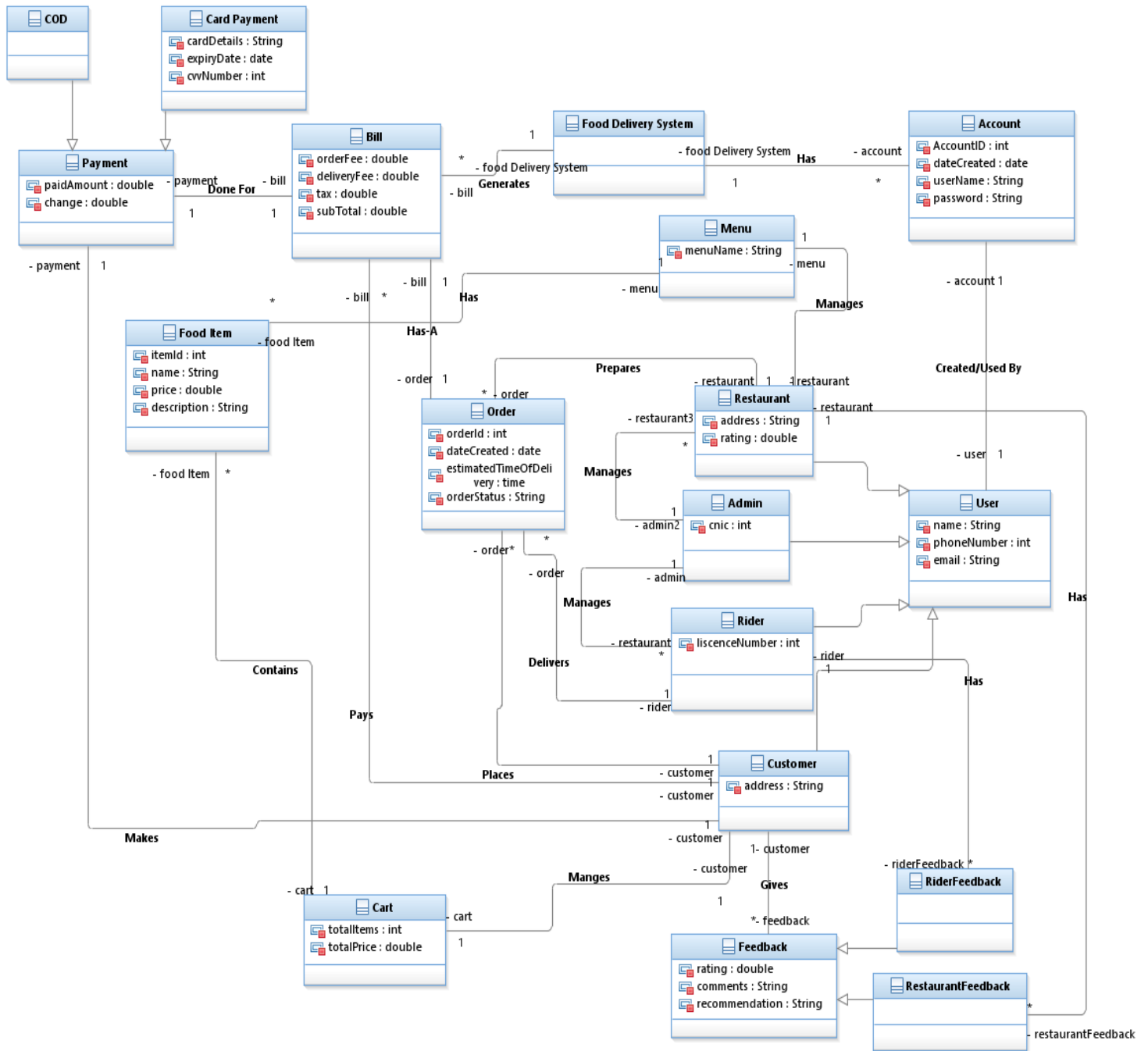
The customer when done adding items into the cart can then click on view cart button and that takes the user to the cart page where the customer is shown all the items that he selected. Below is page for the cart page that is displayed:



The above interface shows the customer that they have selected the shown items and the customer is given the option to go back to the restaurant menu and select more items or confirm the order and place it.

Upon selecting the confirm order and place button the customer is shown a message that the order has been placed successfully and that order is then shown to the restaurant.

4. Domain Model

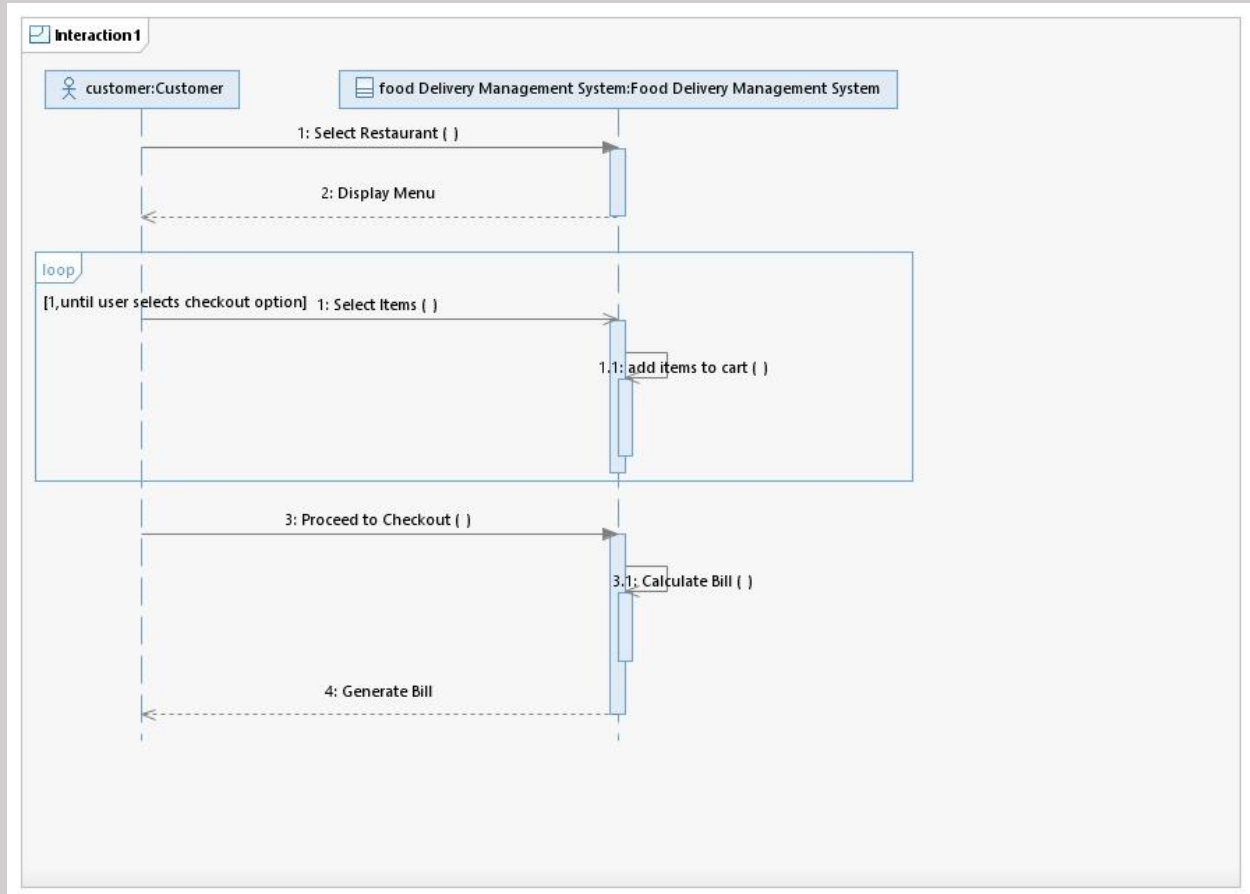


5. System Sequence Diagrams

UC01

Attempted by: Rayed Muhammad Saeed (i201822)

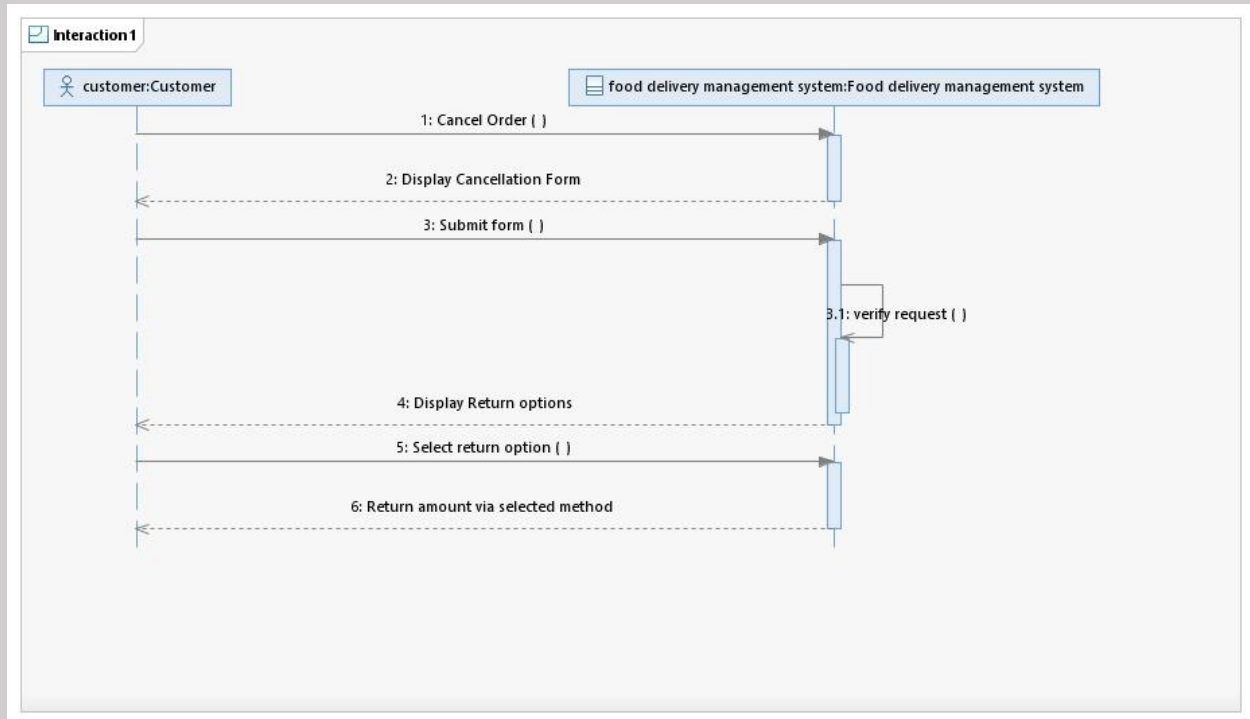
Use case name: Place Order



UC02

Attempted by: Rayed Muhammad Saeed (i201822)

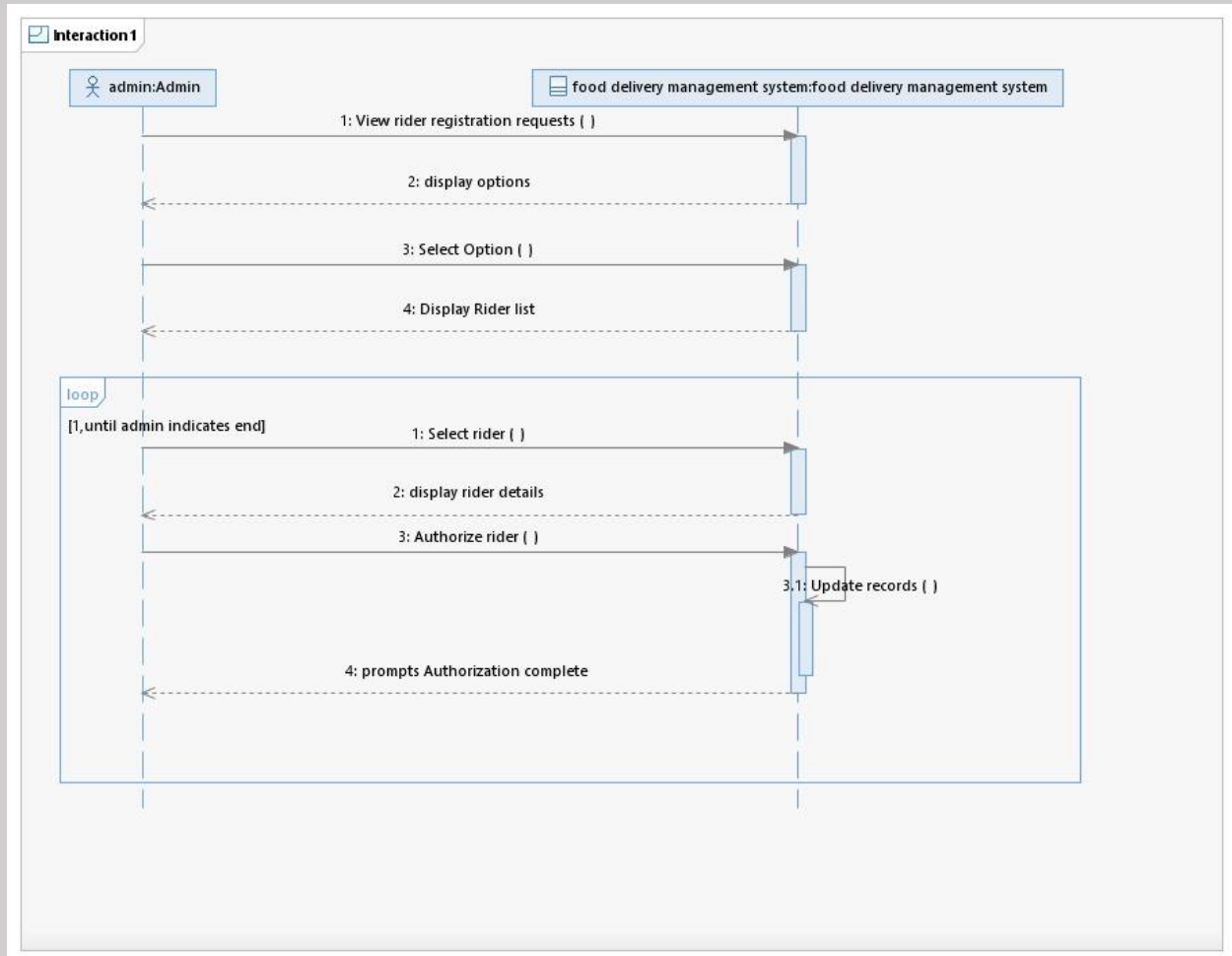
Use case name: Cancel Order



UC03

Attempted by: Rayed Muhammad Saeed (i201822)

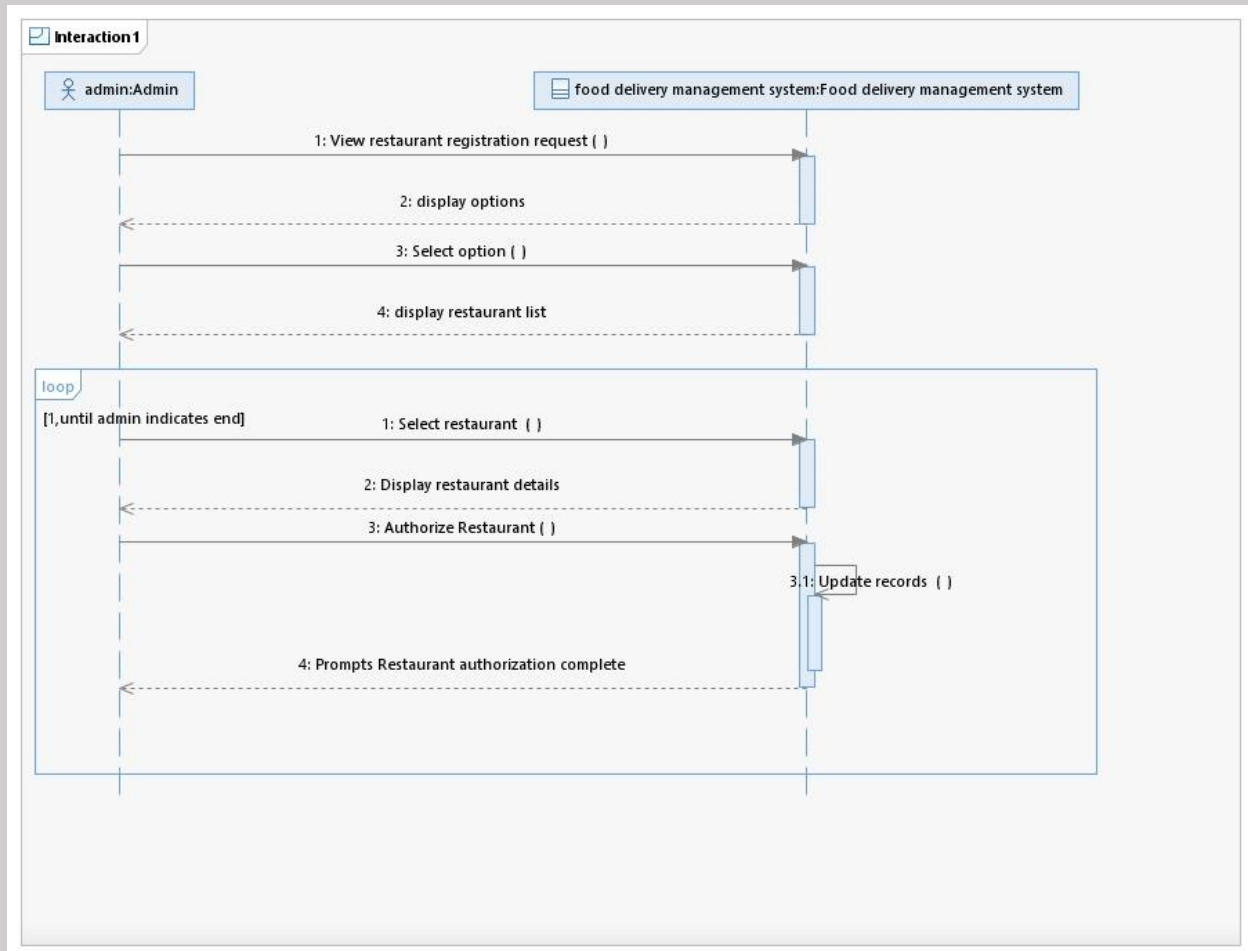
Use case name: Manage Rider



UC04

Attempted by: Rayed Muhammad Saeed (i201822)

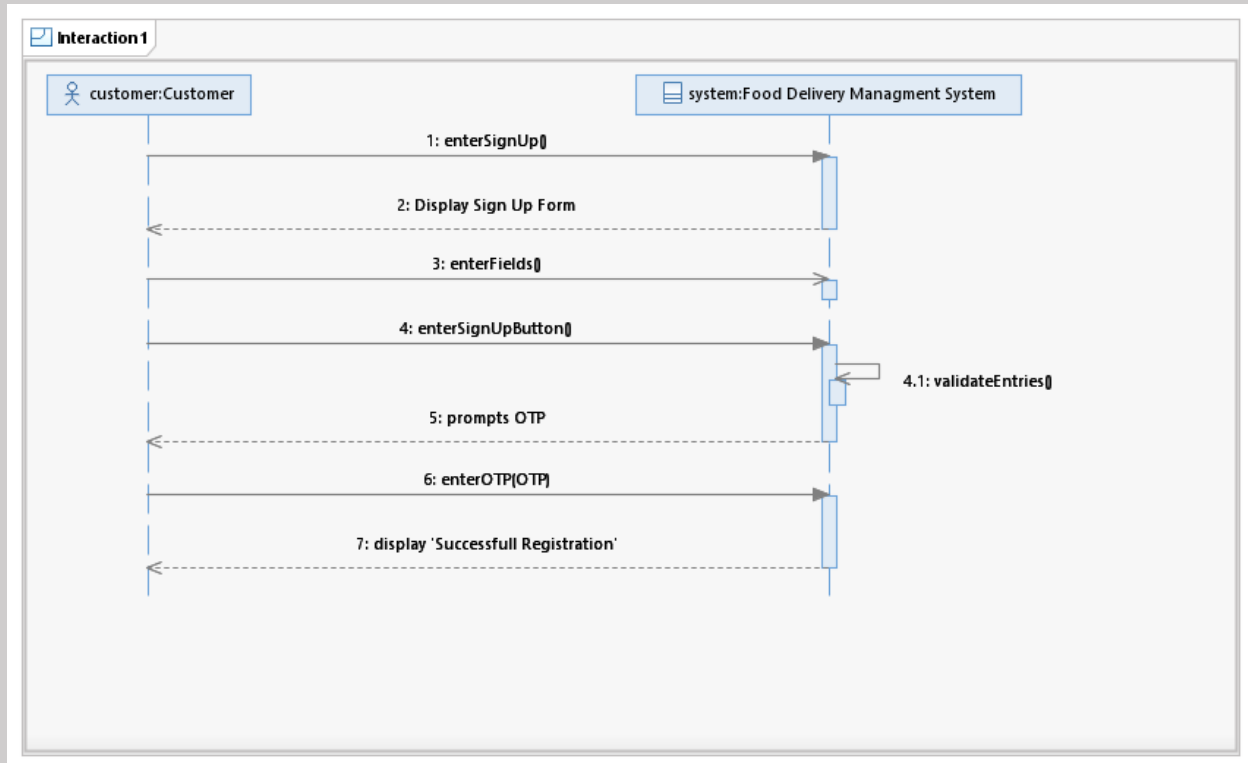
Use case name: Manage Restaurant



UC05

Attempted by: Abdullah Saqib (i200458)

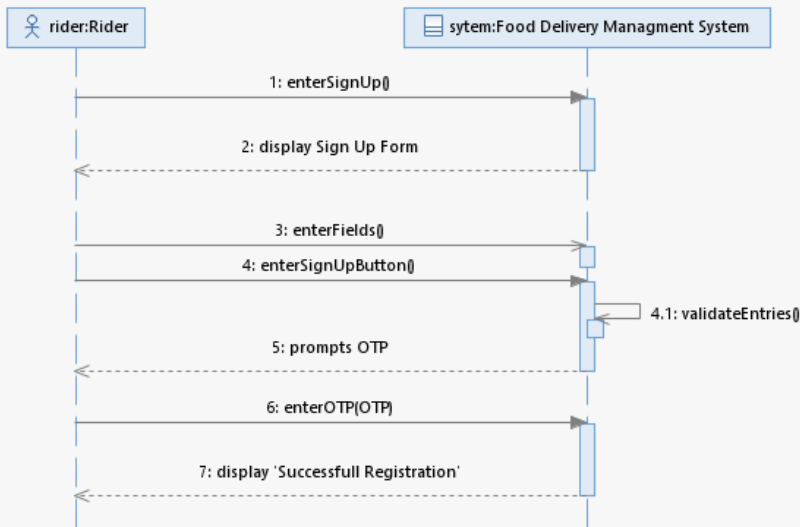
Use Case Name: Register User (Customer)



Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User (rider)

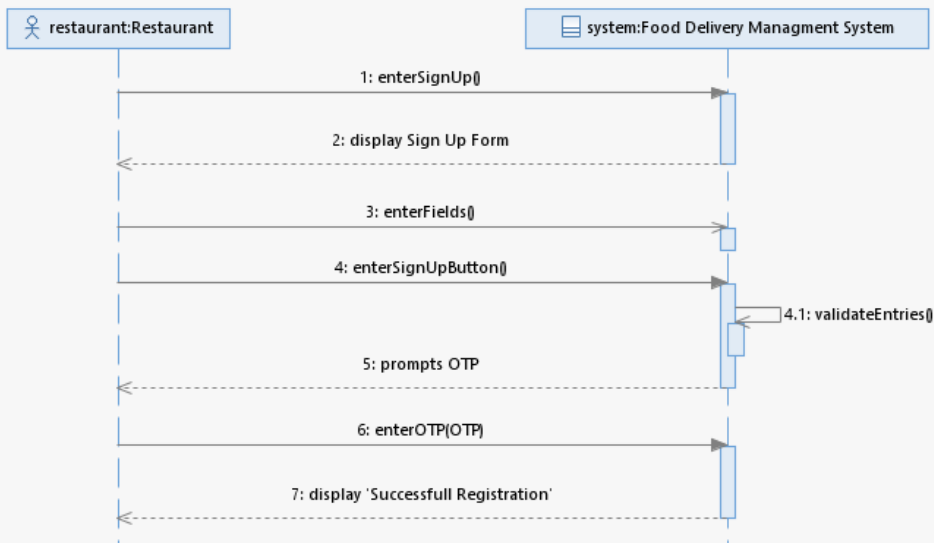
Interaction 1



Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User (restaurant)

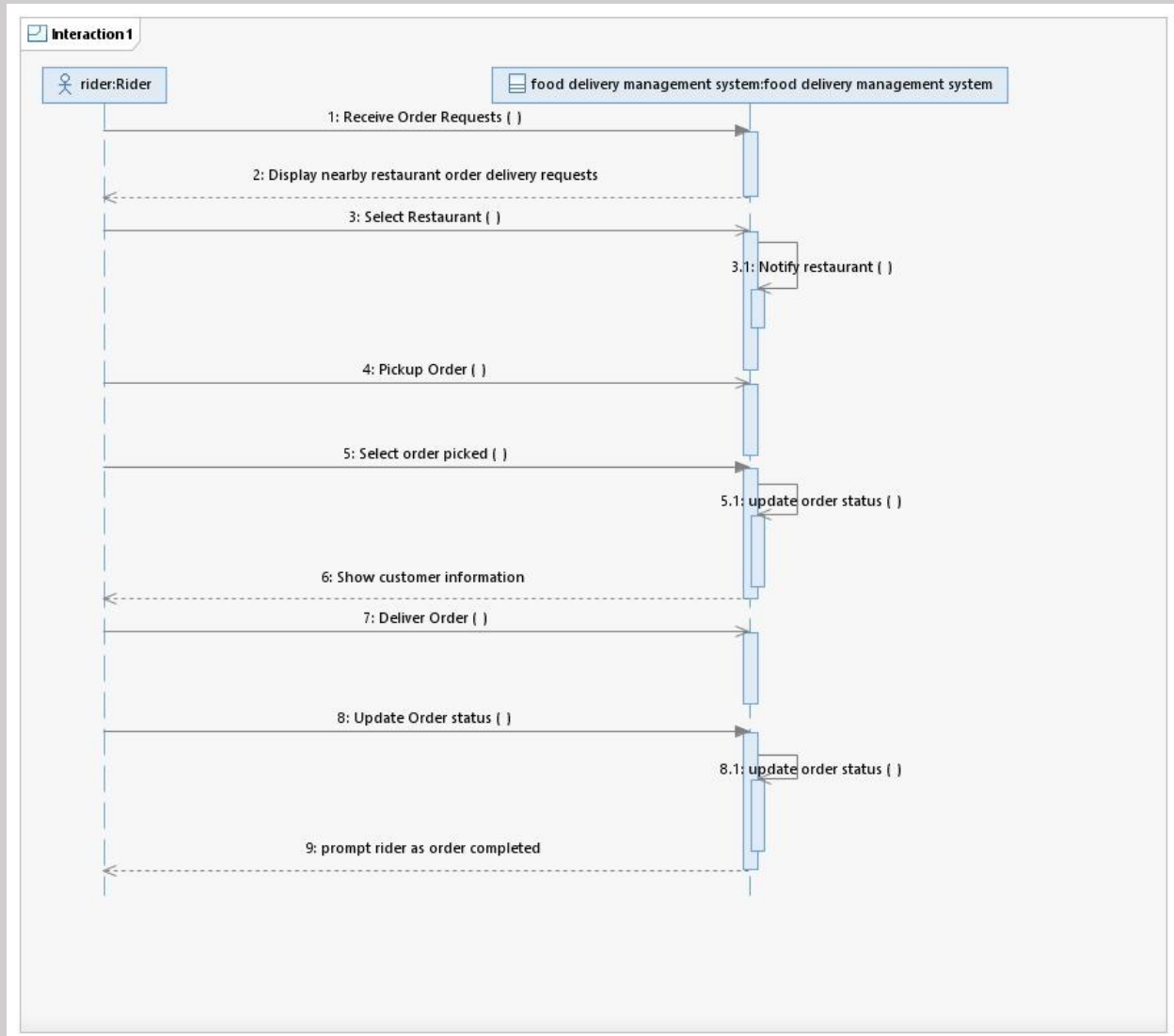
Interaction 1



UC06

Attempted by: Rayed Muhammad Saeed (i201822)

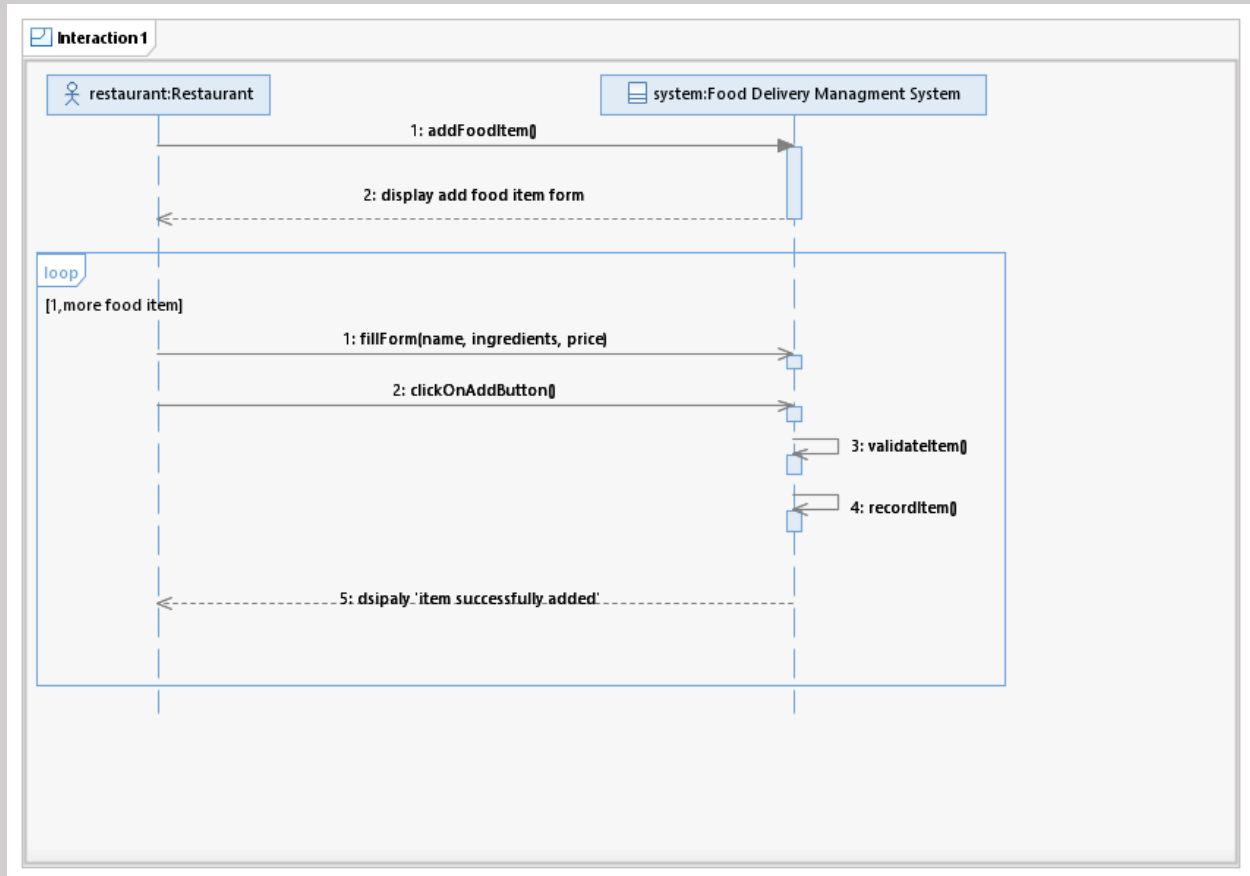
Use case name: Accept Order



UC07

Attempted by: Abdullah Saqib (i200458)

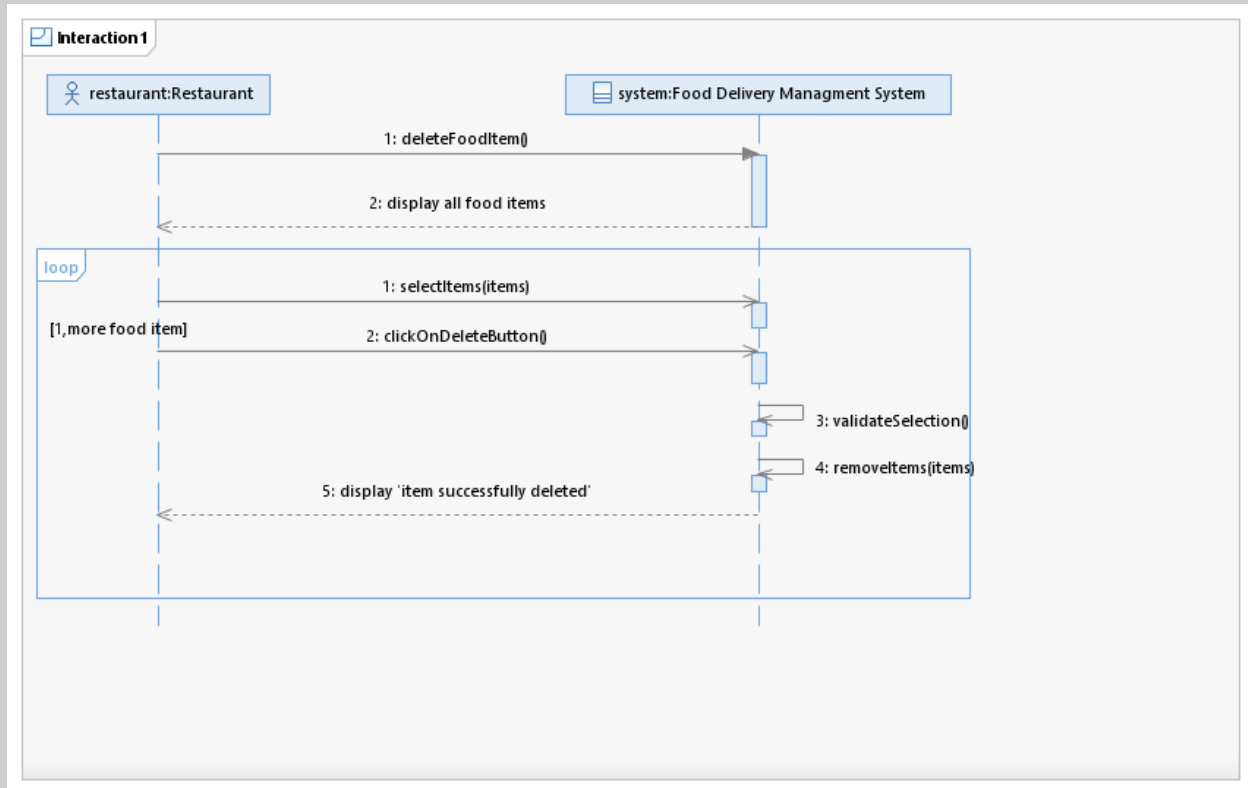
Use Case Name: Add food item



UC08

Attempted by: Abdullah Saqib (i200458)

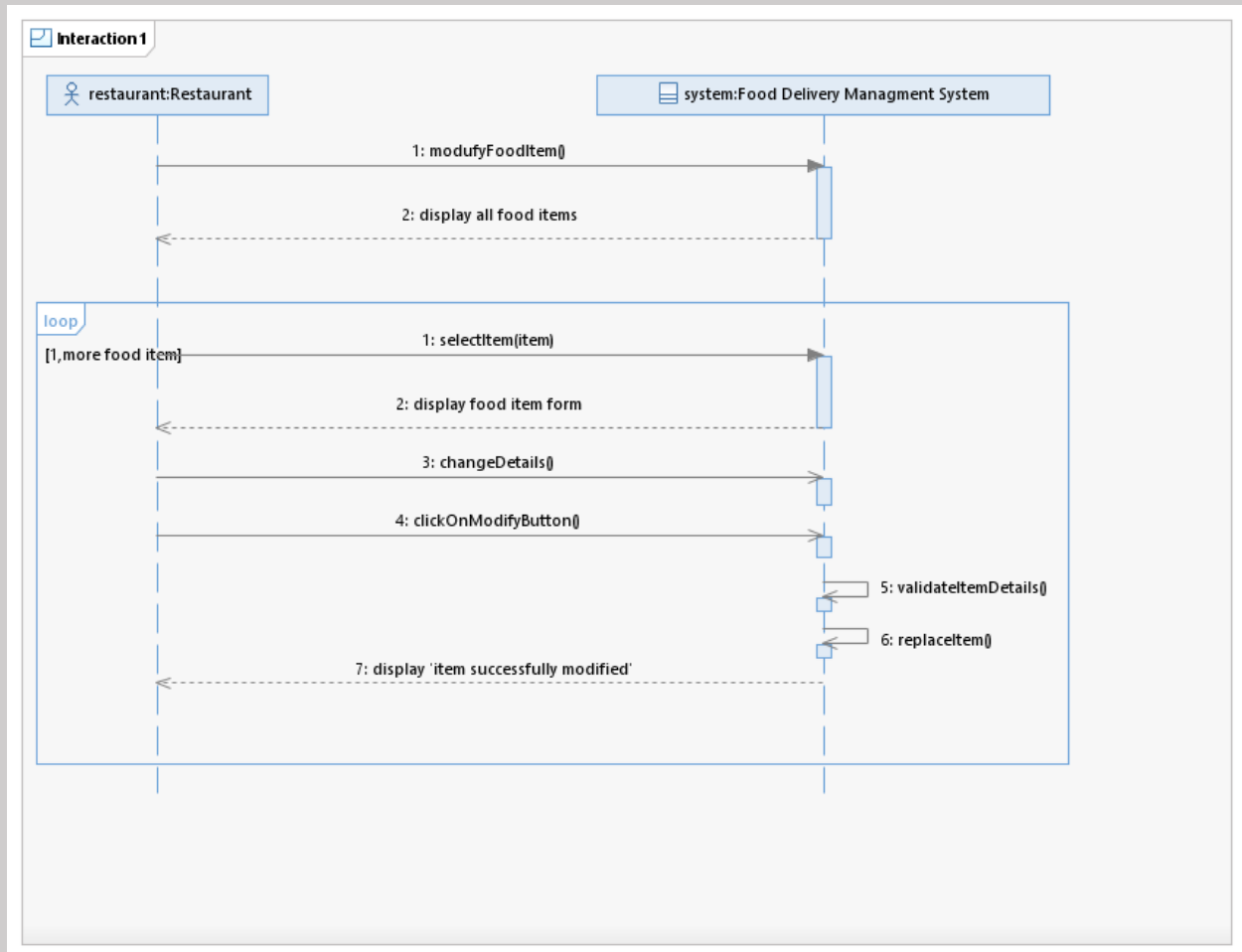
Use Case Name: Delete food item



UC09

Attempted by: Abdullah Saqib (i200458)

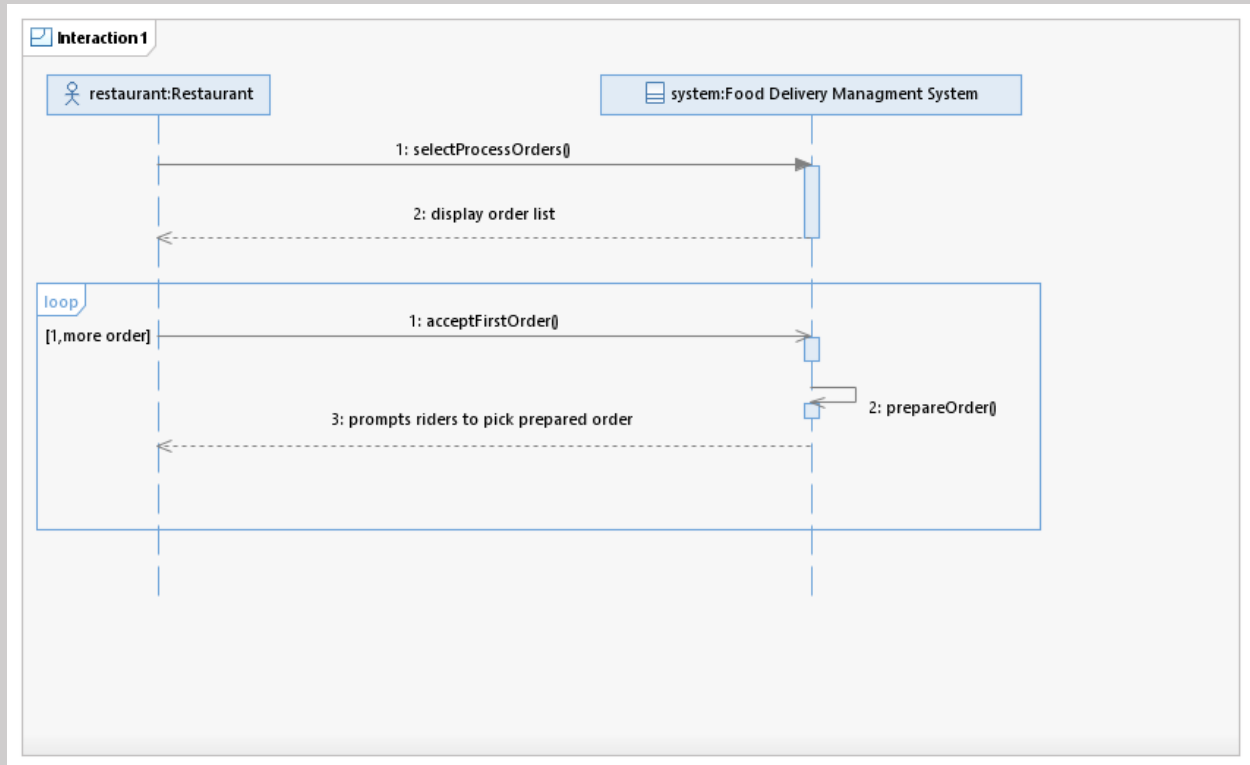
Use Case Name: Modify food item



UC10

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Processing Order

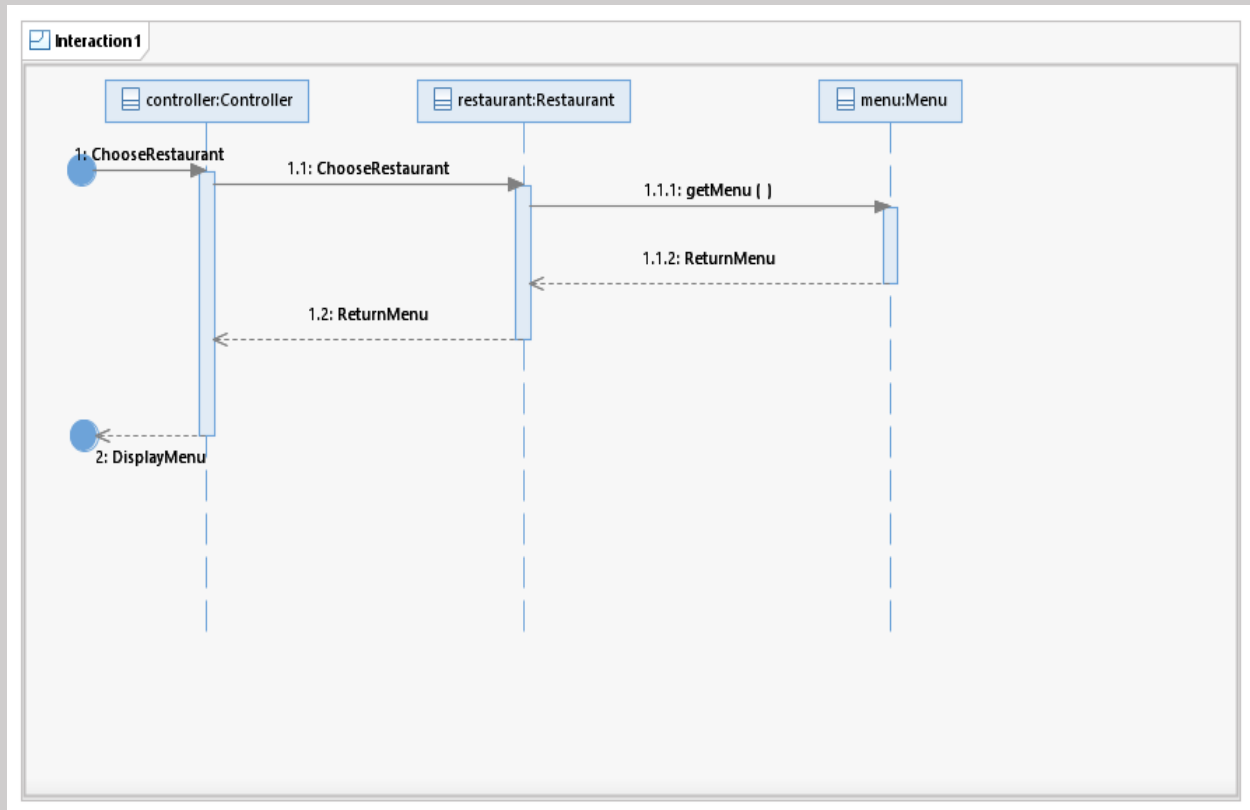


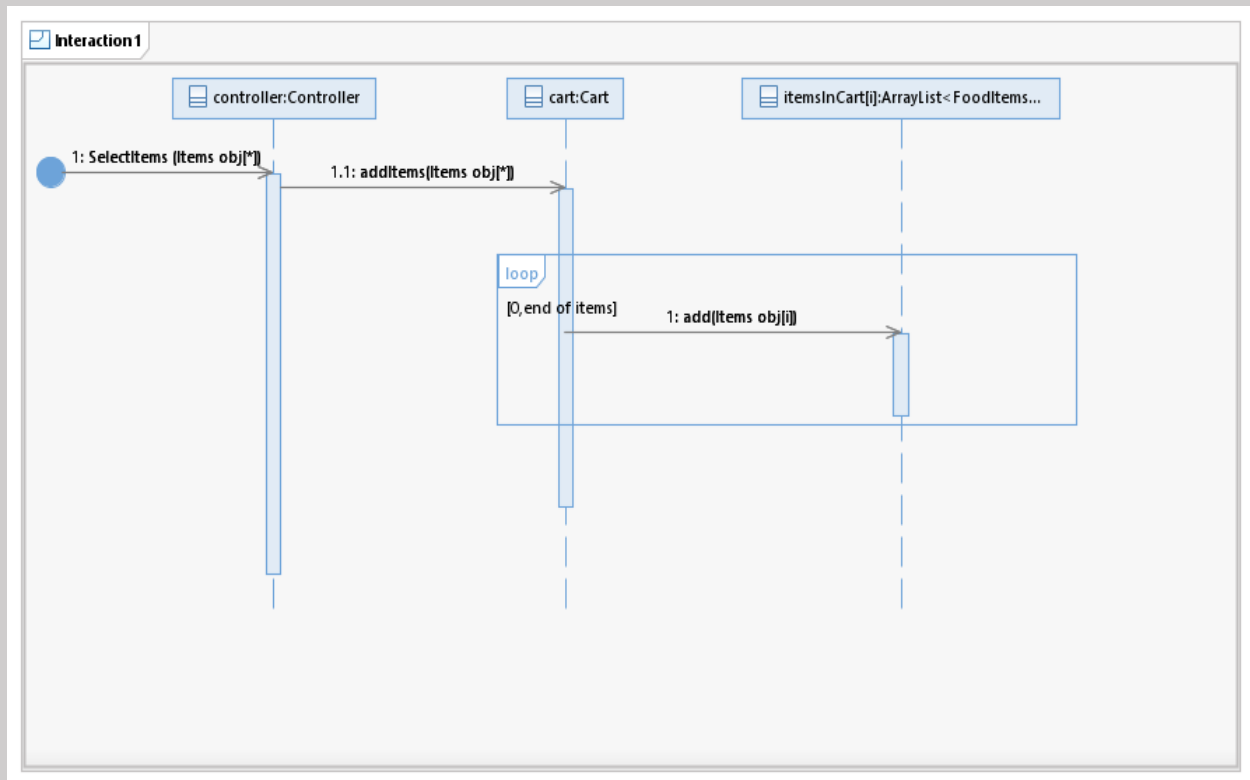
6. Sequence Diagrams

UC01

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Place Order

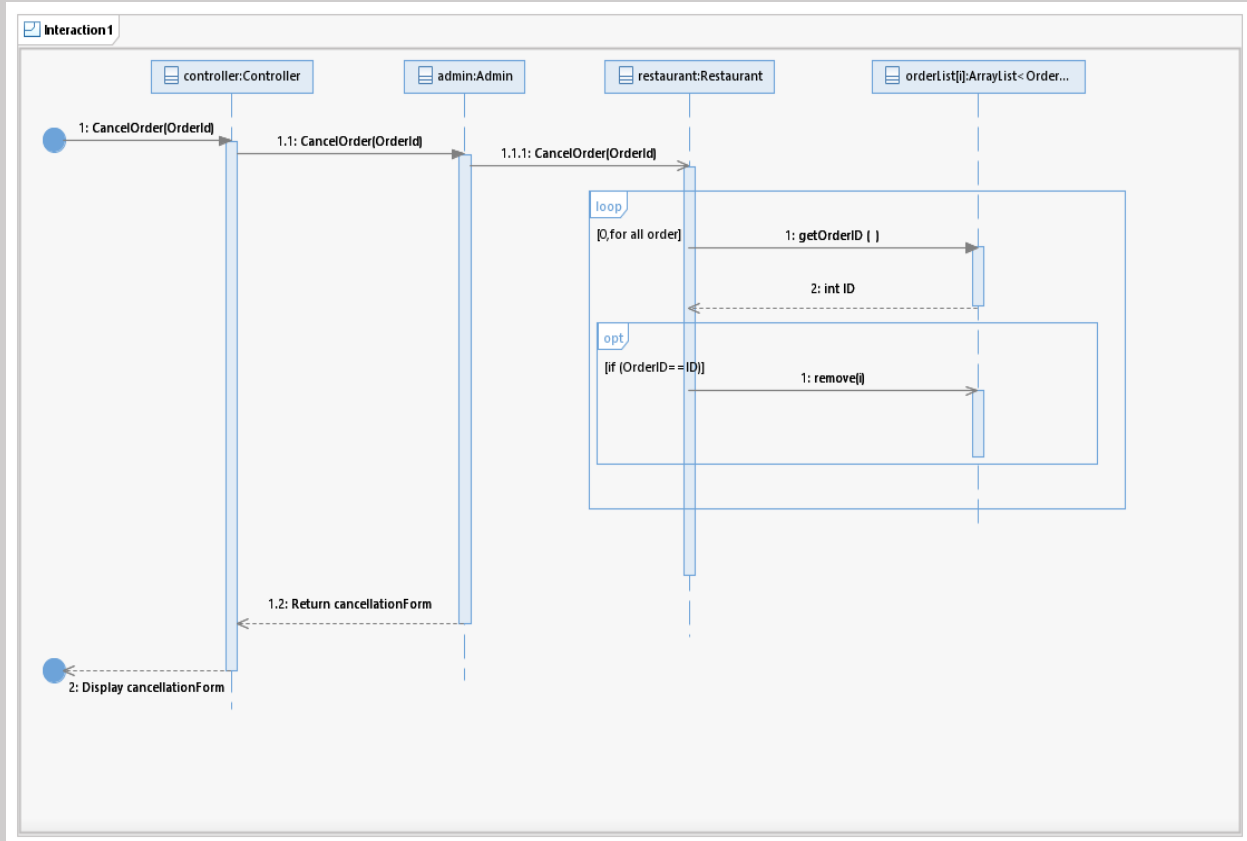




UC02

Attempted by: Rayed Muhammad Saeed (i201822)

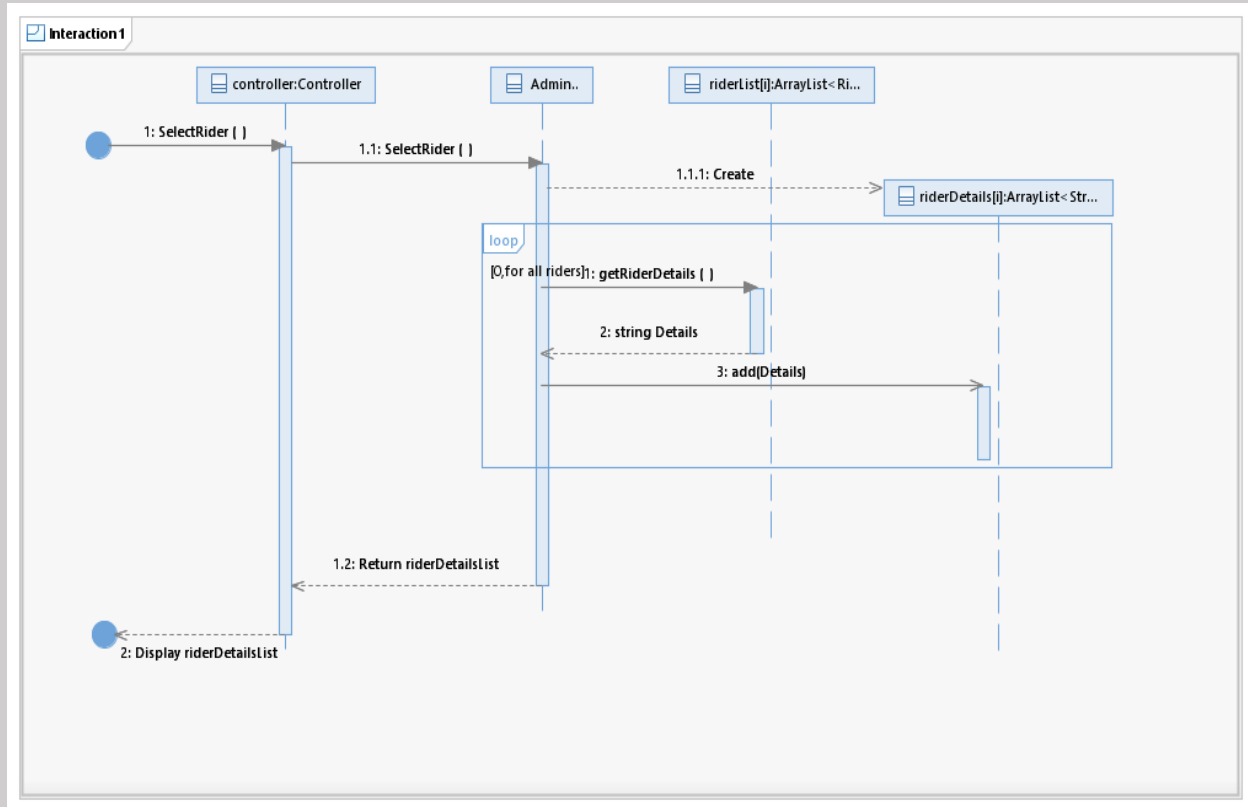
Use case name: Cancel Order



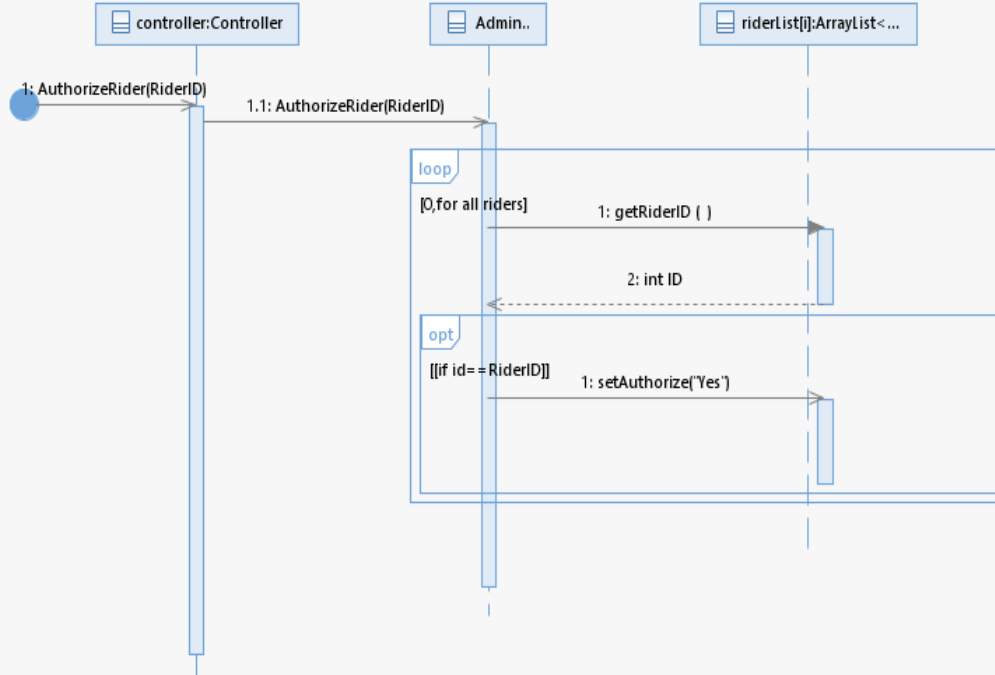
UC03

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Manage Rider



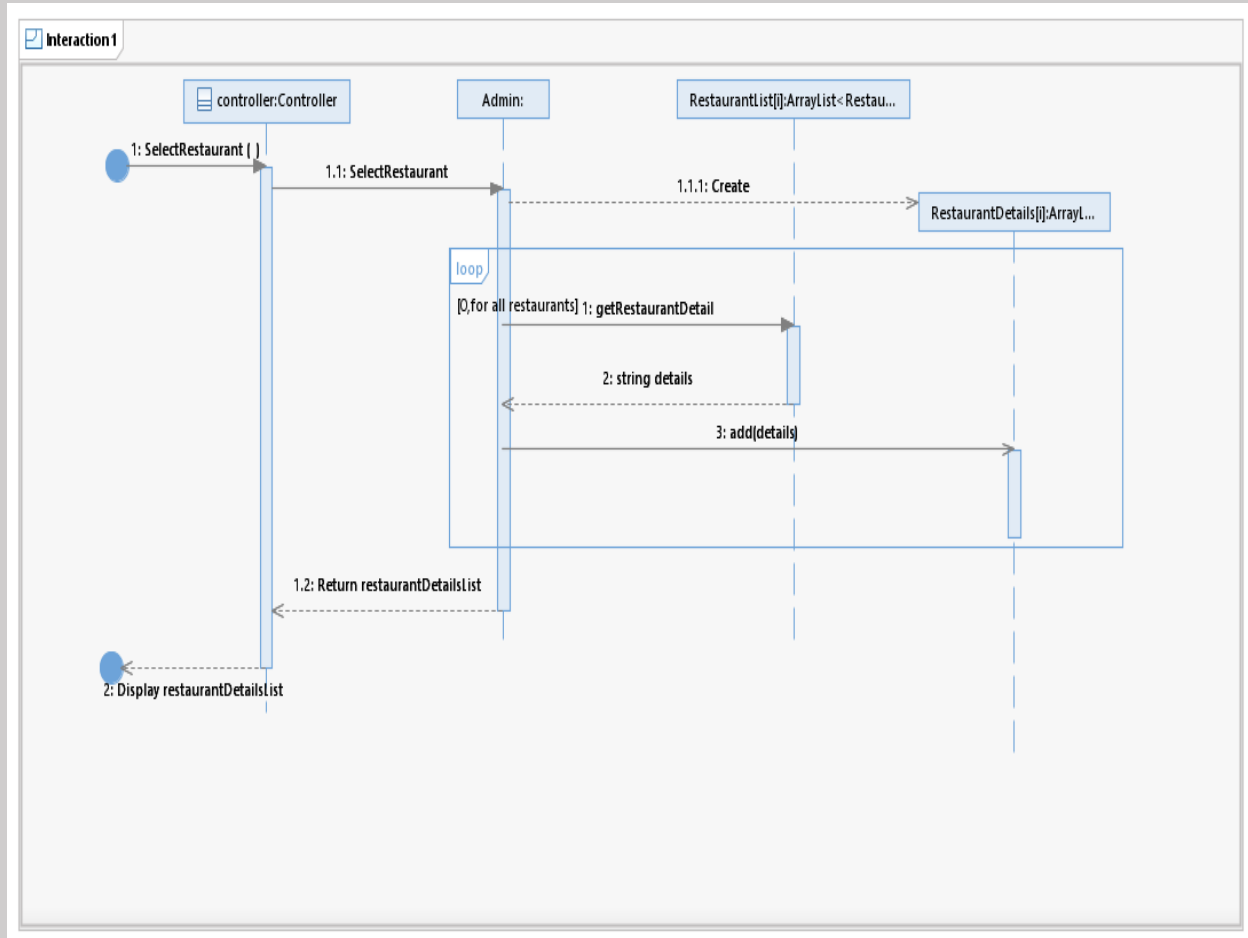
Interaction 1

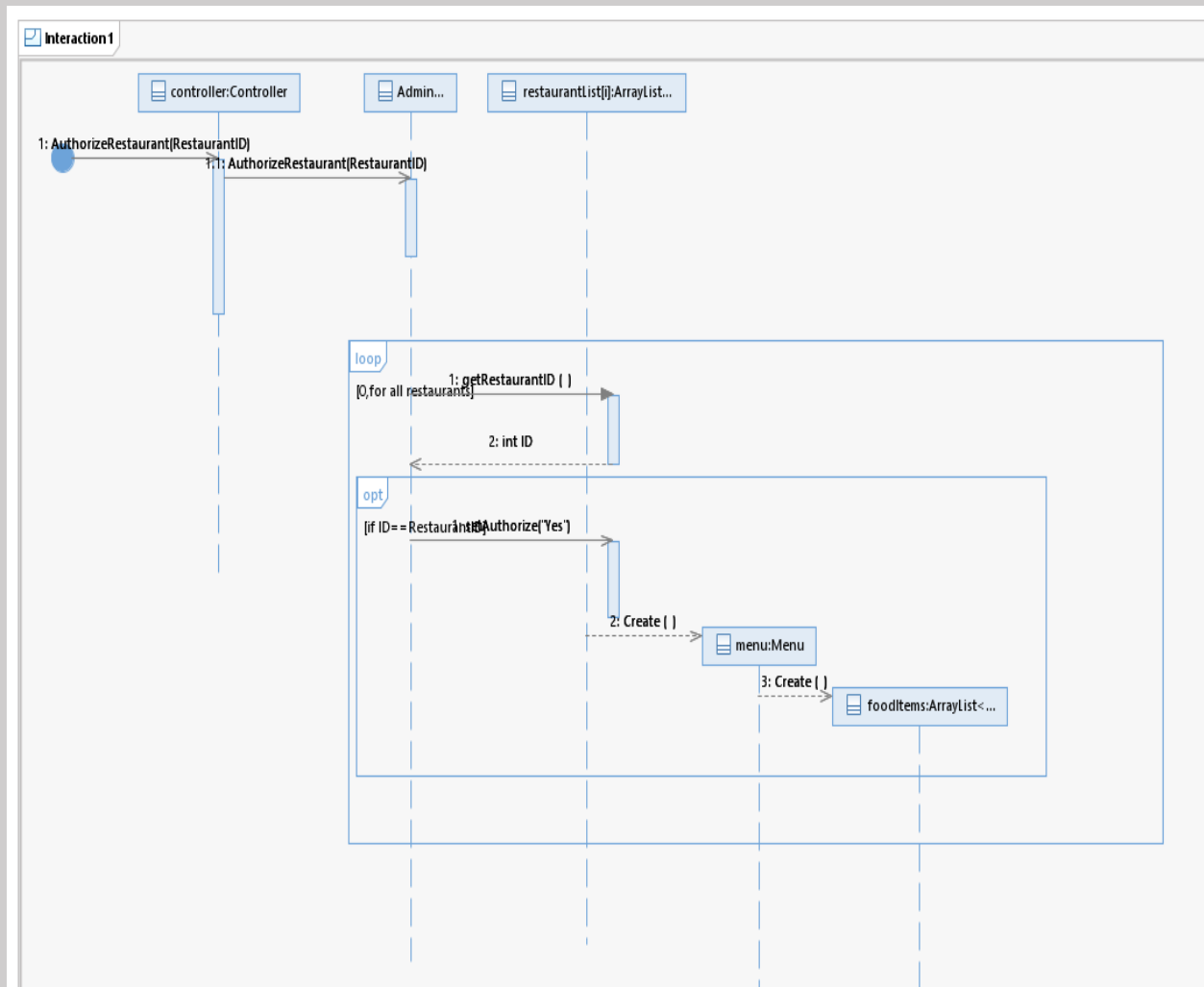


UC04

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Manage Restaurant

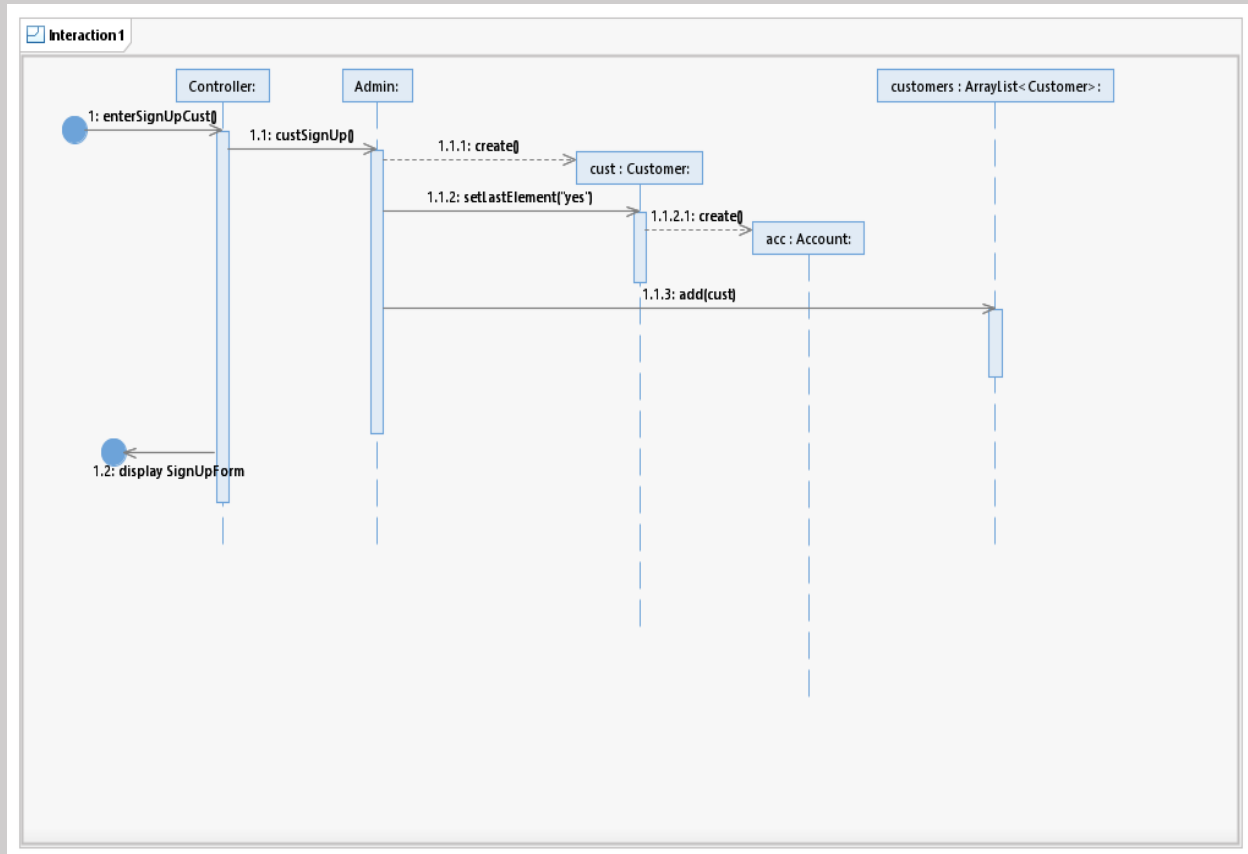


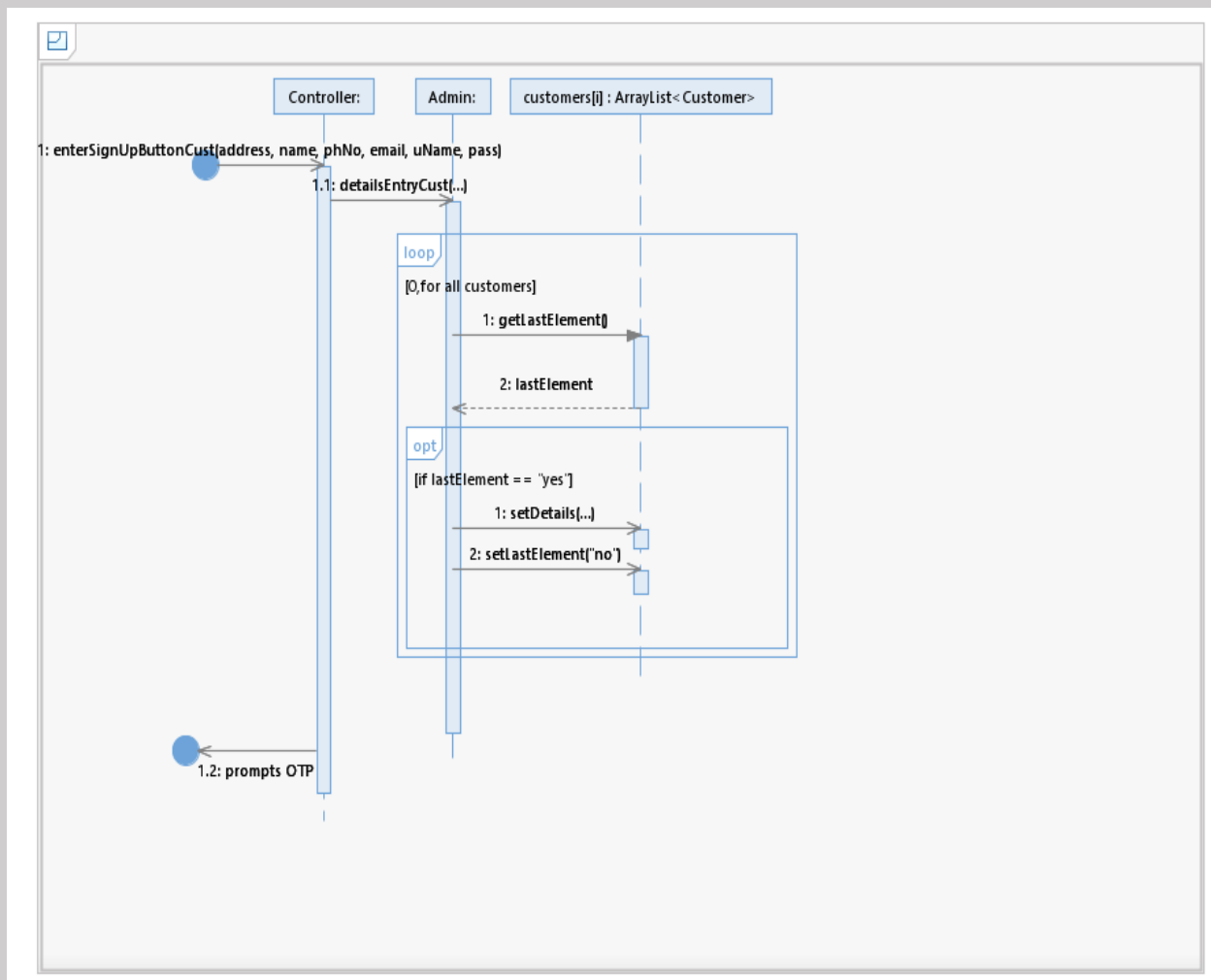


UC05

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User (Customer)

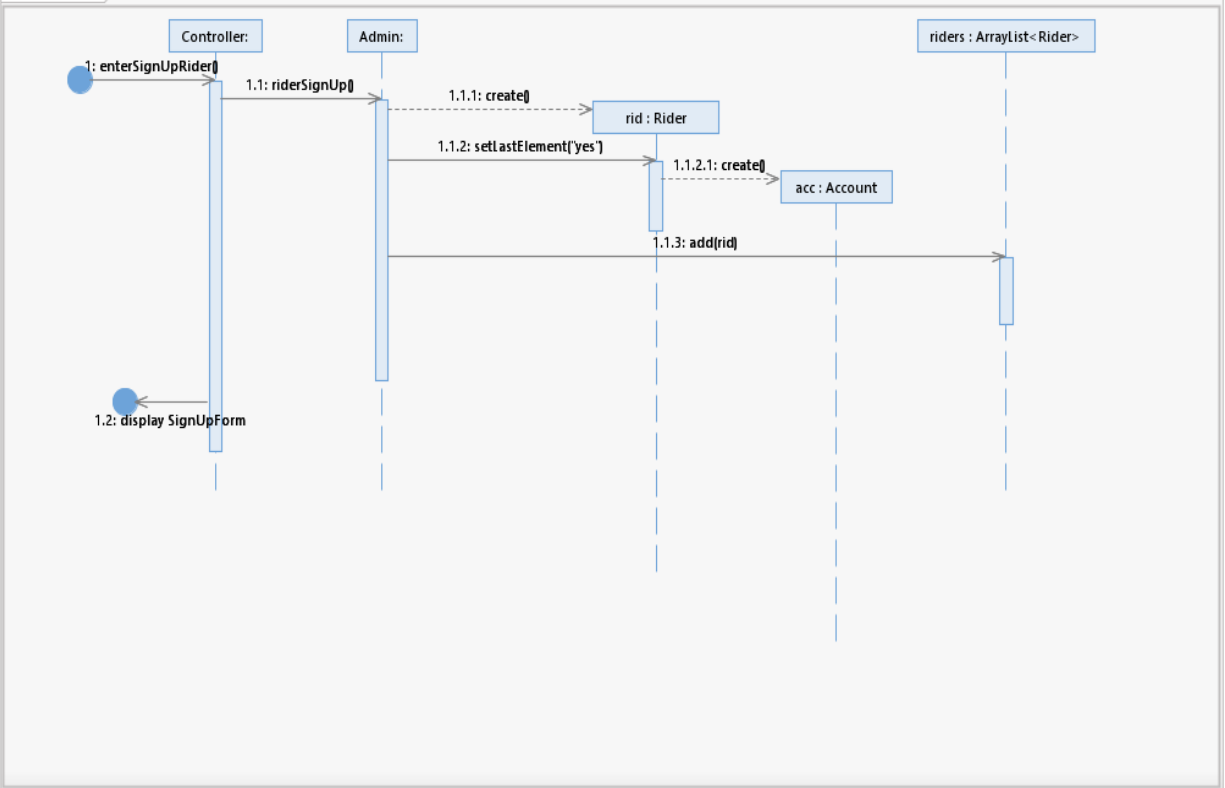


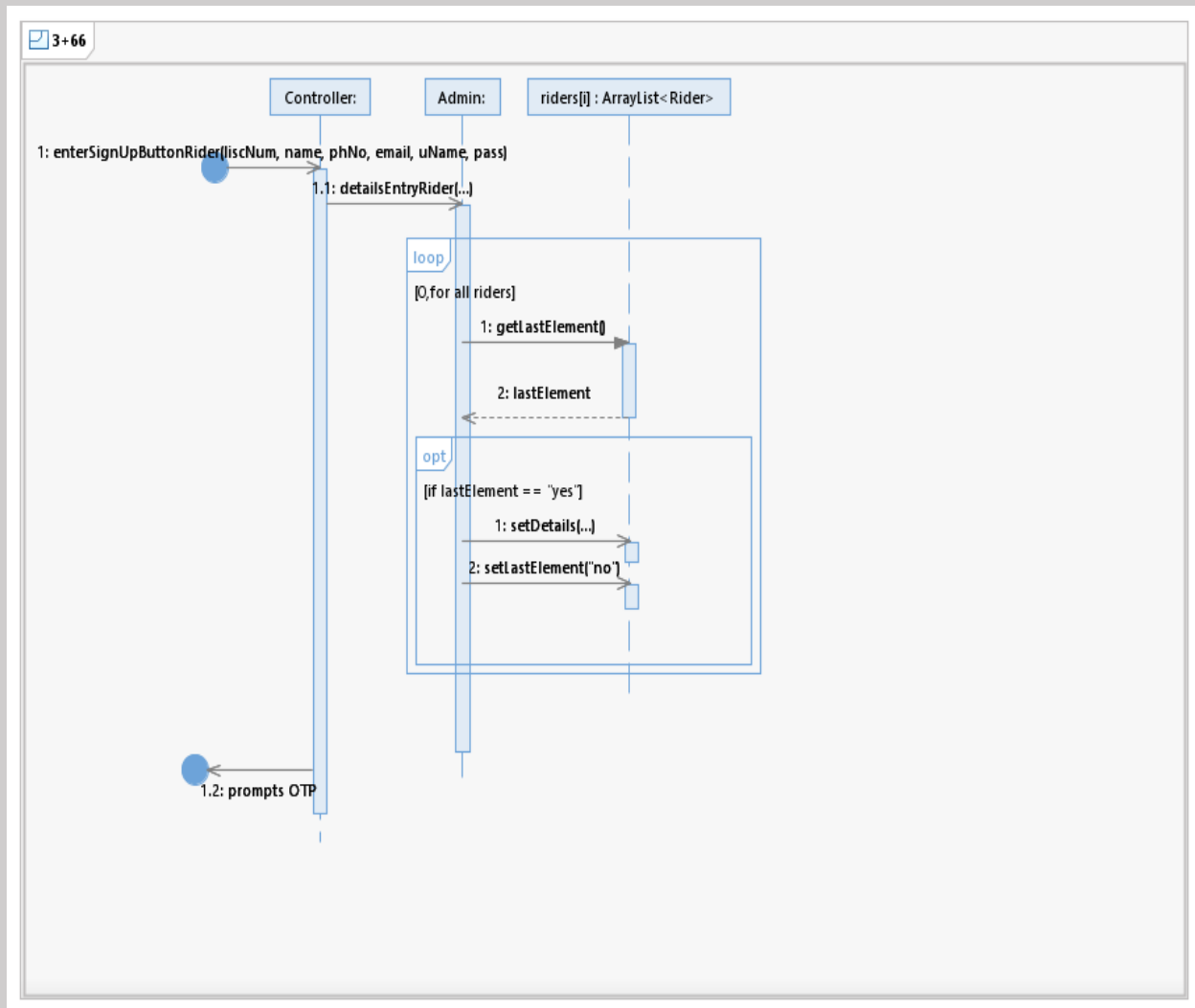


Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User (rider)

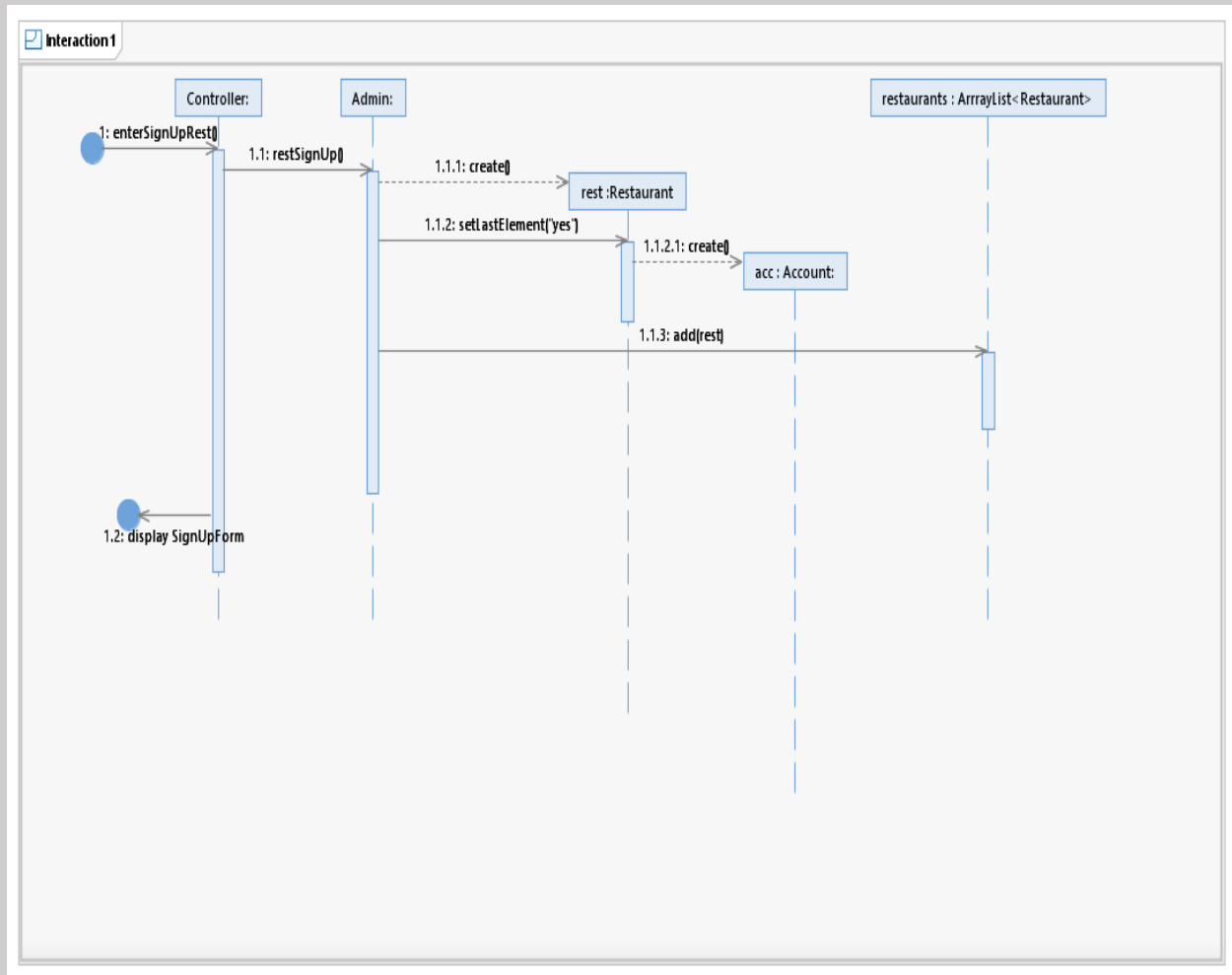
Interaction 1

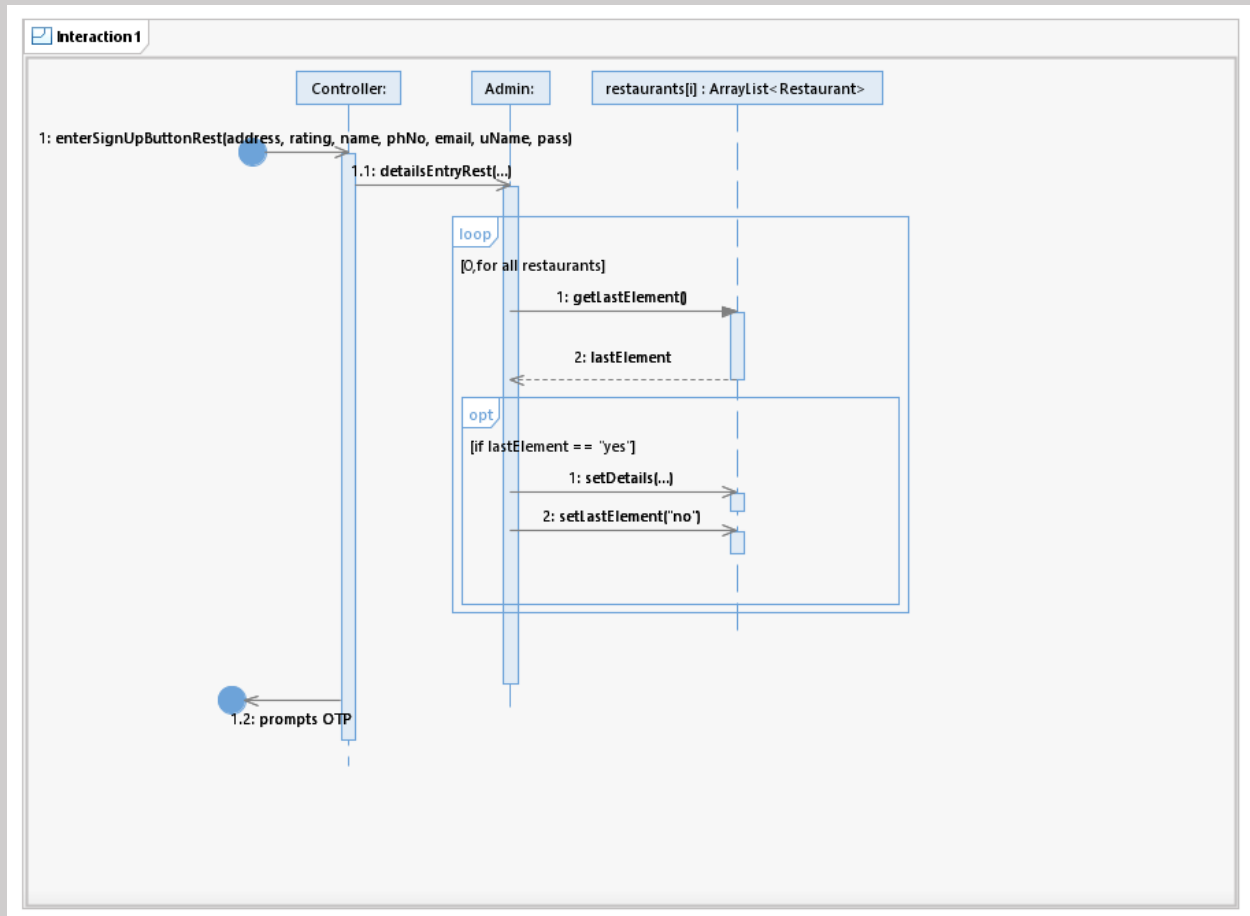




Attempted by: Abdullah Saqib (i200458)

Use Case Name: Register User (restaurant)

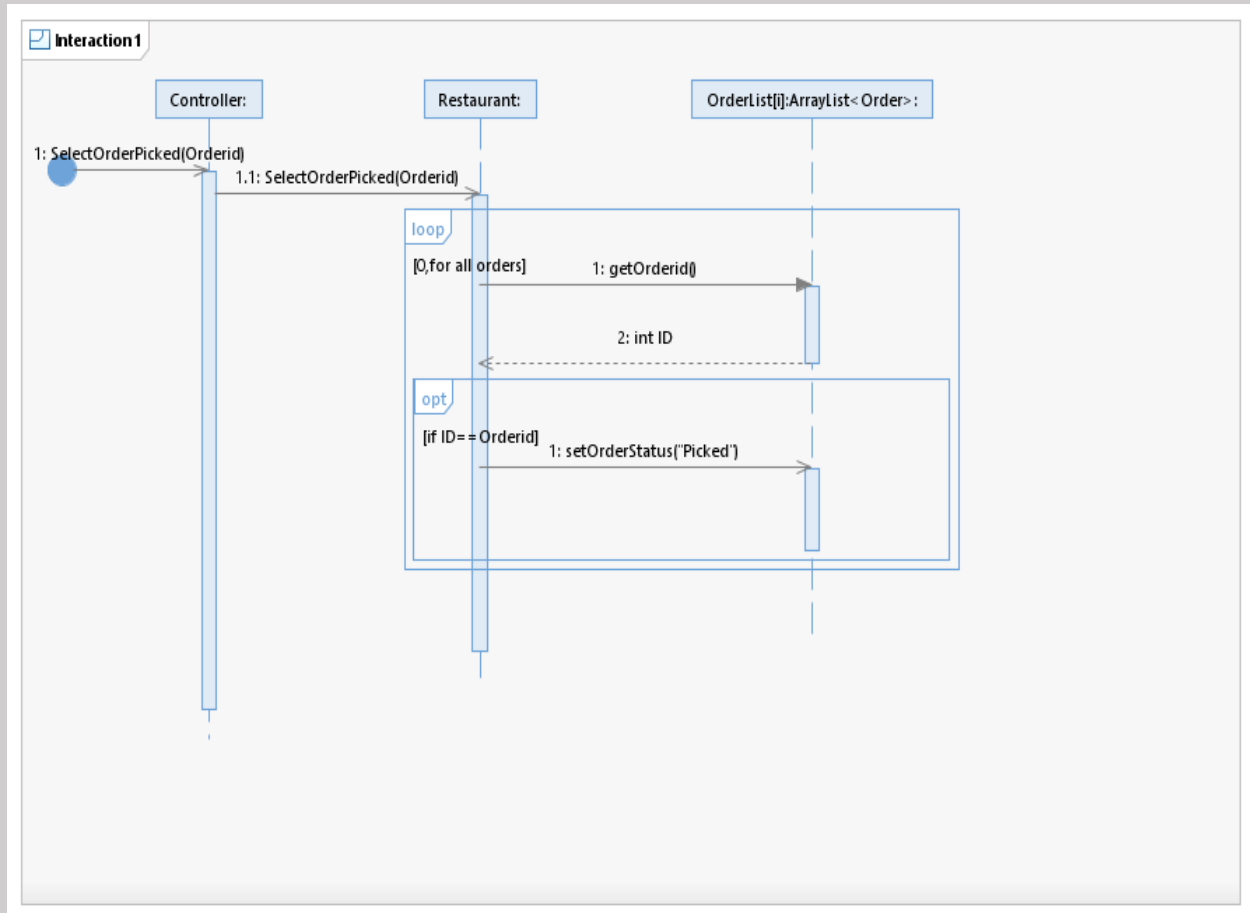




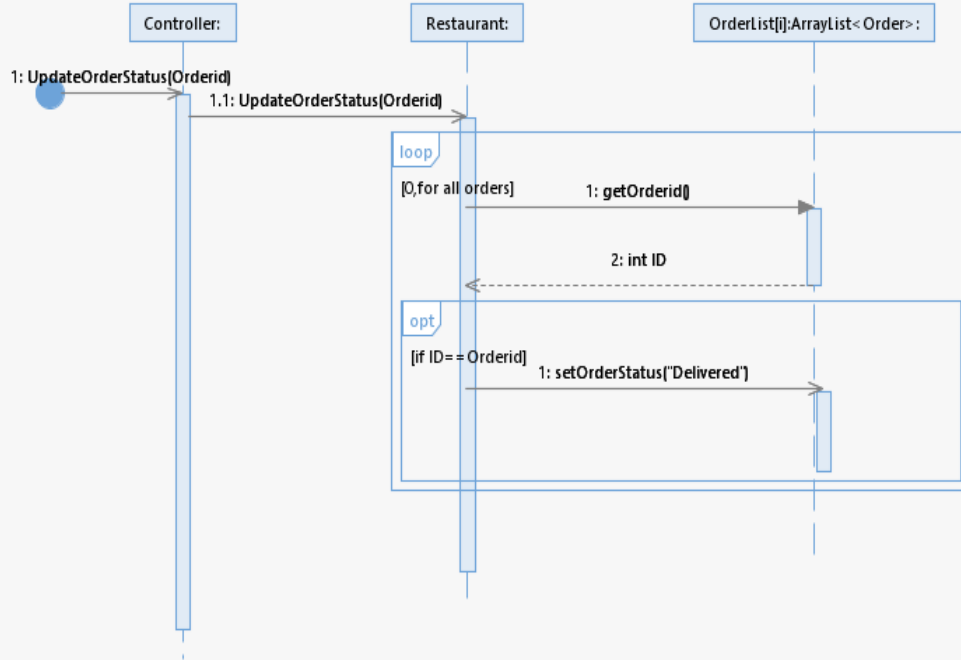
UC06

Attempted by: Rayed Muhammad Saeed (i201822)

Use case name: Accept Order



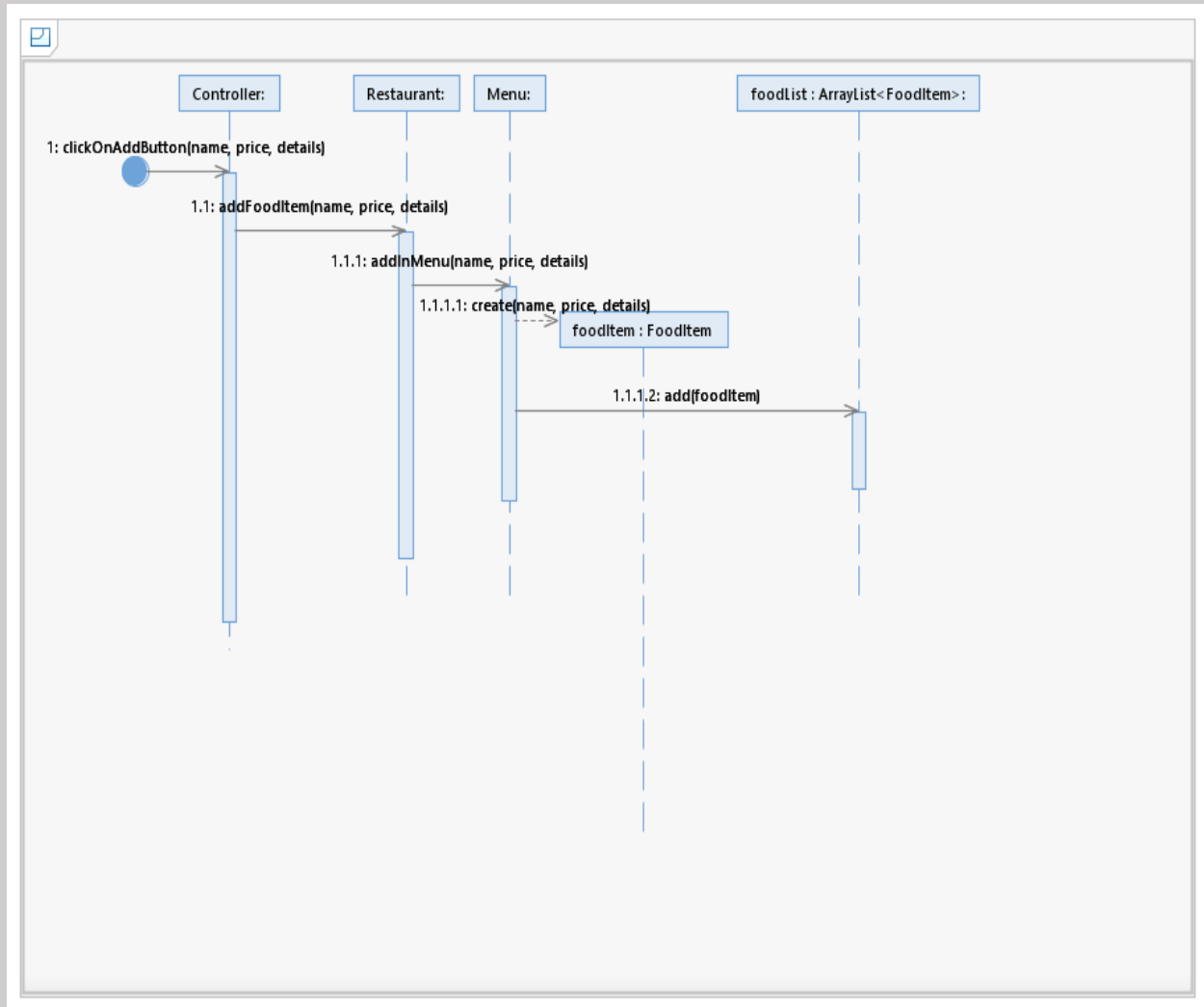
Interaction 1



UC07

Attempted by: Abdullah Saqib (i200458)

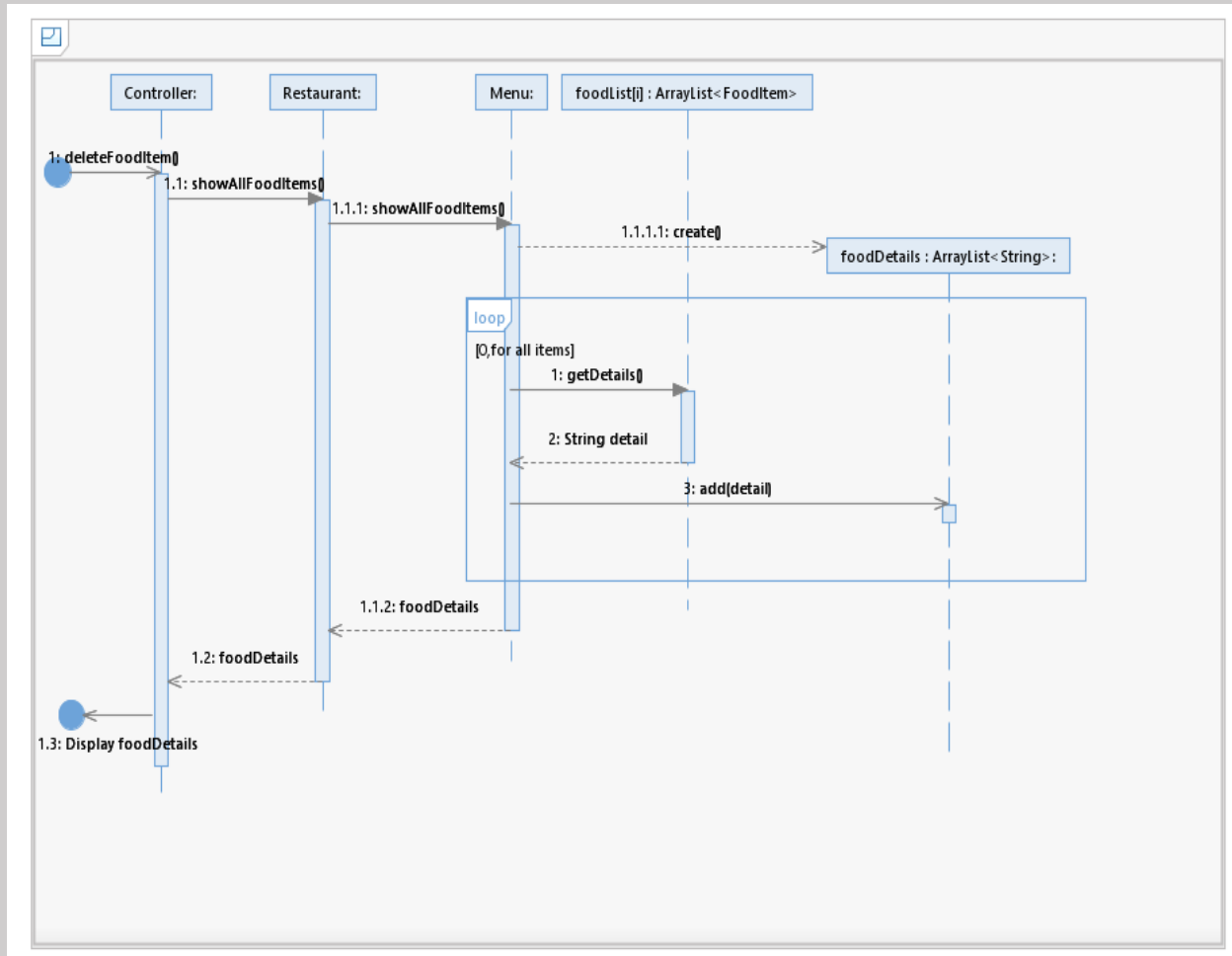
Use Case Name: Add food item

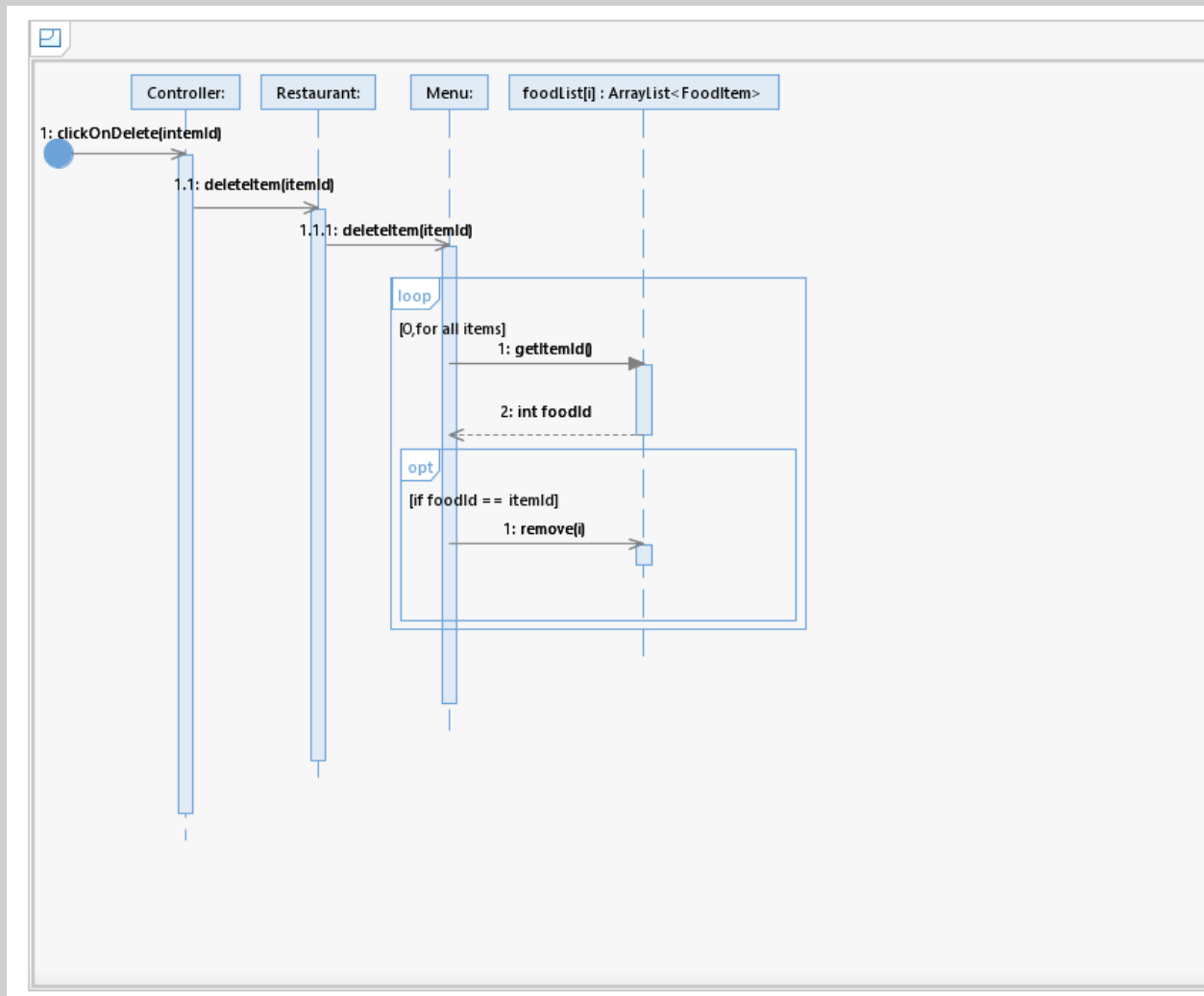


UC08

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Delete food item

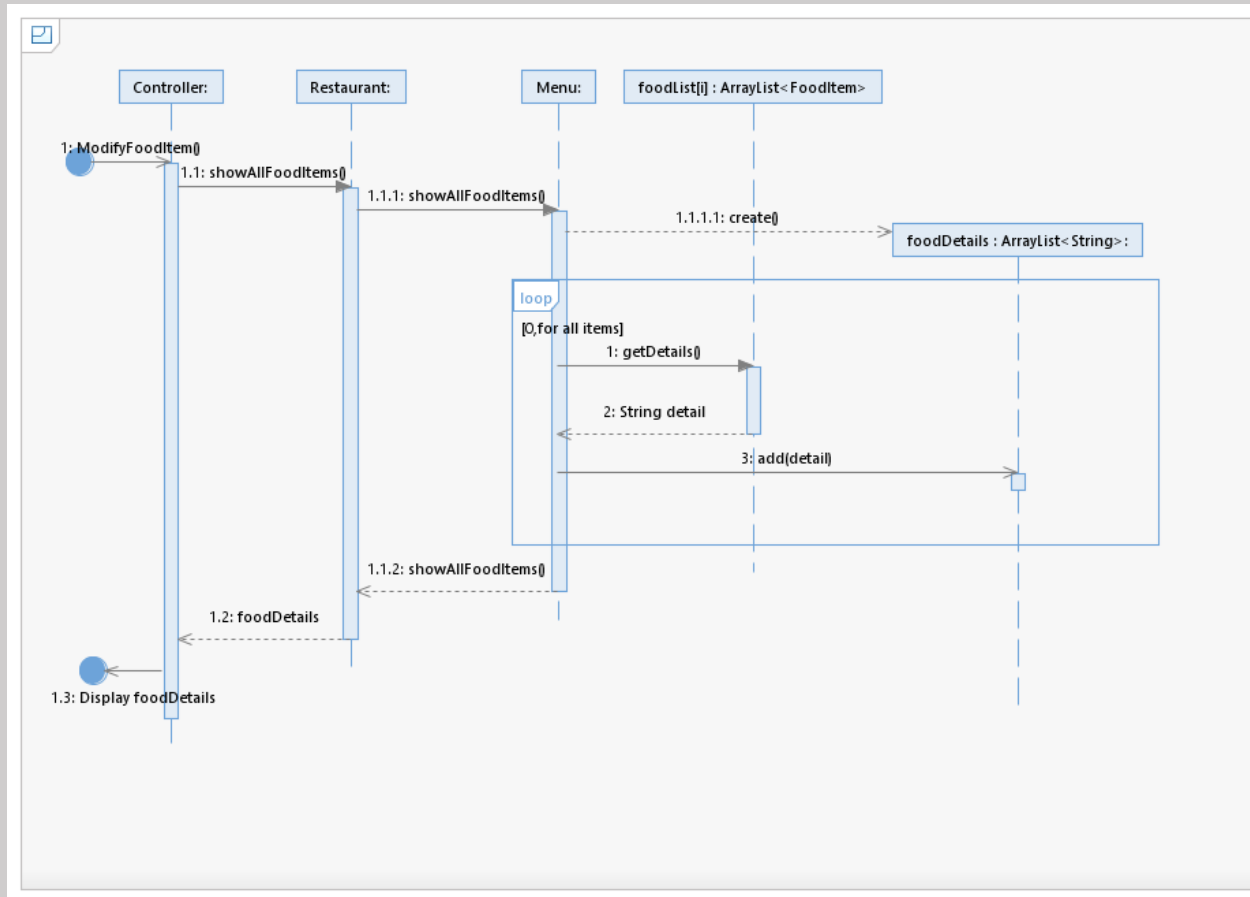


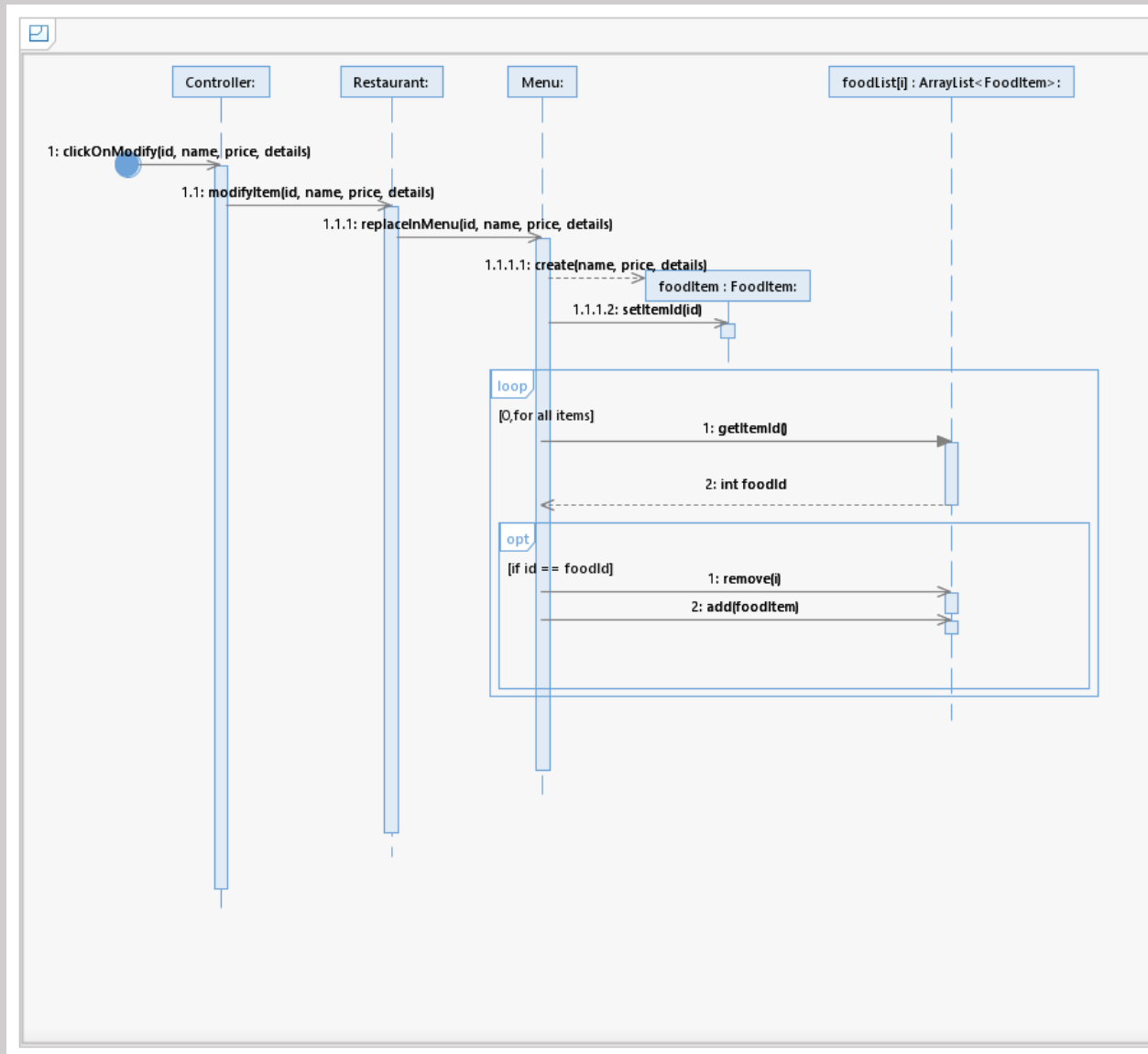


UC09

Attempted by: Abdullah Saqib (i200458)

Use Case Name: Modify food item

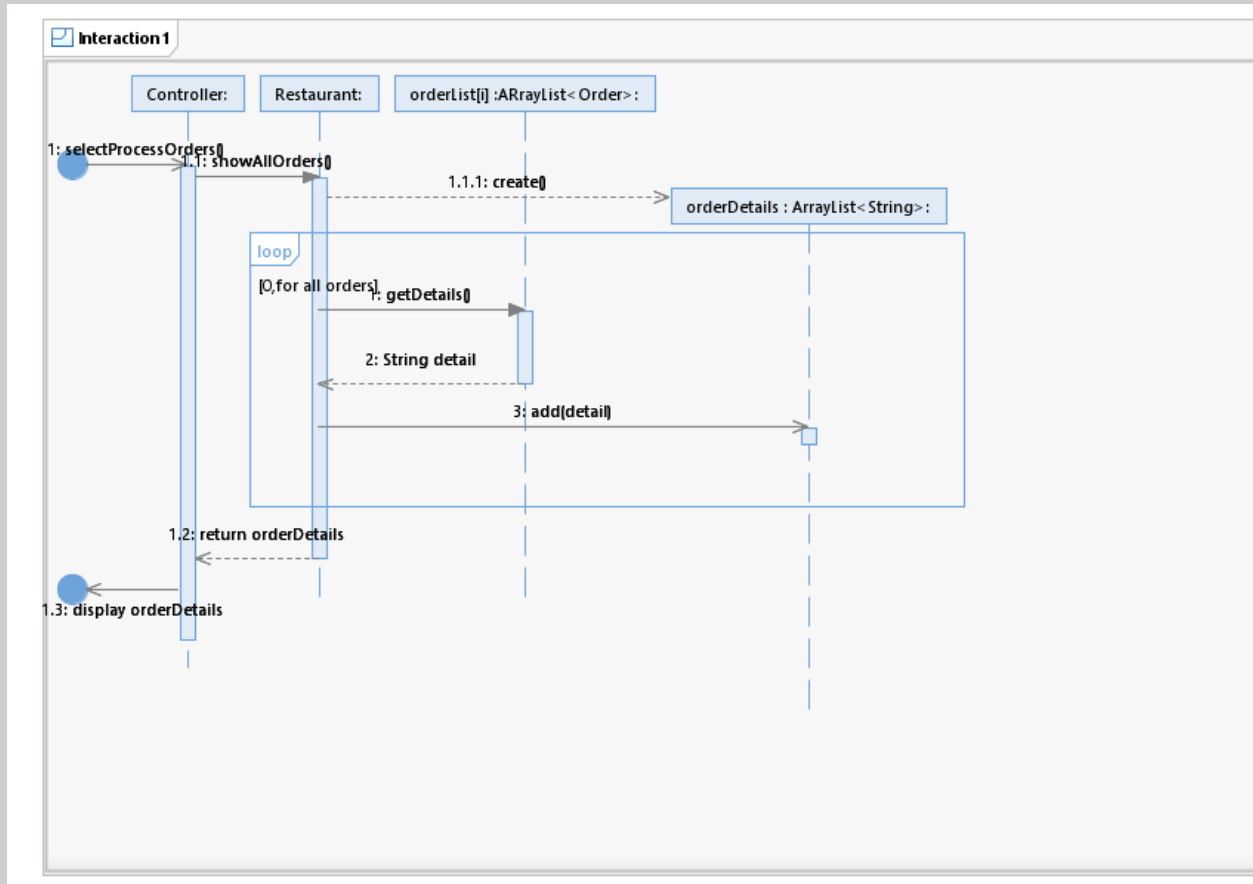




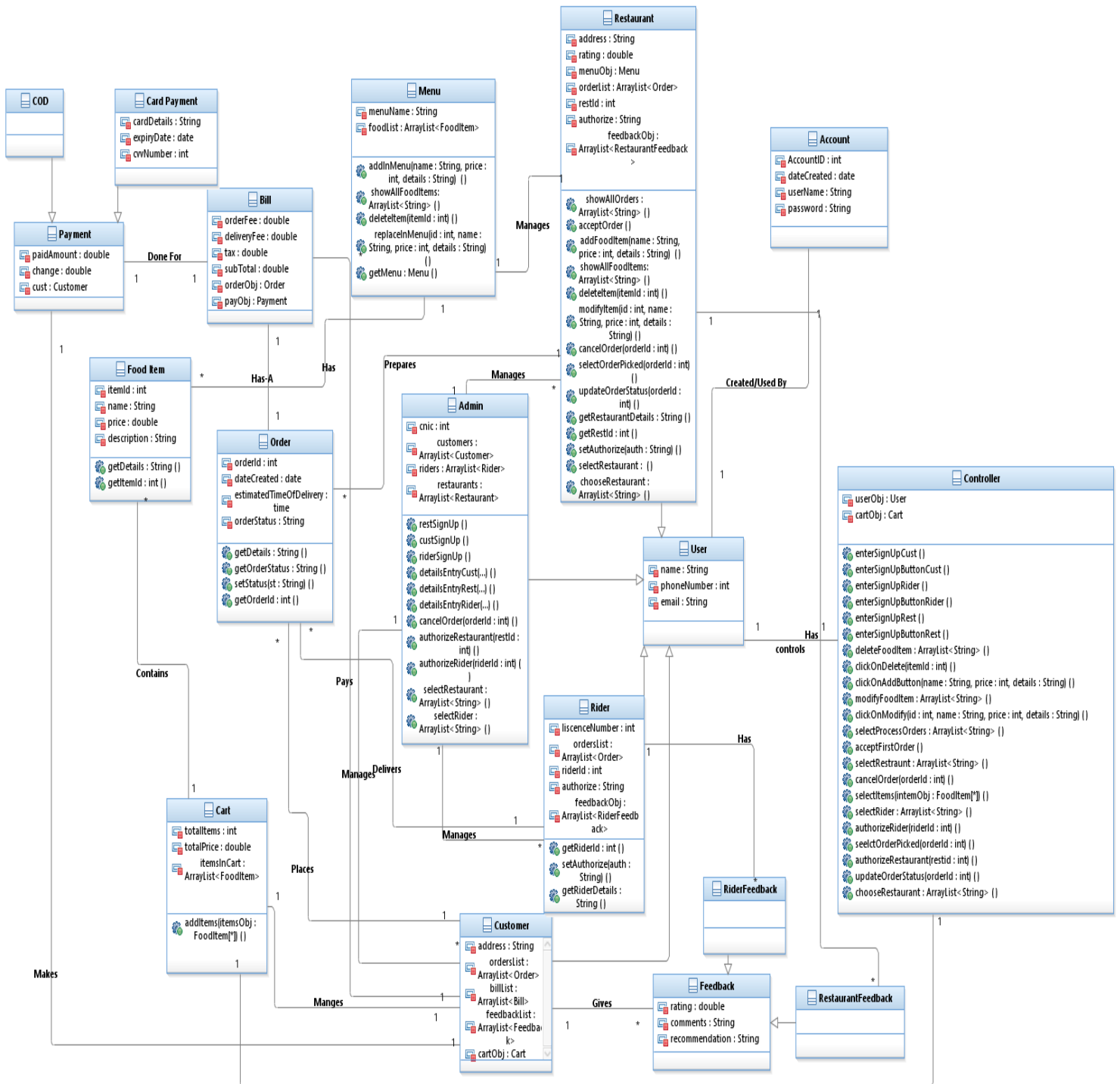
UC10

Attempted by: Abdullah Saqib (i200458)

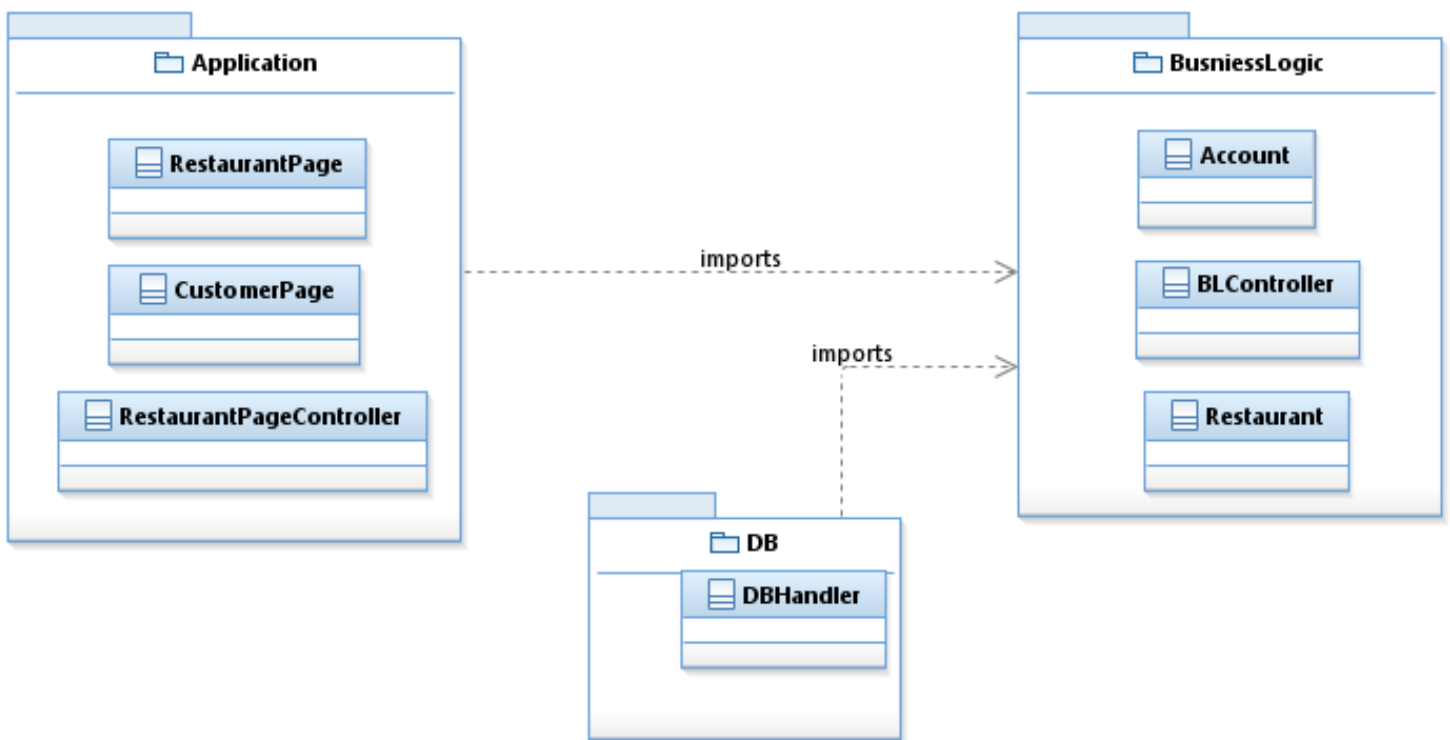
Use Case Name: Processing Order



7. Class Diagram



8. Package Diagram



9. Deployment Diagram