SECP3204: Software Engineering WBL

System Design Descriptions (SDD)

Project Title: Food Ordering System at Arked Angkasa, UTM

Version 1.0

Date: 10 June 2023

School of Computing, Faculty of Engineering

Prepared by: Beta

Table of Contents

3. System Architectural Design

3.1 Architecture Style and Rationale

The Client-Server architecture involves dividing the system into two primary components: the client and the server. In this architecture, clients (such as users accessing the system) send requests to the server, which processes these requests and sends back the appropriate responses. This style is commonly used in web-based systems and is well-suited for distributed environments.

Rationale for choosing the Client-Server architecture:

1. **Separation of concerns**: The Client-Server architecture allows for a clear separation of concerns between the client-side and server-side components. The client is responsible for providing a user-friendly interface and handling user interactions, while the server handles data processing, business logic, and storage. This separation promotes modularity and maintainability.
2. **Scalability**: With a client-server architecture, it is relatively easier to scale the system by adding more servers to handle increased client load. This is particularly important for a food ordering system that may experience varying levels of user demand at different times.
3. **Security**: The Client-Server architecture enables centralized security measures. By centralizing data storage and processing on the server-side, it becomes easier to implement and enforce security measures such as authentication, authorization, and data encryption. This helps protect sensitive user information and ensures secure transactions.
4. **Concurrent access**: The architecture supports concurrent access from multiple clients. Since the food ordering system is likely to have multiple users accessing it simultaneously, the client-server architecture facilitates handling multiple requests concurrently without impacting the system's performance.

3.2 Component Model

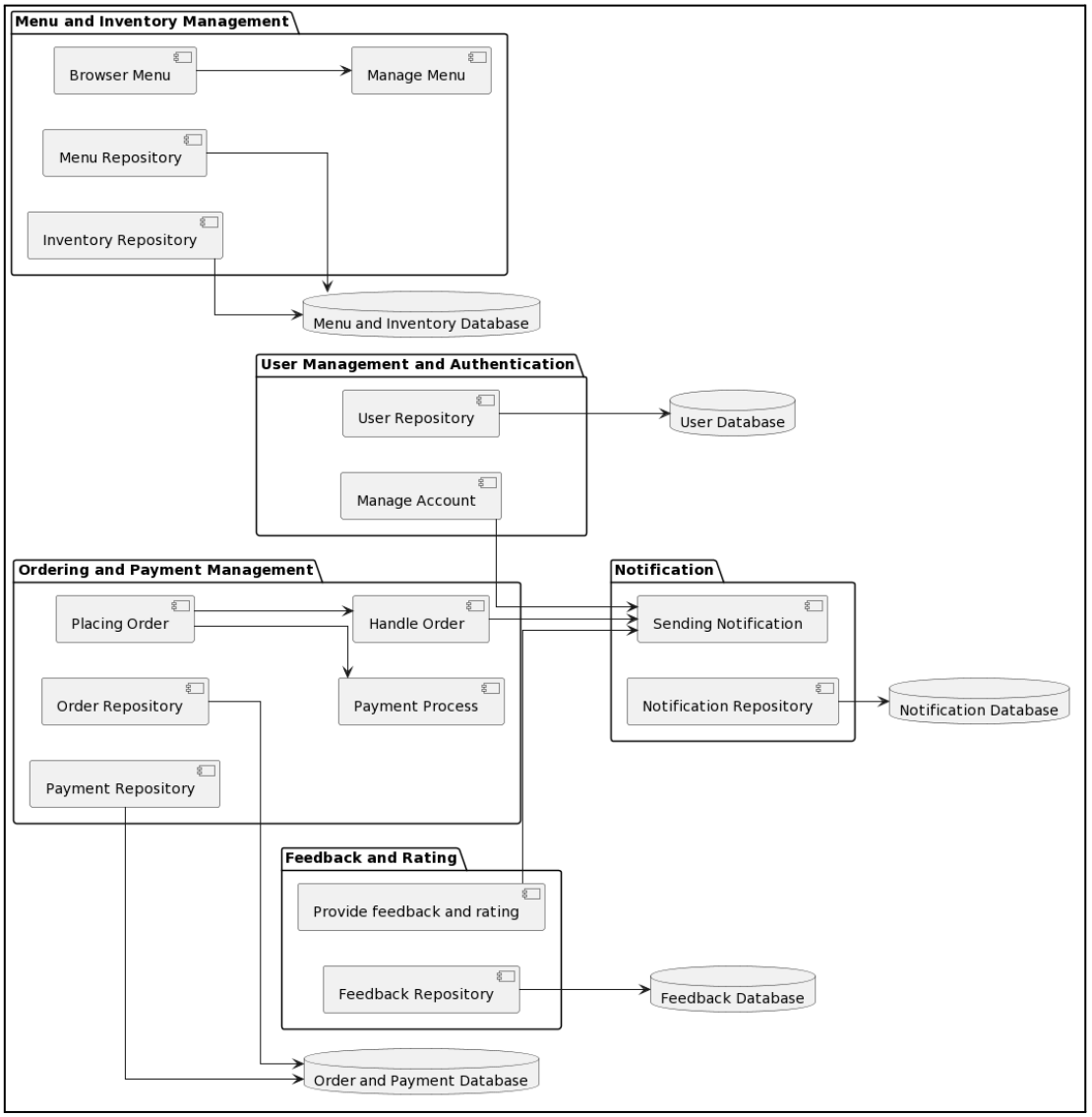


Figure 3.1: Component Diagram of <Food Ordering System at Arked Angkasa, UTM>

The diagram represents the major subsystems and data repositories of the Food Ordering System at Arked Angkasa, UTM.

1. User Management and Authentication:
    - The "User Management and Authentication" subsystem includes the "Manage Account" component responsible for managing user accounts.
    - The "User Repository" represents the data repository where user-related information is stored.
2. Notification:
    - The "Notification" subsystem consists of the "Sending Notification" component responsible for sending notifications to users.

- The "Notification Repository" represents the data repository where notification-related data is stored.

3. Feedback and Rating:
    - The "Feedback and Rating" subsystem includes the "Provide feedback and rating" component, allowing users to provide feedback and ratings.
    - The "Feedback Repository" represents the data repository where feedback and rating data is stored.
4. Menu and Inventory Management:
    - The "Menu and Inventory Management" subsystem consists of the "Manage Menu" and "Browser Menu" components responsible for managing and browsing the menu, respectively.
    - The "Menu Repository" represents the data repository where menu-related information is stored.
    - The "Inventory Repository" represents the data repository where inventory-related data is stored.
5. Ordering and Payment Management:
    - The "Ordering and Payment Management" subsystem includes the "Placing Order," "Payment Process," and "Handle Order" components responsible for order placement, payment processing, and order handling, respectively.
    - The "Order Repository" represents the data repository where order-related information is stored.
    - The "Payment Repository" represents the data repository where payment-related information is stored.

Additionally:

- The arrows indicate the interactions and connections between the components and data repositories.
- The "User Database" represents the data storage for user-related information.
- The "Notification Database" represents the data storage for notification-related information.
- The "Feedback Database" represents the data storage for feedback and rating information.
- The "Menu and Inventory Database" represents the data storage for menu and inventory information.
- The "Order and Payment Database" represents the data storage for order and payment information.

4. Detailed Description of Components

4.1 Complete Package Diagram

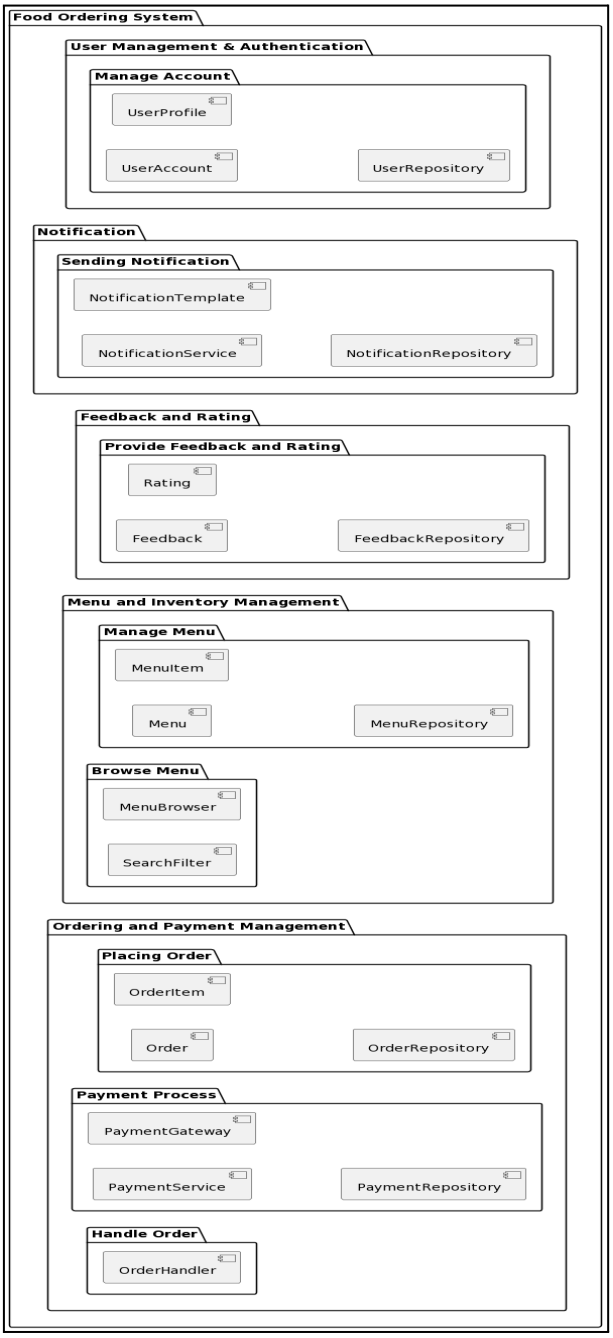


Figure 4.1: Package Diagram for <Food Ordering System at Arked Angkasa, UTM>

4.2 Detailed Description

4.2.1 P001: <Manage Account> Subsystem

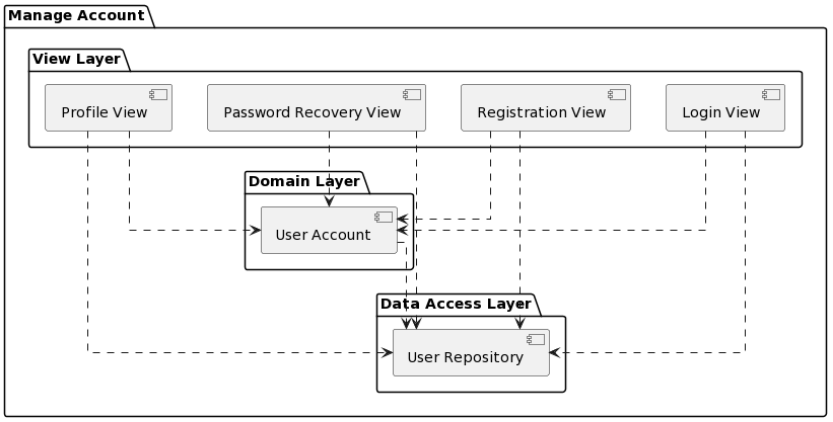


Figure 4.2: Package Diagram for <Manage Account> Subsystem

4.2.1.1 Class Diagram

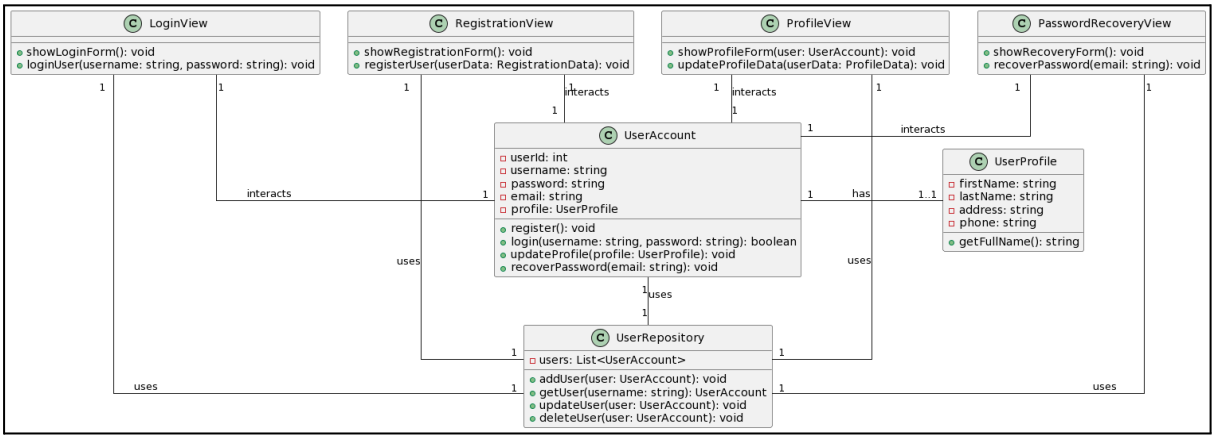


Figure 4.3: Class Diagram for <Manage Account> Subsystem

Entity Name	UserAccount
Method Name	registerAccount
Input	None
Output	None
Algorithm	This method handles the registration process for a user. It collects user information such as username, password, and email, and creates a new user account in the system.

Entity Name	UserAccount
Method Name	login
Input	Username and password strings
Output	Boolean indicating login success or failure
Algorithm	This method verifies the provided username and password against the stored user account information. It returns true if the login is successful, and false otherwise.

Entity Name	UserAccount
Method Name	updateProfile
Input	UserProfile object representing the updated profile information
Output	None
Algorithm	This method updates the profile information of the user account with the provided UserProfile object.

Entity Name	UserAccount
Method Name	recoverPassword
Input	Email address of the user
Output	None
Algorithm	This method initiates the password recovery process for a user account associated with the provided email address. It may involve sending a password reset link or instructions to the user's email.

Entity Name	UserProfile
Method Name	getFullName
Input	None
Output	Full name as a string
Algorithm	This method retrieves the full name of the user from the UserProfile object and returns it as a string.

Entity Name	UserRepository
Method Name	addUser
Input	UserAccount object representing the user account to be added
Output	None
Algorithm	This method adds a new UserAccount object to the repository's list of user accounts.

Entity Name	UserRepository
Method Name	getUser
Input	Username string
Output	UserAccount object
Algorithm	This method retrieves a UserAccount object from the repository based on the provided username.

Entity Name	UserRepository
Method Name	updateUser
Input	UserAccount object representing the updated user account information
Output	None
Algorithm	This method updates the user account information in the repository with the provided UserAccount object.

Entity Name	UserRepository
Method Name	deleteUser
Input	UserAccount object representing the user account to be deleted
Output	None
Algorithm	This method removes the provided UserAccount object from the repository.

Entity Name	RegistrationView
Method Name	showRegistration
Input	None
Output	None
Algorithm	This method displays the registration form to the user, allowing them to enter their registration information.

Entity Name	RegistrationView
Method Name	registerUser
Input	RegistrationData object containing the user's registration information
Output	None
Algorithm	This method handles the user's registration data, typically received from the registration form. It creates a UserAccount object based on the provided data and initiates the registration process.

Entity Name	LoginView
Method Name	showLoginForm
Input	None
Output	None
Algorithm	This method displays the login form to the user, allowing them to enter their credentials.

Entity Name	LoginView
Method Name	loginUser
Input	Username and password strings
Output	None
Algorithm	This method handles the user's login data, typically received from the login form. It calls the login method of UserAccount, passing the provided username and password.

Entity Name	ProfileView
Method Name	showProfileForm
Input	UserAccount object representing the user account
Output	None
Algorithm	This method displays the profile form to the user, pre-filled with the user's current profile information.

Entity Name	ProfileView
Method Name	updateProfileData
Input	ProfileData object containing the updated profile information
Output	UserAccount object representing the user account
Algorithm	This method handles the user's updated profile data, typically received from the profile form. It calls the updateProfile method of UserAccount, passing the provided ProfileData object.

Entity Name	PasswordRecoveryView
Method Name	showRecoveryForm
Input	None
Output	None
Algorithm	This method displays the password recovery form to the user, allowing them to enter their email address.

Entity Name	PasswordRecoveryView
Method Name	recoverPassword
Input	Email address string
Output	None
Algorithm	This method initiates the password recovery process by calling the recoverPassword method of UserAccount, passing the provided email address.

4.2.1.2 Sequence Diagram

a) SD001: Sequence diagram for Manage Account as Customer

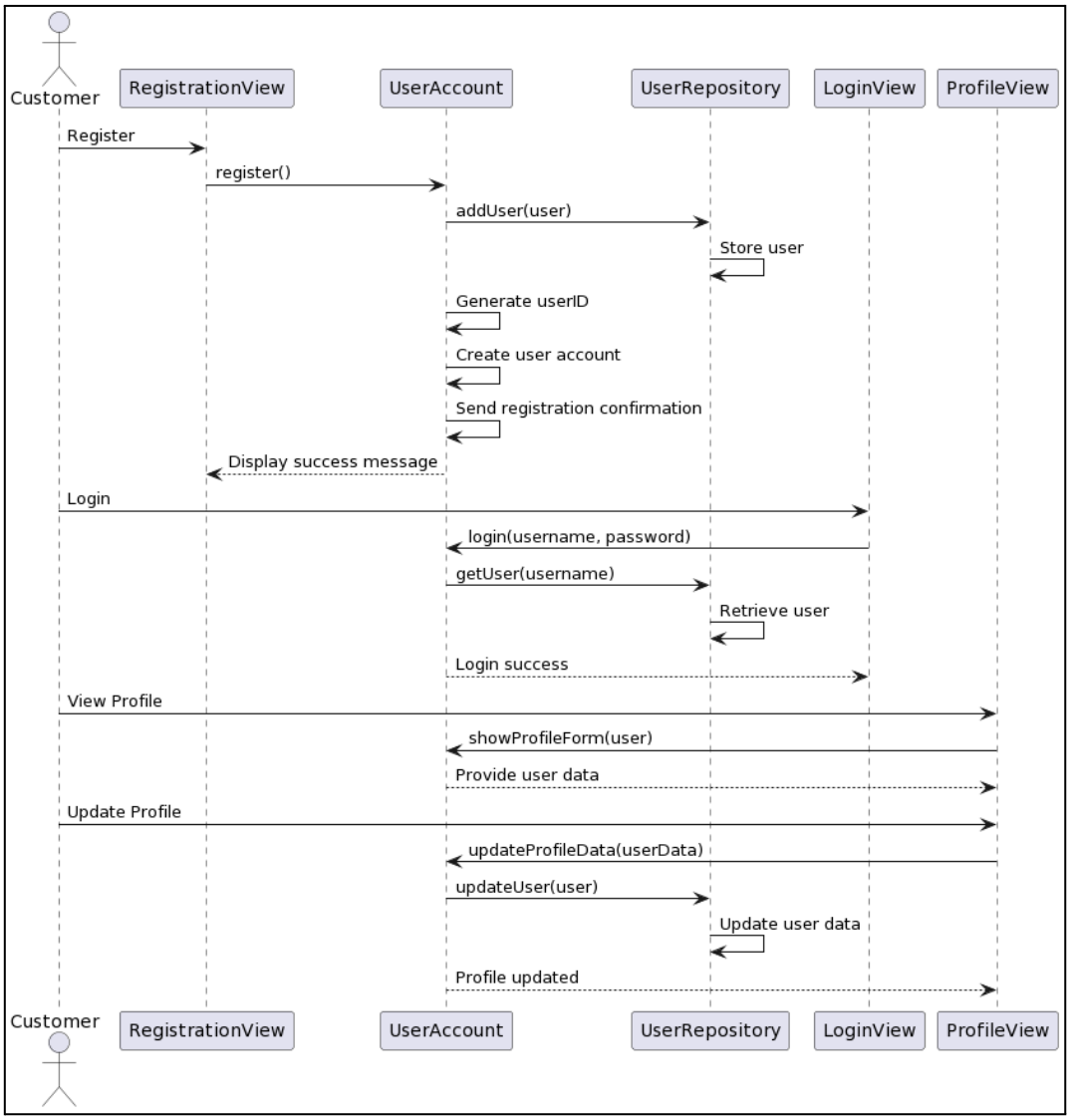


Figure 4.4: Sequence Diagram for <Manage Account Scenario as Customer>

b) SD002: Sequence diagram for Manage Account Scenario as Administrator

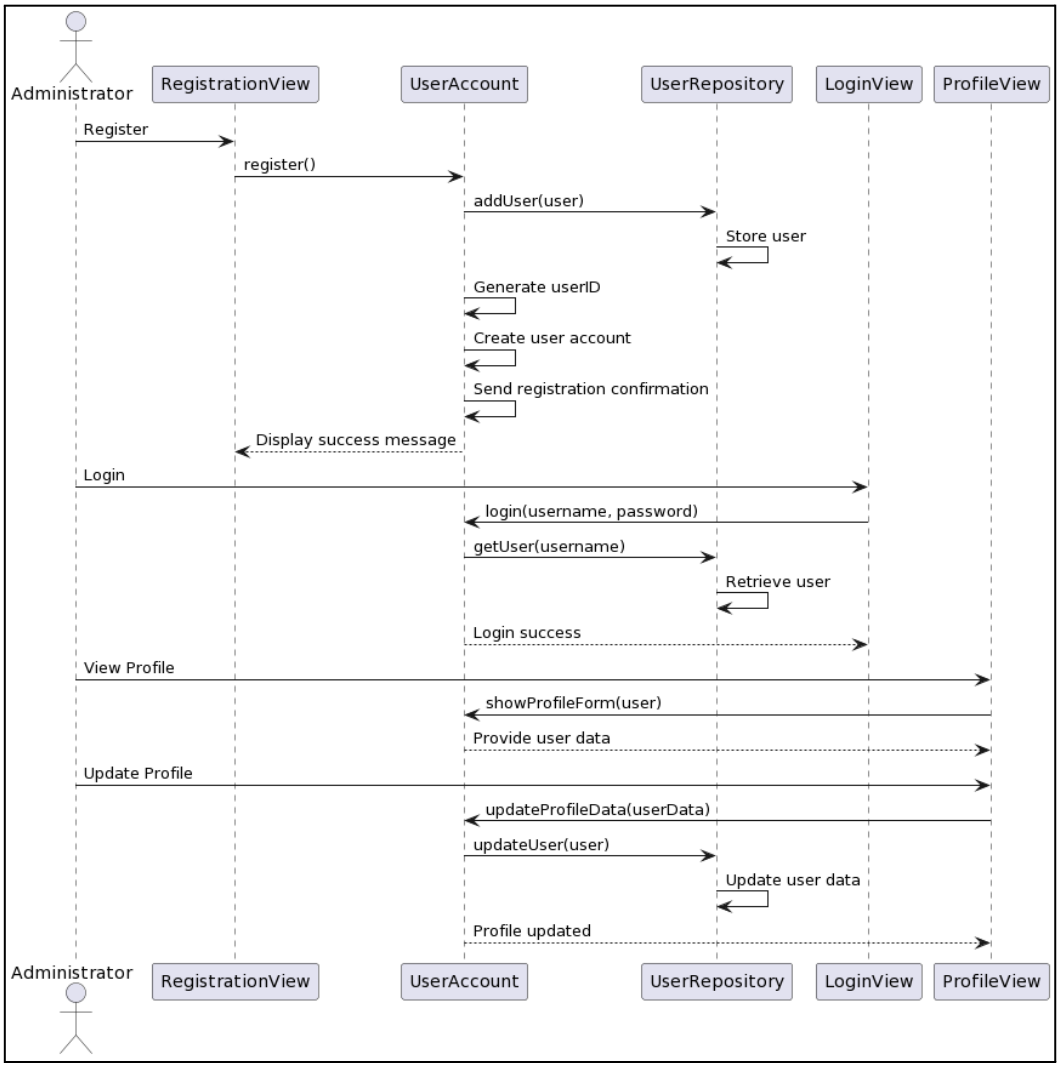


Figure 4.5: Sequence Diagram for <Manage Account Scenario as Administrator>

4.2.2 P002: < Sending notification > Subsystem

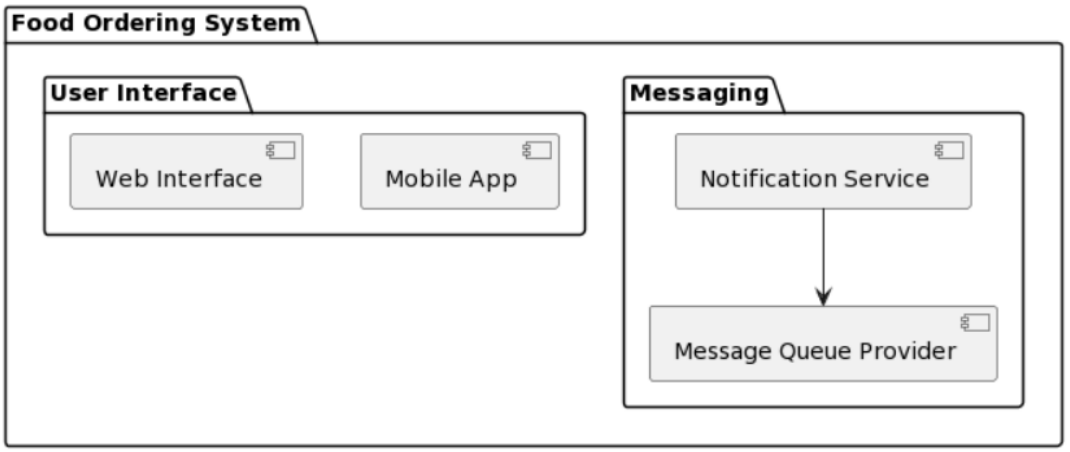


Figure 4.6: Package Diagram for <Sending Notification>

4.2.2.1 Class Diagram

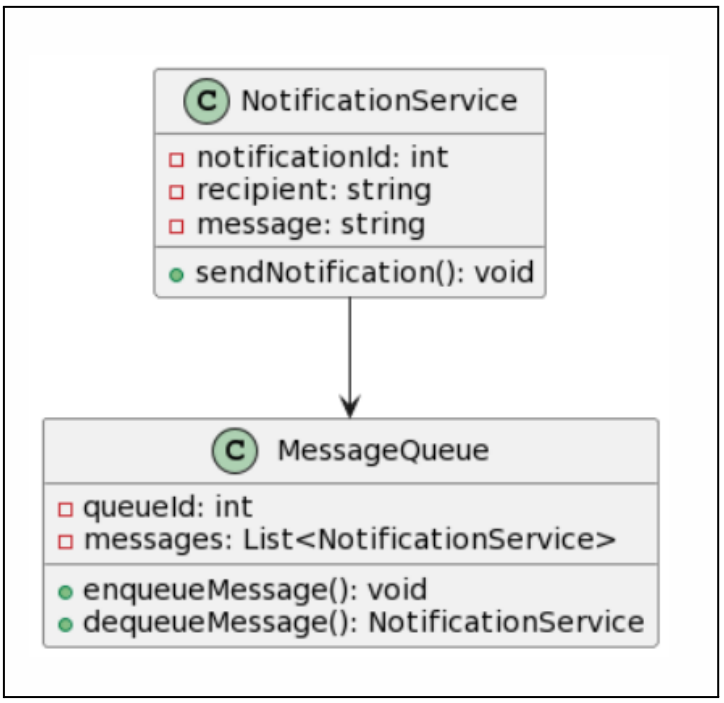


Figure 4.7: Class Diagram for <Sending Notification>

Entity Name	NotificationService
Method Name	NotificationService: - sendNotification()
Input	sendNotification(): - recipient (string): the recipient of the notification - message (string): the content of the notification
Output	sendNotification(): - None (void)
Algorithm	sendNotification(): - Receive recipient and message as input. - Create a new instance of the NotificationService. - Set the recipient and message properties of the notification. - Send the notification to the recipient.

Entity Name	MessageQueue
Method Name	MessageQueue: - enqueueMessage() - dequeueMessage()
Input	enqueueMessage(): - notification (NotificationService): the notification to be added to the message queue dequeueMessage(): - None (no input required)
Output	enqueueMessage(): - None (void) dequeueMessage(): - notification (NotificationService): the next notification in the message queue
Algorithm	enqueueMessage(): 1. Receive the notification as input. 2. Add the notification to the message queue. dequeueMessage(): 1. Retrieve the next notification from the message queue. 2. Return the notification.

4.2.2.2Sequence Diagram

4.2.2.2.1 Sequence diagram scenario order complete

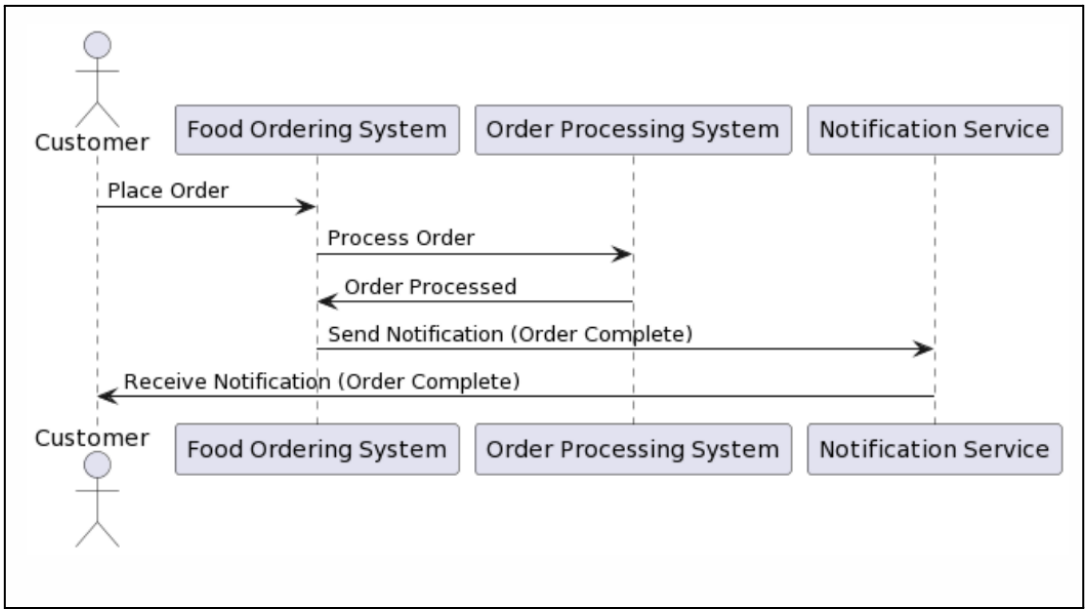


Figure 4.8.1: Class Diagram for <Sending Notification>

4.2.2.2.2 Sequence diagram scenario order accepted

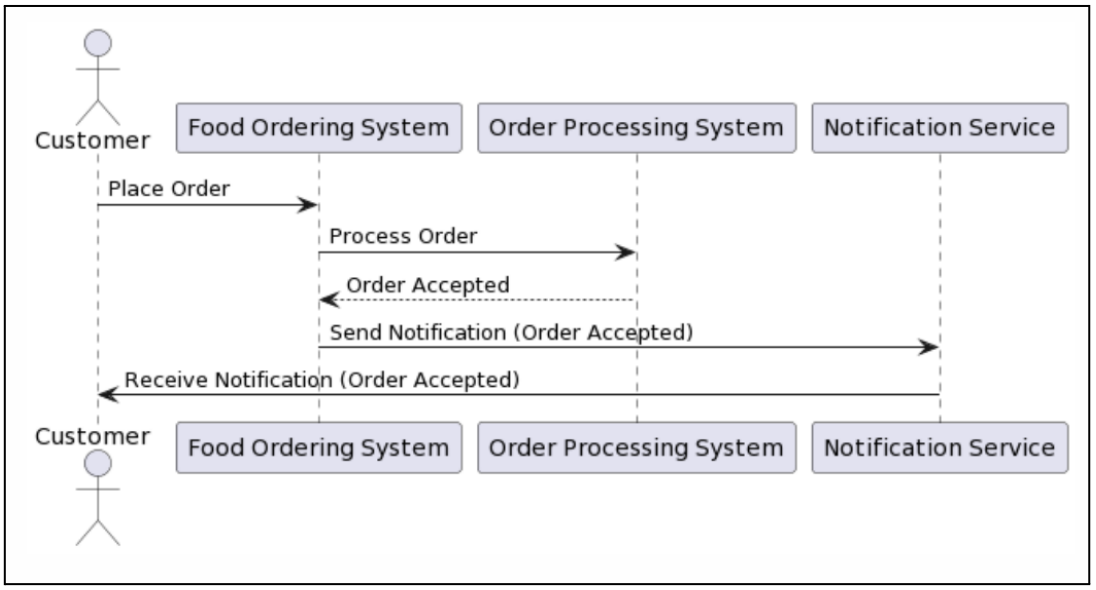


Figure 4.8.2: Sequence Diagram for <Sending Notification>

4.2.3 P003: <Provide Feedback and Rating> Subsystem

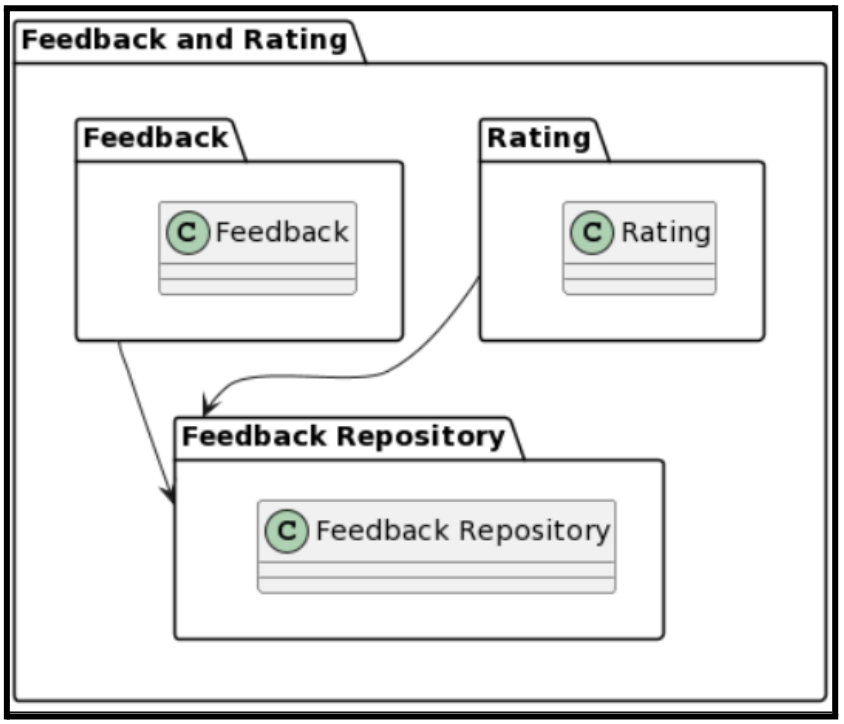


Figure 4.9: Package Diagram for <Feedback and Rating>

4.2.3.1 Class Diagram

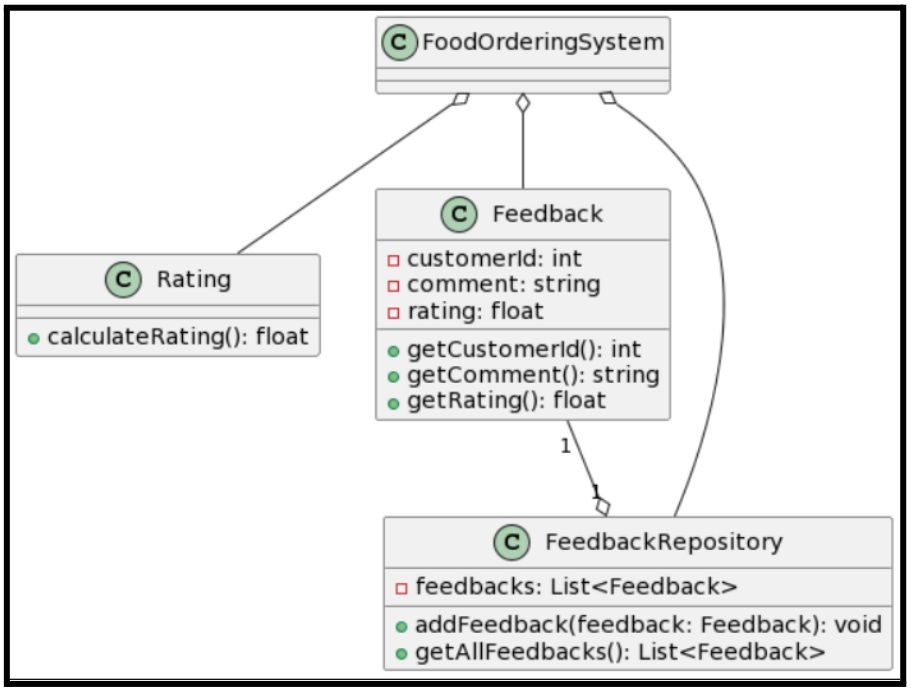


Figure 4.9: Class Diagram for <Feedback and Rating>

Entity Name	Rating
Method Name	calculateRating()
Input	-
Output	Average rating value
Algorithm	-

Entity Name	Feedback
Method Name	getCustomerId()
Input	-
Output	Customer ID
Algorithm	Return the customer ID attribute.

Entity Name	Feedback
Method Name	getComment()
Input	-
Output	Feedback Comment
Algorithm	Return the comment attribute of the feedback.

Entity Name	Feedback
Method Name	getRating()
Input	-
Output	Rating Value
Algorithm	Return the rating attribute of the feedback.

Entity Name	FeedbackRepository
Method Name	addFeedback(feedback)
Input	feedback: Feedback
Output	-
Algorithm	Add the provided feedback object to the list of feedbacks

Entity Name	FeedbackRepository
Method Name	getAllFeedbacks()
Input	-
Output	List of all Feedbacks
Algorithm	Return the list of all feedback stored in the repository.

4.2.3.2 Sequence Diagram

4.2.3.2.1 Sequence Diagram of Scenario Showing Customer Make a Feedback and Rating

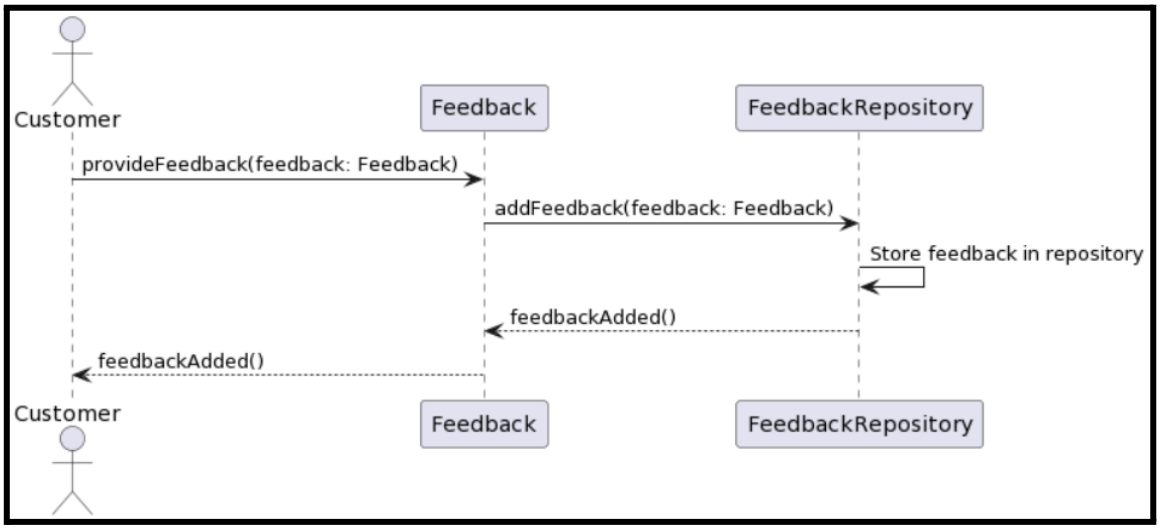


Figure 4.10: Sequence Diagram for <Feedback and Rating as Customer>

4.2.4 P004: <Menu and Inventory Management> Subsystem

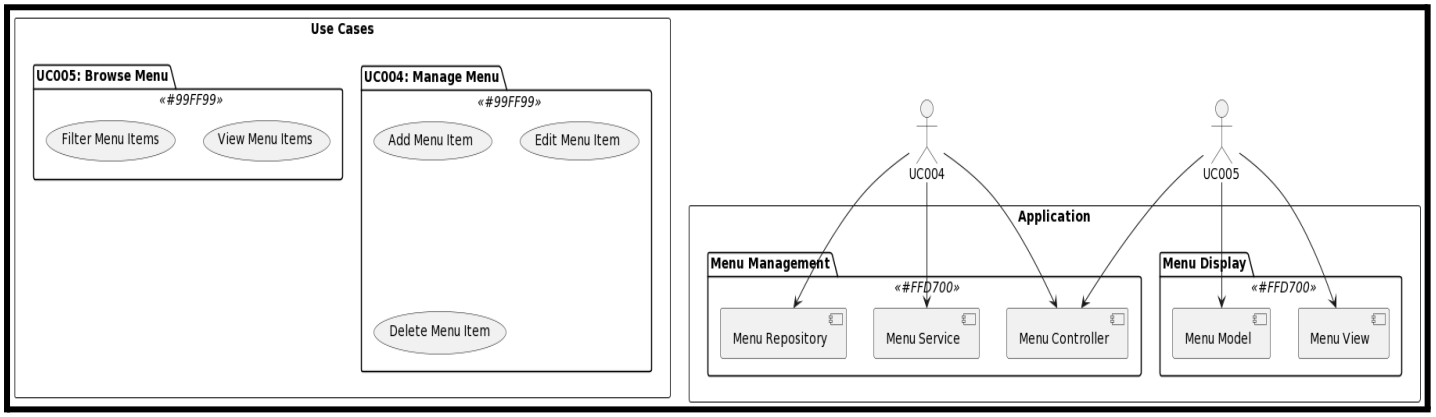


Figure 4.11: Package Diagram for <Menu and Inventory Management> Subsystem

4.2.4.1 Class Diagram

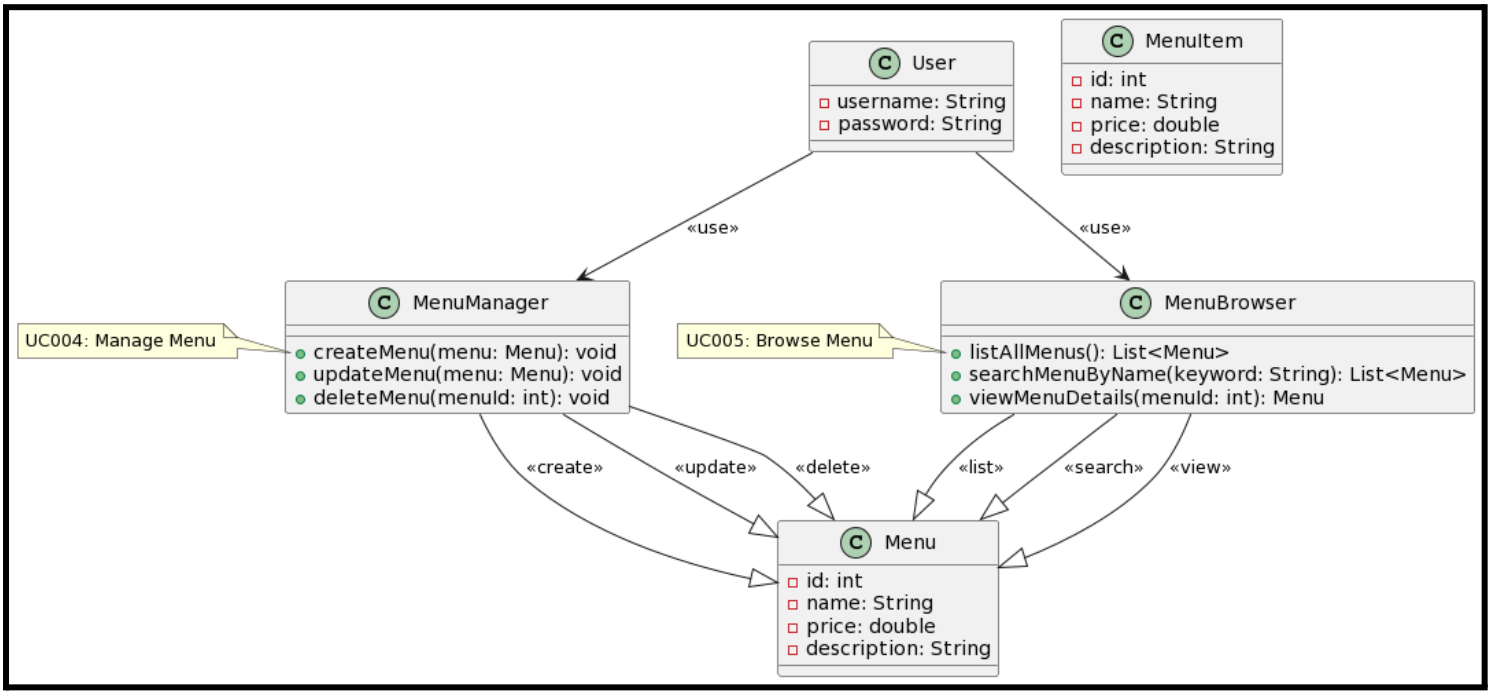


Figure 4.12: Class Diagram for <Menu and Inventory Management > Subsystem

Entity Name	MenuManager
Method Name	MenuManager : ● createMenu ()
Input	createMenu () : ● id (int) ● name (string) ● price (double) ● description (string)
Output	MenuItem
Algorithm	createMenu () : 1. Create a new menu. 2. Put a name on the menu. 3. Set the price on the menu. 4. Give a description about the menu.

Entity Name	MenuManager
Method Name	MenuManager : ● updateMenu ()
Input	updateMenu () : ● id (int) ● name (string) ● price (double) ● description (string)
Output	MenuItem
Algorithm	updateMenu () : 1. Select the existing menu. 2. Update the name on the menu. 3. Change the price on the menu. 4. Modify the description about the menu.

Entity Name	MenuManager
Method Name	MenuManager : ● deleteMenu ()
Input	deleteMenu () : ● id (int) ● name (string) ● price (double) ● description (string)
Output	MenuItem
Algorithm	deleteMenu () : 1. Select the existing menu. 2. delete permanently the menu.

Entity Name	MenuBrowser
Method Name	MenuBrowser : ● listAllMenus ()
Input	None
Output	MenuItem
Algorithm	listAllMenus () 1. All the list of menu shows on the screen.

Entity Name	MenuBrowse
Method Name	MenuBrowser : ● searchMenuByName ()
Input	searchMenuByName () : ● keyword (string)
Output	MenuItem
Algorithm	searchMenuByName () : 1. Click on the search. 2. Insert some keywords from the menu.

Entity Name	MenuBrowse
Method Name	MenuBrowser : ● viewMenuDetails ()
Input	viewMenuDetails () : ● menuid (int)
Output	MenuItem
Algorithm	viewMenuDetails () : 1. Click a menu from the list. 2. Click the button description menu.

4.2.4.2 Sequence Diagram

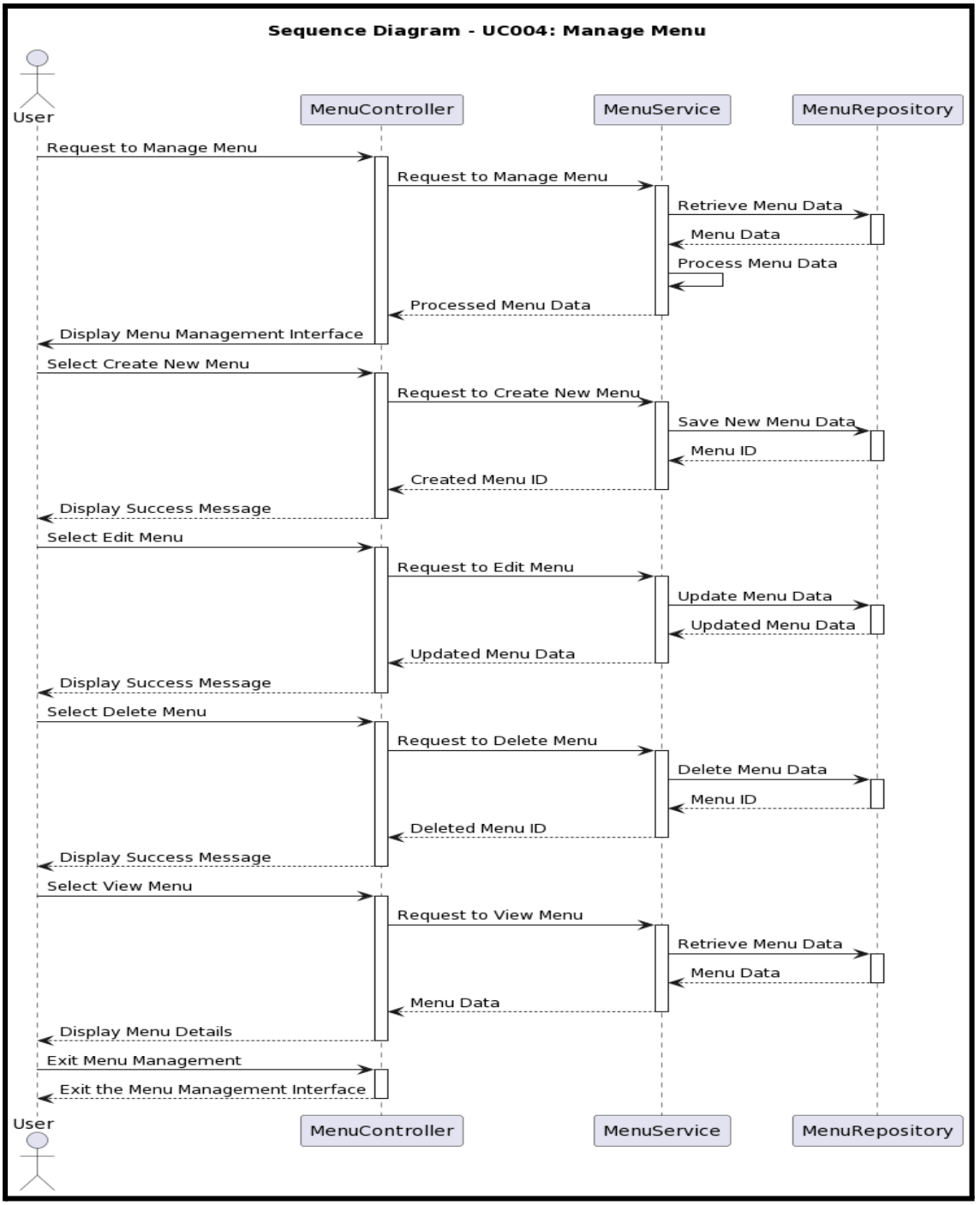


Figure 4.13.1: Sequence Diagram for < Admin Manage Menu >

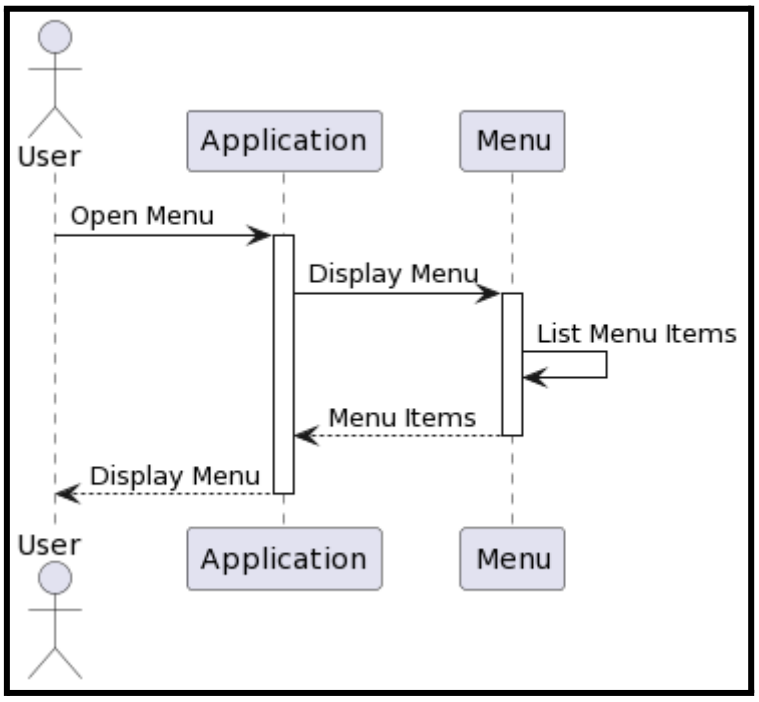


Figure 4.13.2: Sequence Diagram for <User Browse Menu>

4.2.5 P005: <Ordering and Payment Management> Subsystem

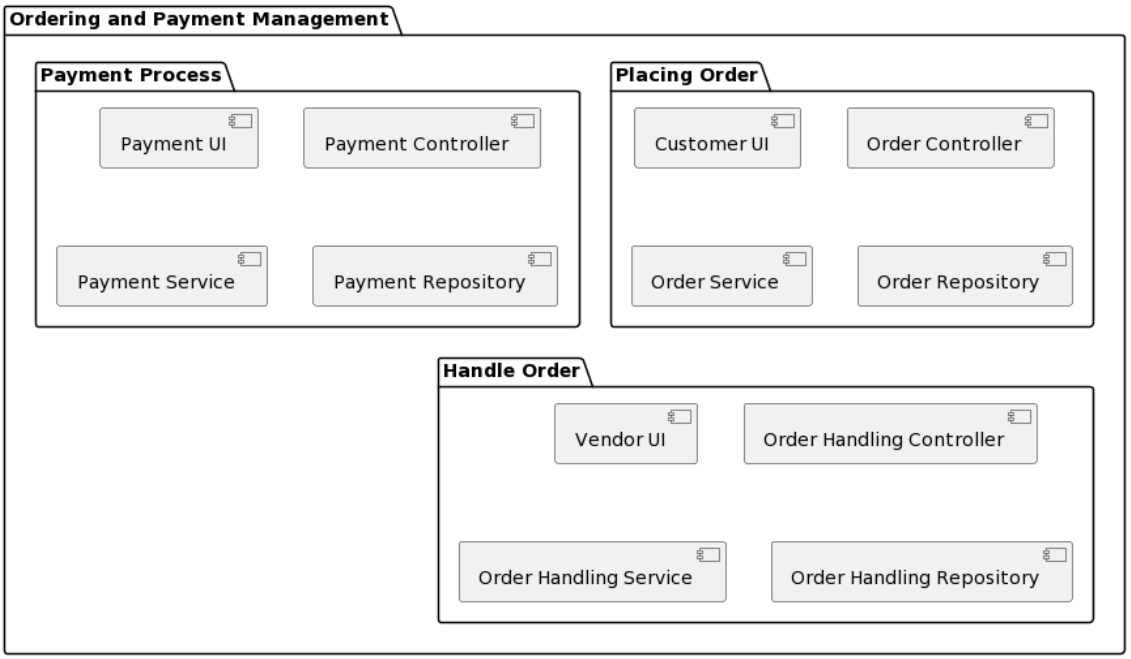


Figure 4.14: Package Diagram for < Ordering and Payment Management > Subsystem

4.2.5.1 Class Diagram

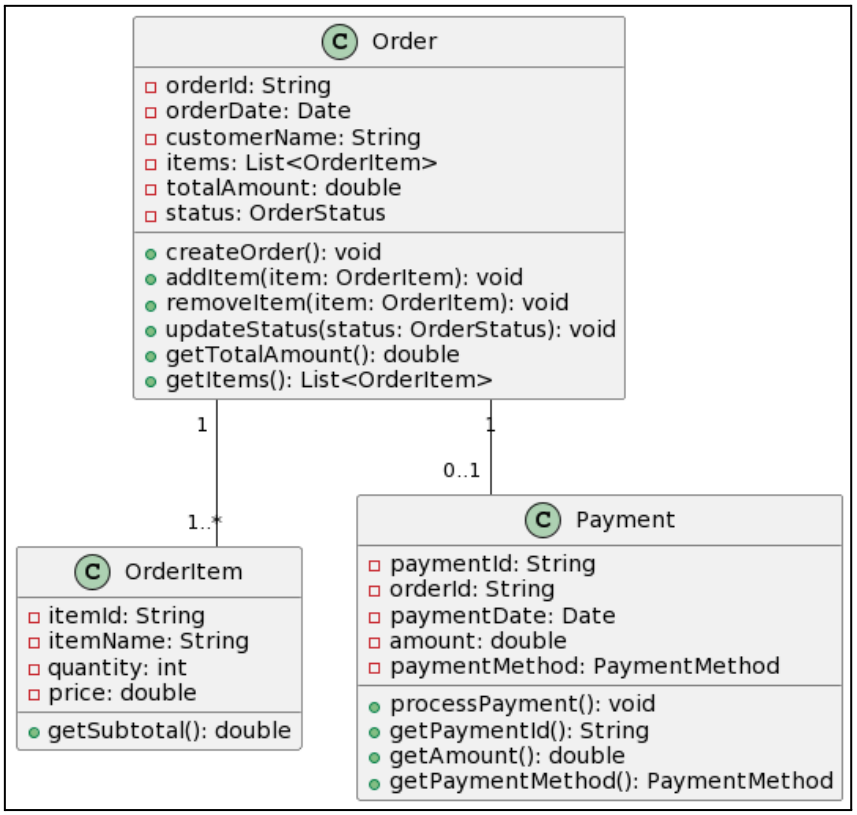


Figure 4.15: Class Diagram for < Ordering and Payment Management > Subsystem

Entity Name	Order
Method Name	createOrder() addItem(item: OrderItem) removeItem(item: OrderItem) updateStatus(status: OrderStatus) getTotalAmount() getItems()
Input	createOrder() - None addItem(item: OrderItem) - OrderItem object removeItem(item: OrderItem) - OrderItem object updateStatus(status: OrderStatus) - OrderStatus getTotalAmount() - None getItems() - None
Output	createOrder() - Void addItem(item: OrderItem) - Void removeItem(item: OrderItem) - Void updateStatus(status: OrderStatus) - Void getTotalAmount() - Total amount(double) getItems() - OrderItem object

Algorithm	createOrder() - Generate a unique order ID and assign it to the orderId attribute. - Set the orderDate attribute to the current date. - Prompt the user to enter the customer's name and assign it to the customerName attribute. - Initialize the items list as an empty list. - Set the totalAmount attribute to 0. - Set the status attribute to an initial status (e.g., "PENDING"). addItem(item: OrderItem) - Add the provided OrderItem object to the items list. - Update the totalAmount attribute by adding the subtotal of the added item. removeItem(item: OrderItem) - Remove the specified OrderItem object from the items list. - Update the totalAmount attribute by subtracting the subtotal of the removed item. updateStatus(status: OrderStatus) - Update the status attribute to the provided OrderStatus value. getTotalAmount() - Calculate the total amount by summing up the subtotals of all items in the items list. - Return the calculated total amount. getItems() - Return the items list.

Entity Name	OrderItem
Method Name	getSubtotal()
Input	None
Output	double
Algorithm	- Calculate the subtotal amount by multiplying the quantity attribute by the price attribute. - Return the calculated subtotal amount.

Entity Name	Payment
Method Name	processPayment() getPaymentId() getAmount() getPaymentMethod()
Input	processPayment() - None getPaymentId() - None getAmount() - None getPaymentMethod() - None
Output	processPayment() - Void getPaymentId() - Payment ID (String) getAmount() - Payment amount (double) getPaymentMethod() - Payment method enumeration
Algorithm	processPayment() - Process the payment associated with the order. - Perform any necessary operations, such as charging the payment method or updating payment details. getPaymentId() - Return the value of the paymentId attribute. getAmount() - Return the value of the amount attribute. getPaymentMethod() - Return the value of the paymentMethod attribute.

4.2.5.2 Sequence Diagram

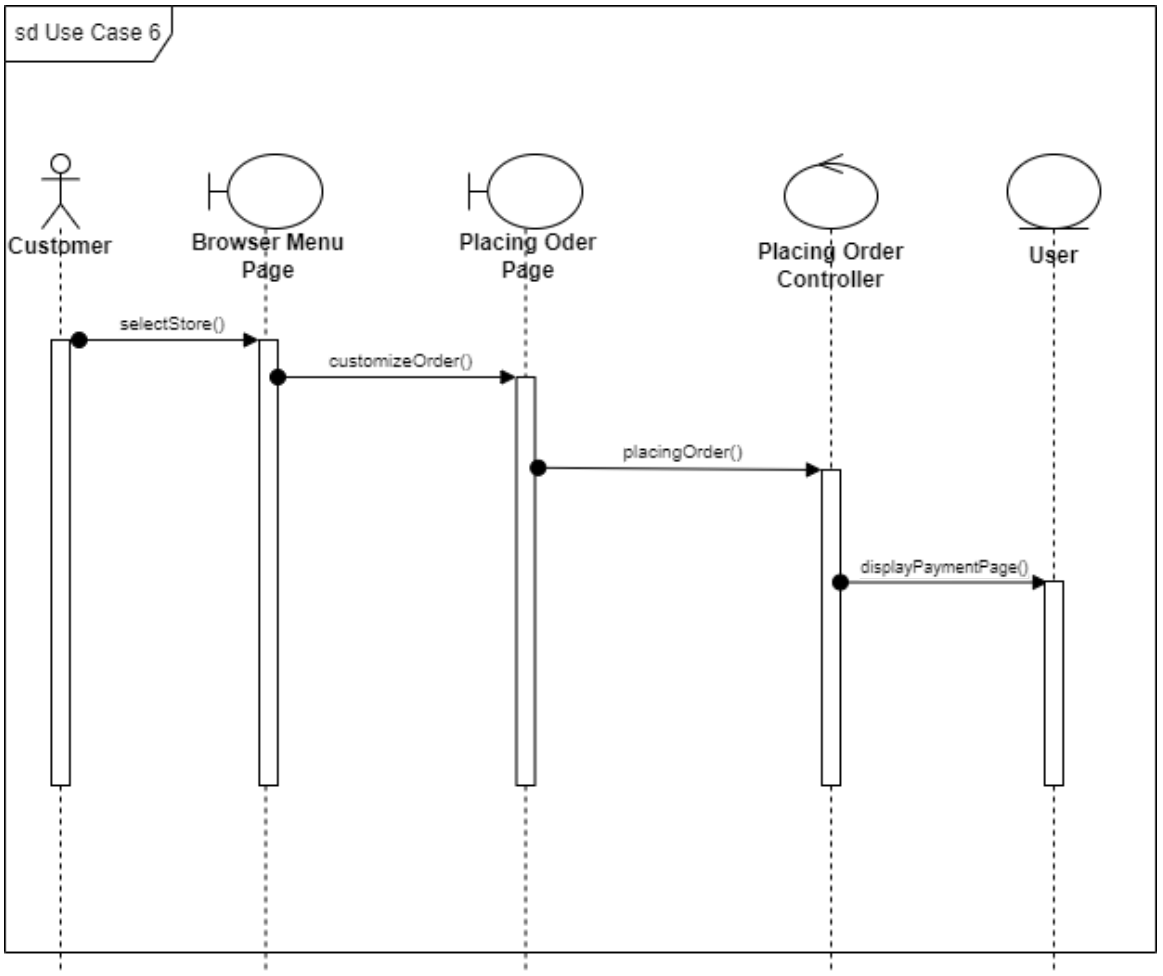


Figure 4.16.1: Sequence Diagram for <Placing Order >

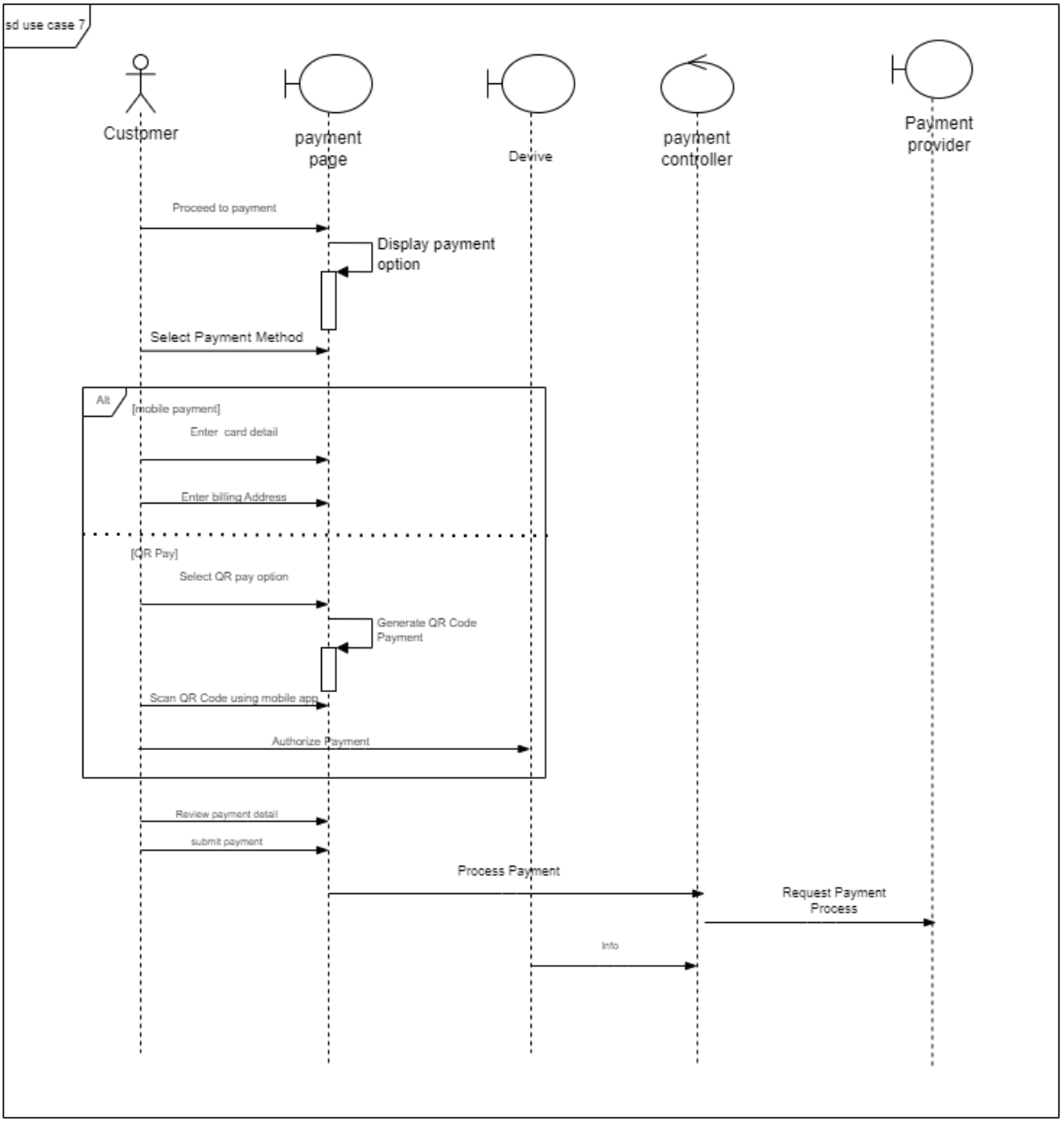


Figure 4.16.2: Sequence Diagram for <Payment Process as Customer >

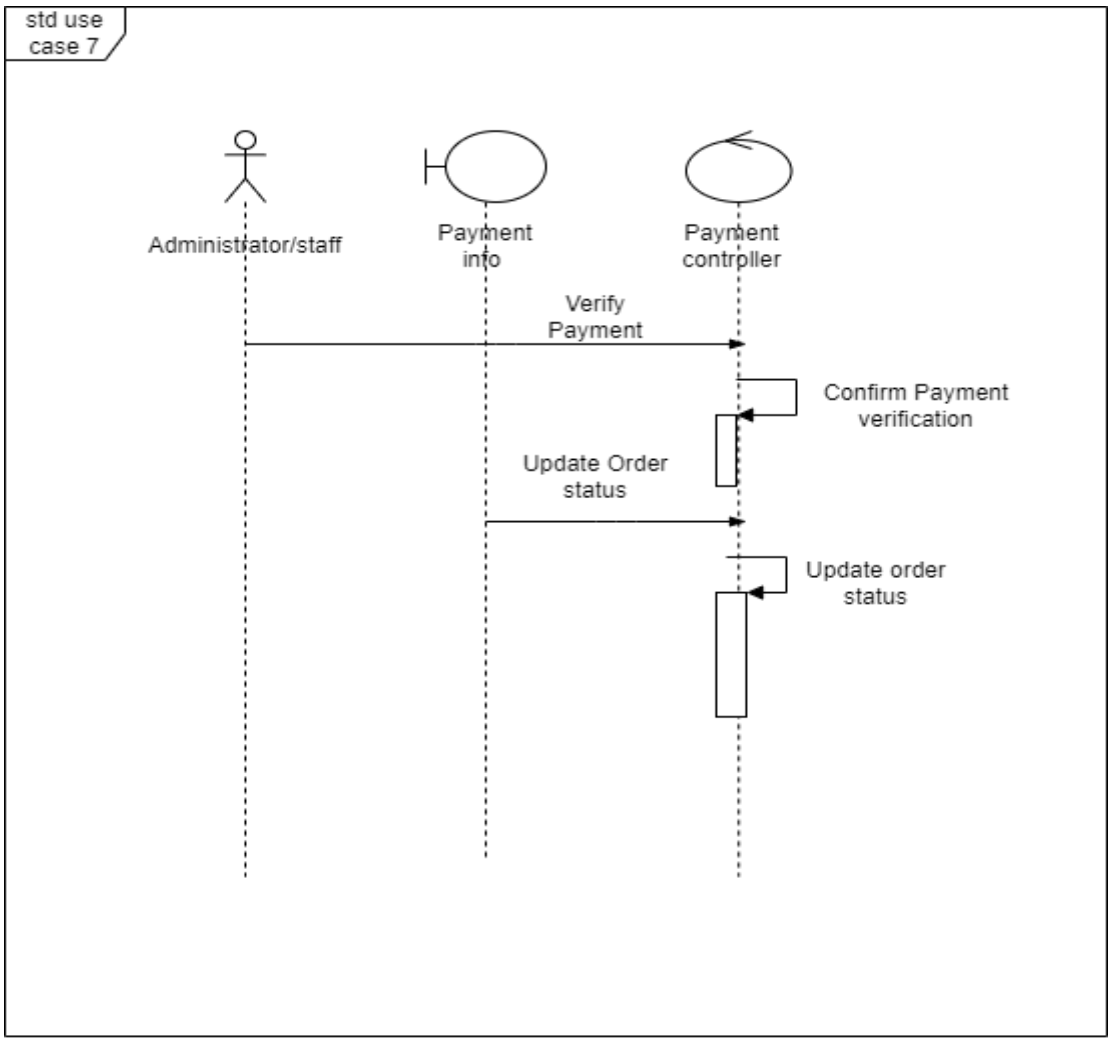


Figure 4.16.3: Sequence Diagram for <Payment Process as Administrator >

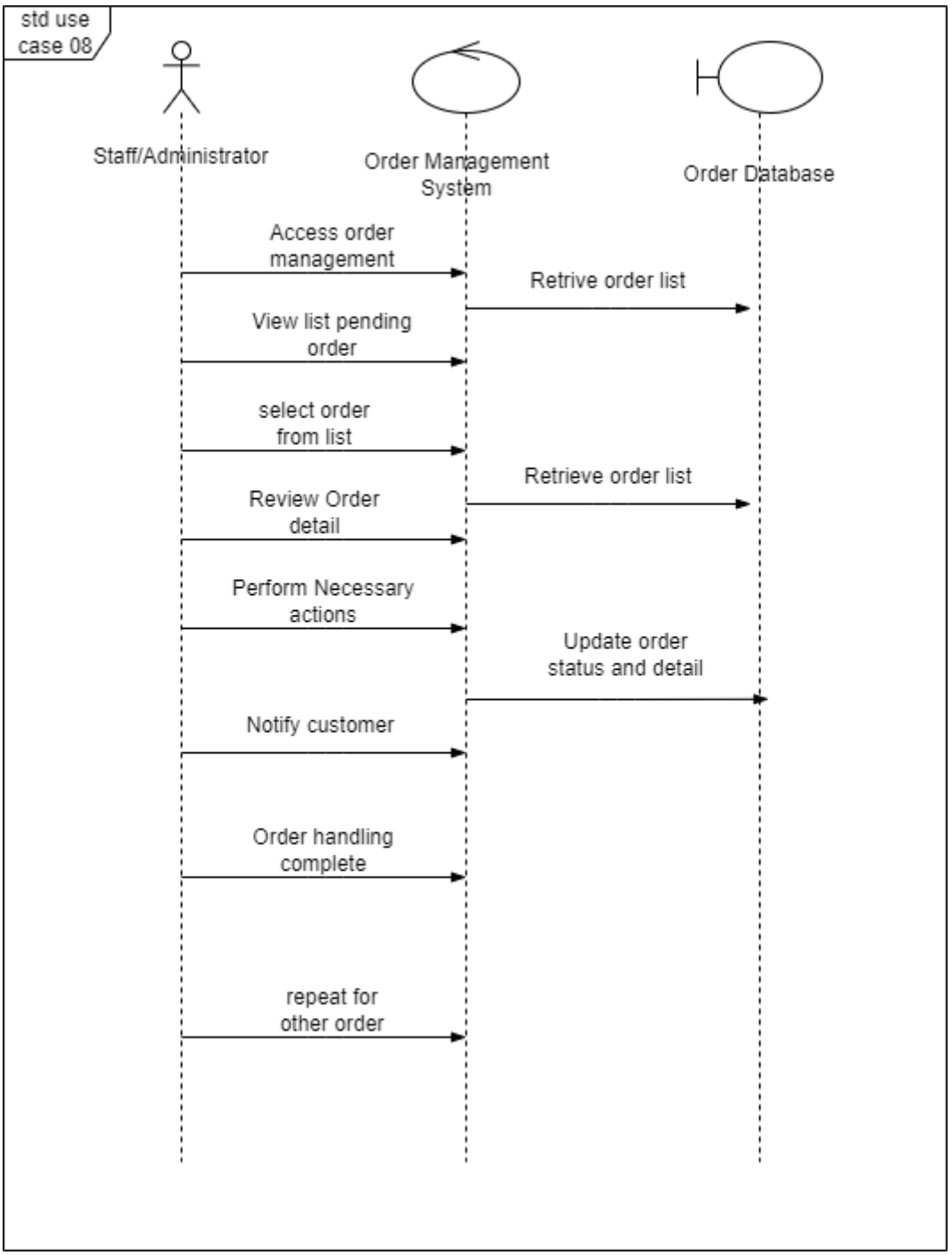


Figure 4.16.4: Sequence Diagram for <Handling order >

5. Data Design

5.1 Data Description

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1.	UserAccount	Represents a user's account in the food ordering system, storing information
2.	UserProfile	Stores additional details about a user, including their name, contact information, and delivery address, enhancing the user's profile in the system.
3.	UserRepository	Manages the storage and retrieval of user account information, providing functionalities for creating, updating, and deleting user accounts, as well as fetching user details from the underlying data source.
4.	RegistrationView	A user interface where individuals can create new accounts within the food ordering system, typically featuring input fields for username, email, password, and other required information.
5.	LoginView	Provides a user interface for users to log into their existing accounts, usually with fields for username/email and password, allowing access to personalized features and functionalities.
6.	ProfileView	Displays the user's profile information, allowing them to view and update their personal details, such as name, contact information, and delivery address.
7.	PasswordRecoveryView	Offers a user interface for recovering a forgotten password, typically by requesting an email or phone number to send a password reset link or verification code.
8.	NotificationService	A class responsible for sending notifications to users, such as order updates, delivery status, or promotional messages. It may utilize various communication channels like email, SMS, or push notifications.
9.	MessageQueue	Manages the queue of notifications to be sent by the NotificationService. It ensures that notifications are sent in a reliable and efficient manner, handling priorities, message ordering, and delivery failures.
10.	Feedback	Represents user feedback or reviews regarding the food ordering system, allowing users to share their experiences, suggestions, or complaints.

No.	Entity Name	Description
11.	Rating	Captures user ratings for specific menu items, delivery service, or overall experience, helping to assess and improve the quality of the food ordering system.
12.	FeedbackRepository	Handles the storage and retrieval of user feedback and ratings, providing methods for saving, retrieving, and analyzing feedback data from the underlying data source.
13.	MenuManager	Manages the menu items available for ordering, including adding new items, updating existing ones, and removing discontinued items. It ensures the accuracy and availability of menu options for users.
14.	MenuBrowser	Provides a user interface for browsing the available menu items, allowing users to view descriptions, prices, and other details to make informed choices when placing orders.
15	Order	Represents an order placed by a customer in the food ordering system.
16	OrderItem	Represents an item included in an order.
17	Payment	Represents a payment made for an order.

5.2 Data Dictionary

5.2.1 Entity: <UserAccount>

Attribute Name	Type	Description
userid	int	Unique identifier for the user account
username	string	User's chosen username for the account
password	string	User's password for the account
email	string	User's email address associated with the account
profile	UserProfile	User's profile information associated with the account

5.2.2 Entity: <UserProfile>

Attribute Name	Type	Description
firstName	string	User's first name
lastName	string	User's last name
address	string	User's delivery address
phone	string	User's contact phone number

5.2.3 Entity: <NotificationService>

Attribute Name	Type	Description
notificationId	int	An identifier for the notification.
recipient	string	The recipient of the notification
messages	string	The content of the notification. It holds the actual message to be sent to the recipient.

5.2.4 Entity: <MessageQueue>

Attribute Name	Type	Description
queueId	int	An identifier for the message queue
messages	string	A collection or list of messages in the message queue.

5.2.5 Entity: <User>

Attribute Name	Type	Description
username	string	Name of user
password	string	Password of user account

5.2.6 Entity: <MenuItem>

Attribute Name	Type	Description
id	int	Number id of menu
name	string	Name of menu
price	double	Price of menu
description	string	Description of menu

5.2.7 Entity: <Rating>

Attribute Name	Type	Description
calculateRating()	float	Calculate average rating

5.2.8 Entity: <Feedback>

Attribute Name	Type	Description
getCustomerId()	int	Return the value of customerId attribute
getComment()	string	Return the value of the comment attribute.

5.2.9 Entity: <FeedbackRepository>

Attribute Name	Type	Description
addFeedback(feedback)	void	Add the provided feedback object to the list of feedbacks.
getAllFeedbacks()	string	Return the list of all feedback stored in the repository.

5.2.10 Entity: <Order>

Attribute Name	Type	Description
createOrder()	void	Creates a new order with a unique ID, sets the order date, customer name, and initializes other attributes.
addItem(item: OrderItem)	void	Adds an OrderItem object to the list of items in the order.
removeItem(item: OrderItem)	void	Removes an OrderItem object from the list of items in the order.
updateStatus(status: OrderStatus)	void	Updates the status of the order to the provided OrderStatus value.
getTotalAmount()	double	Calculates and returns the total amount of the order.
getItems()	string	Returns the list of items in the order.

5.2.11 **Entity: <OrderItem>**

Attribute Name	**Type**	**Description**
getSubtotal()	double	Calculates and returns the subtotal amount for the item based on its quantity and price.

5.2.12 **Entity: <Payment>**

Attribute Name	**Type**	**Description**
processPayment()	void	Processes the payment associated with the order, performs necessary actions such as charging the payment method or updating payment details.
getPaymentId()	String	Returns the payment ID associated with the payment.
getAmount()	double	Returns the amount of the payment.
getPaymentMethod()	int	Returns the payment method used for the payment.

6. User Interface Design

6.1 Overview of User Interface

From the user's perspective, the food ordering system offers a convenient and user-friendly platform to order food from various restaurants. The system provides a range of features that enable users to complete their food orders efficiently.

1. Signup
    - When a new user visits the platform for the first time, they will be directed to the signup page.
    - The user will be prompted to enter their email address, a secure password, and other required information, such as their name and contact number.
    - After filling in the details, the user can submit the form to create a new account.

2. Login
    - For returning users, they can access their accounts by clicking on the "Login" button on the homepage.
    - The user will be prompted to enter their registered email and password.
    - Once the credentials are verified, the user will be logged into their account.

3. Browsing and Restaurant Selection
    - After logging in, the user will be taken to the main page of the food ordering system.
    - Here, they can browse through a list of restaurants available for delivery or takeout.
    - The system will display restaurant names, cuisine types, ratings, and estimated delivery times to help users make informed decisions.

4. Food Selection and Customization
    - Once the user selects a restaurant, they will be taken to the restaurant's menu page.
    - The menu will contain a list of available dishes, along with prices and descriptions.
    - Users can add items to their cart and specify any customizations or special requests, such as ingredient exclusions or additional toppings.

5. Cart Review and Checkout
    - The user can access their cart at any time to review the selected items and their total cost.
    - They can modify the order, add or remove items, or adjust quantities as needed.
    - When satisfied with the order, the user can proceed to the checkout page.

6. Payment and Order Confirmation

- At checkout, the user will be asked to choose their preferred payment method (credit card, PayPal, etc.).
- The system will process the payment securely and generate an order confirmation.
- The user will receive an email or SMS notification confirming the order details and estimated delivery time.

7. Order Tracking and Status Updates

- After placing the order, the user can track its status through the system.
- They will receive real-time updates on the preparation, packaging, and delivery progress.
- Once the order is out for delivery, the user may receive live tracking information.

8. Feedback and Reviews

- After receiving the order, the user will have the opportunity to provide feedback and rate their experience.
- The system may prompt users to leave reviews for the restaurant or individual dishes.
- Users can also view the overall ratings and reviews of the restaurants to aid their future decisions.

The feedback information displayed for the user may include confirmation messages after successful actions (e.g., signup, order placement, payment), error messages in case of any issues (e.g., incorrect login credentials, payment failure), and notifications at various stages of the order process (e.g., order confirmation, out for delivery, delivered).

6.2 Screen Images

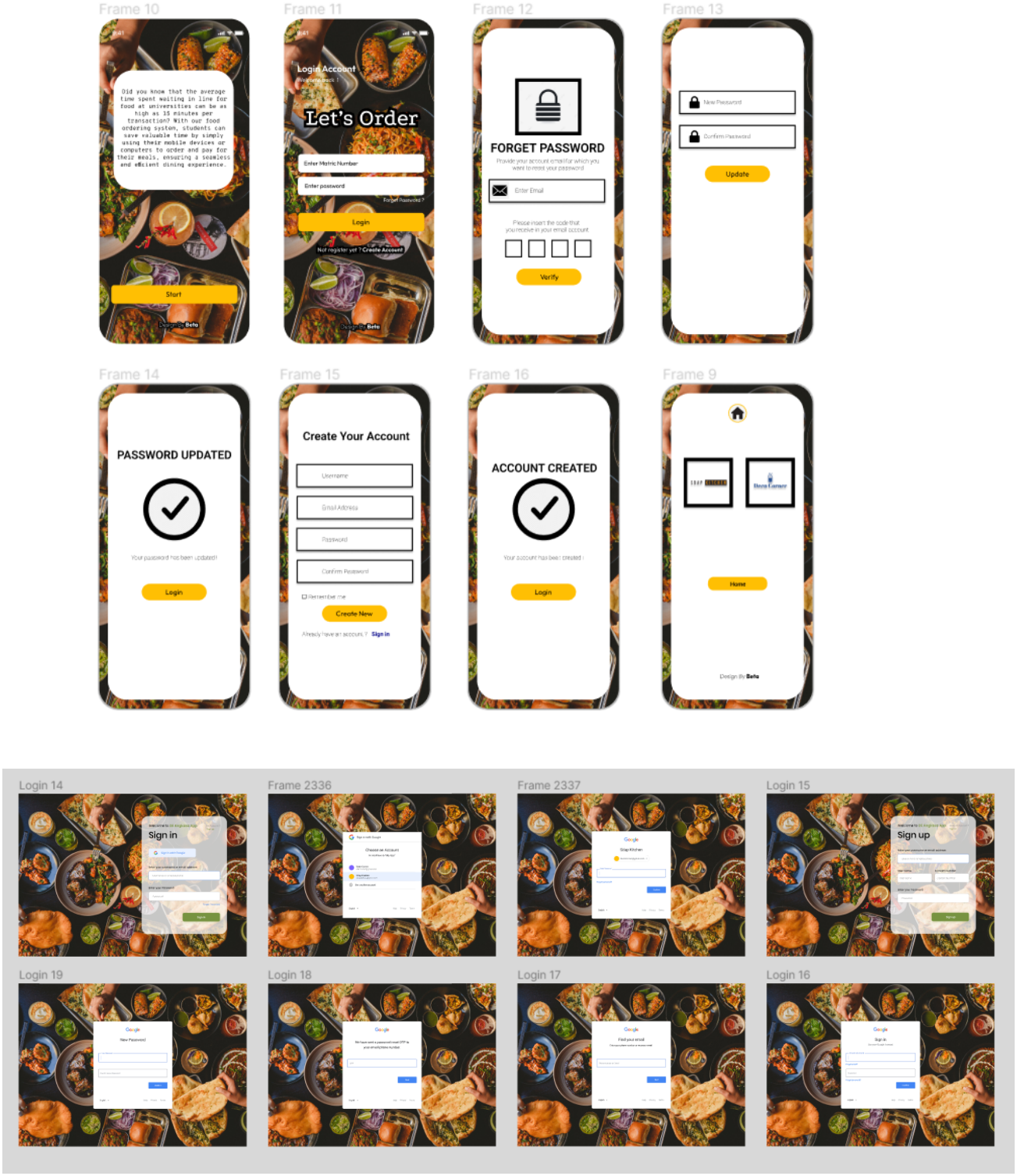


Figure 6.1: Interface for < Manage Account >

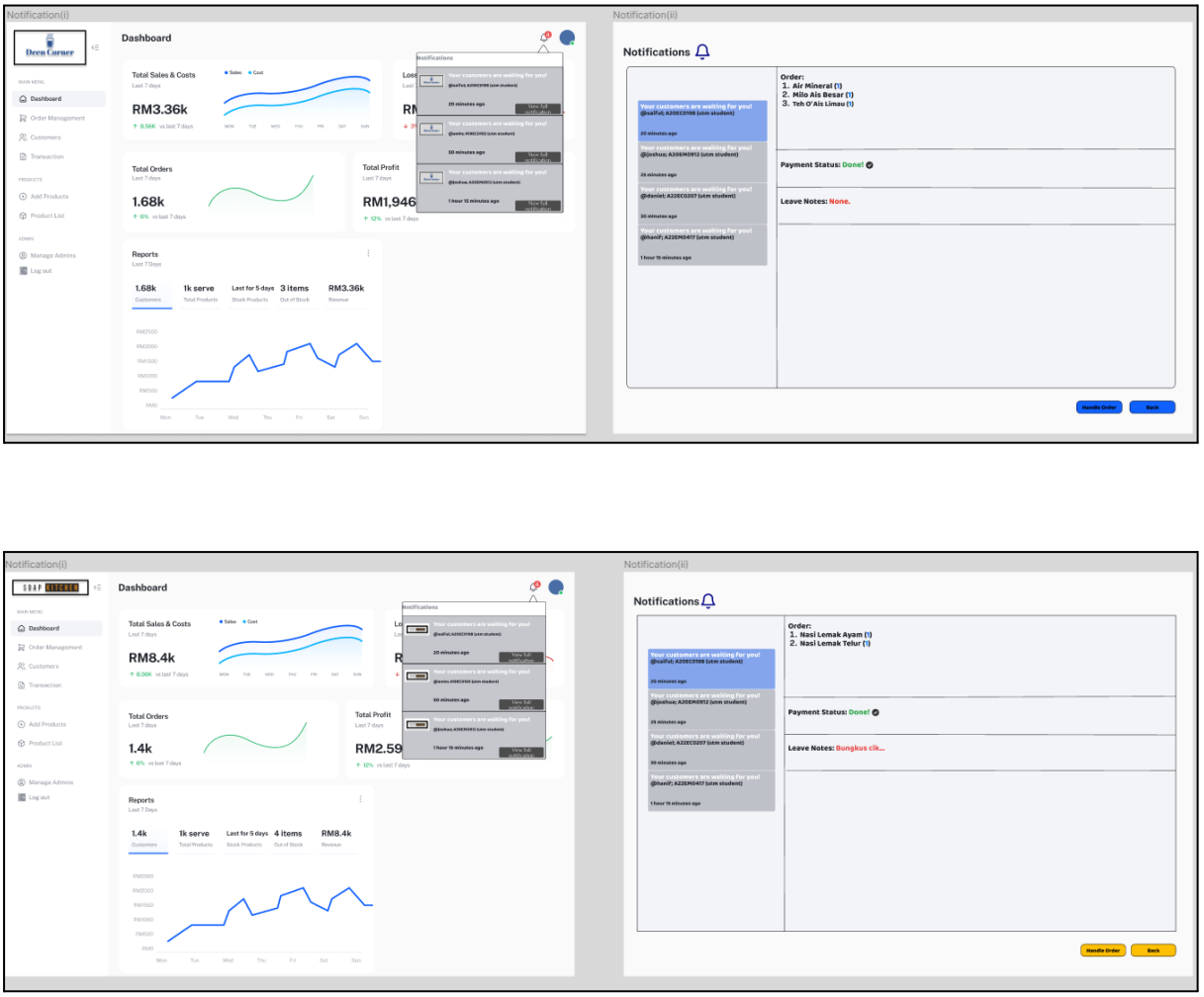


Figure 6.2: Interface for < Sending Notifications >

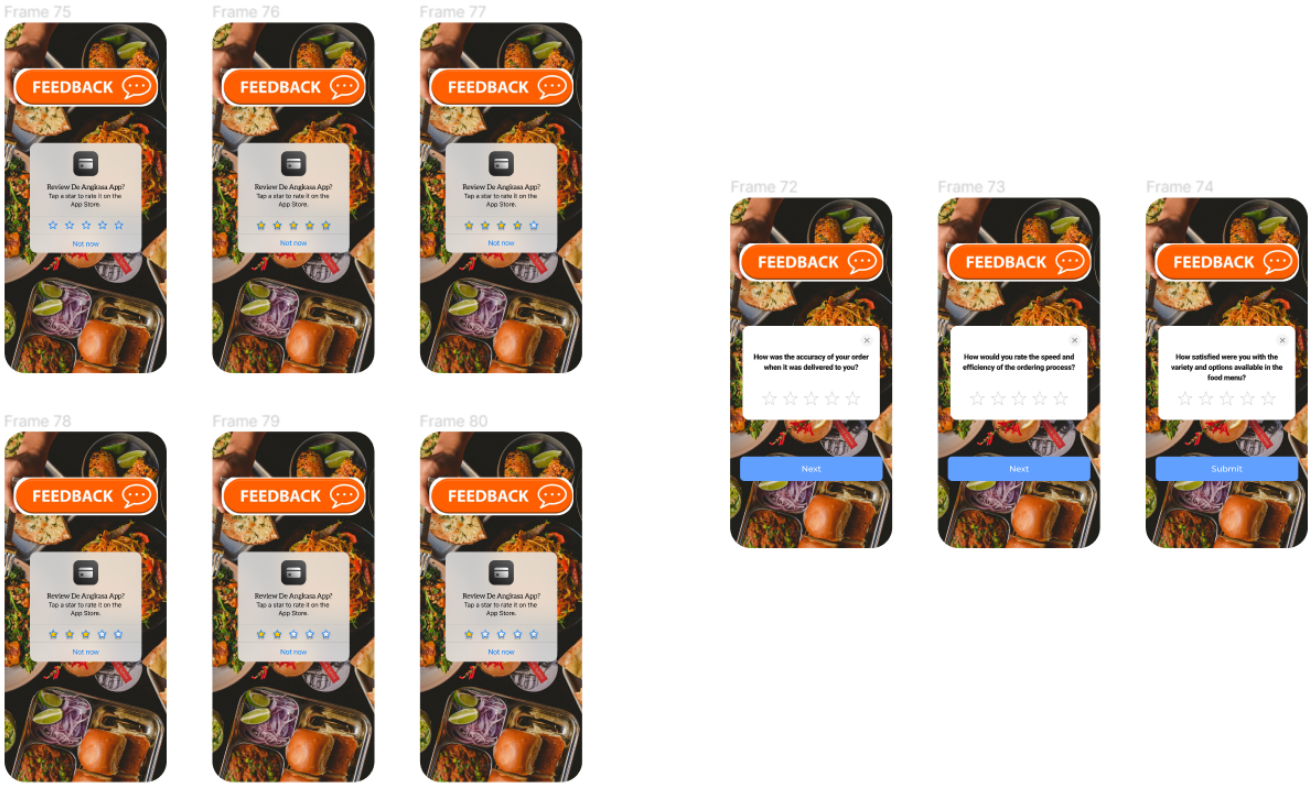
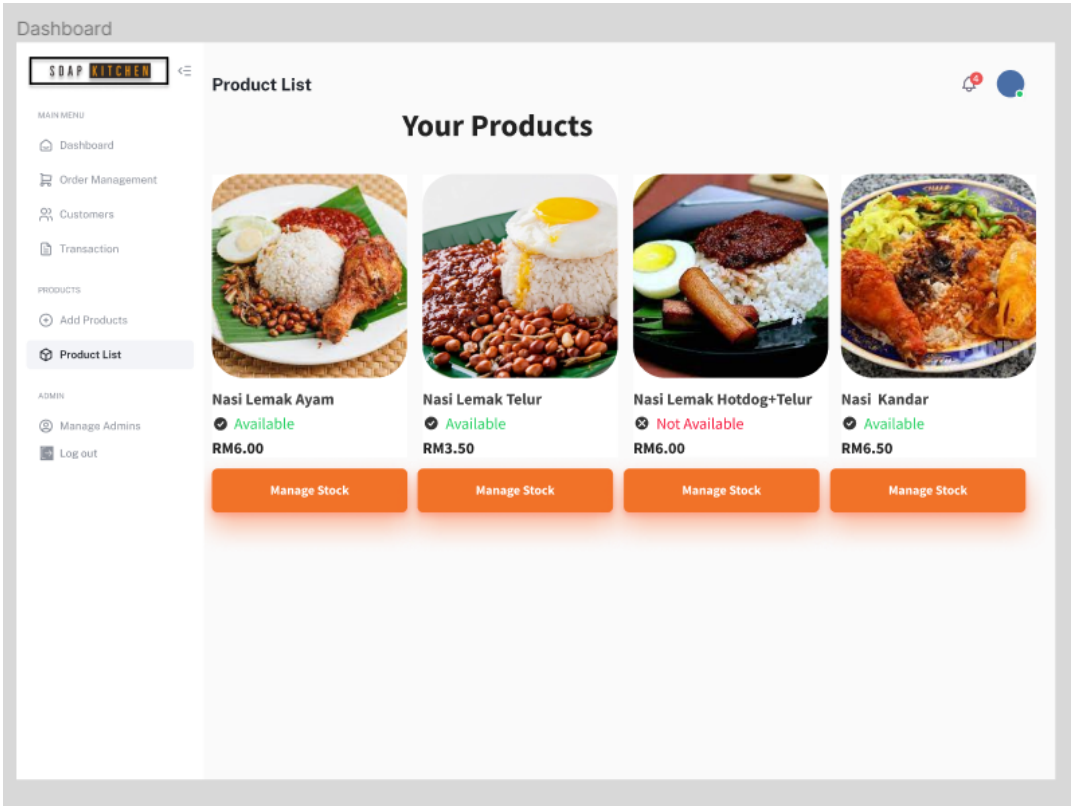


Figure 6.3: Interface for < Provide Feedback and Rating >



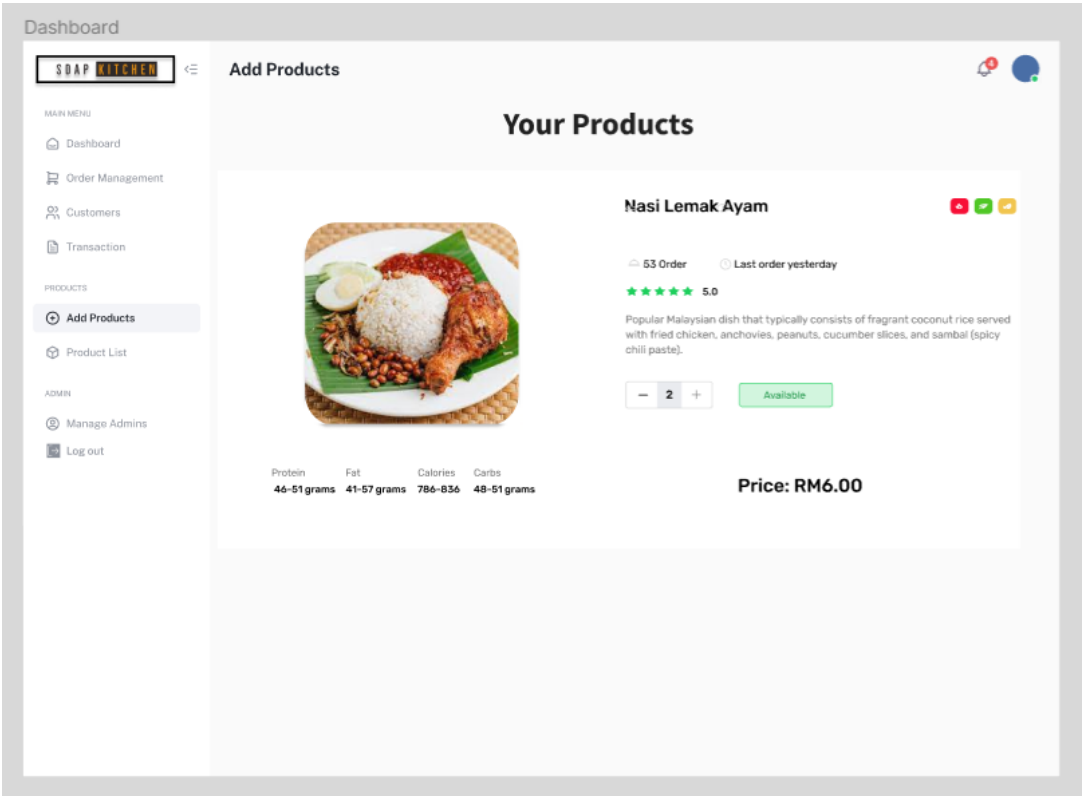


Figure 6.4: Interface for < Manage Menu >

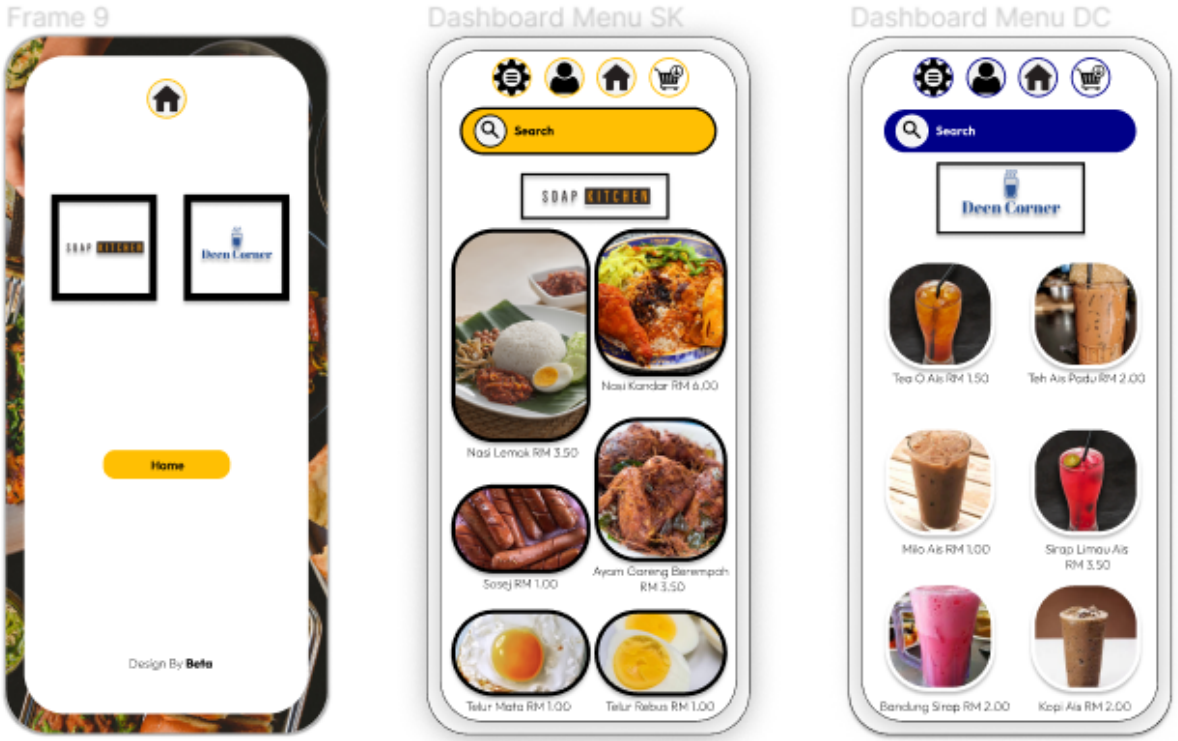


Figure 6.5: Interface for < Browse Menu >

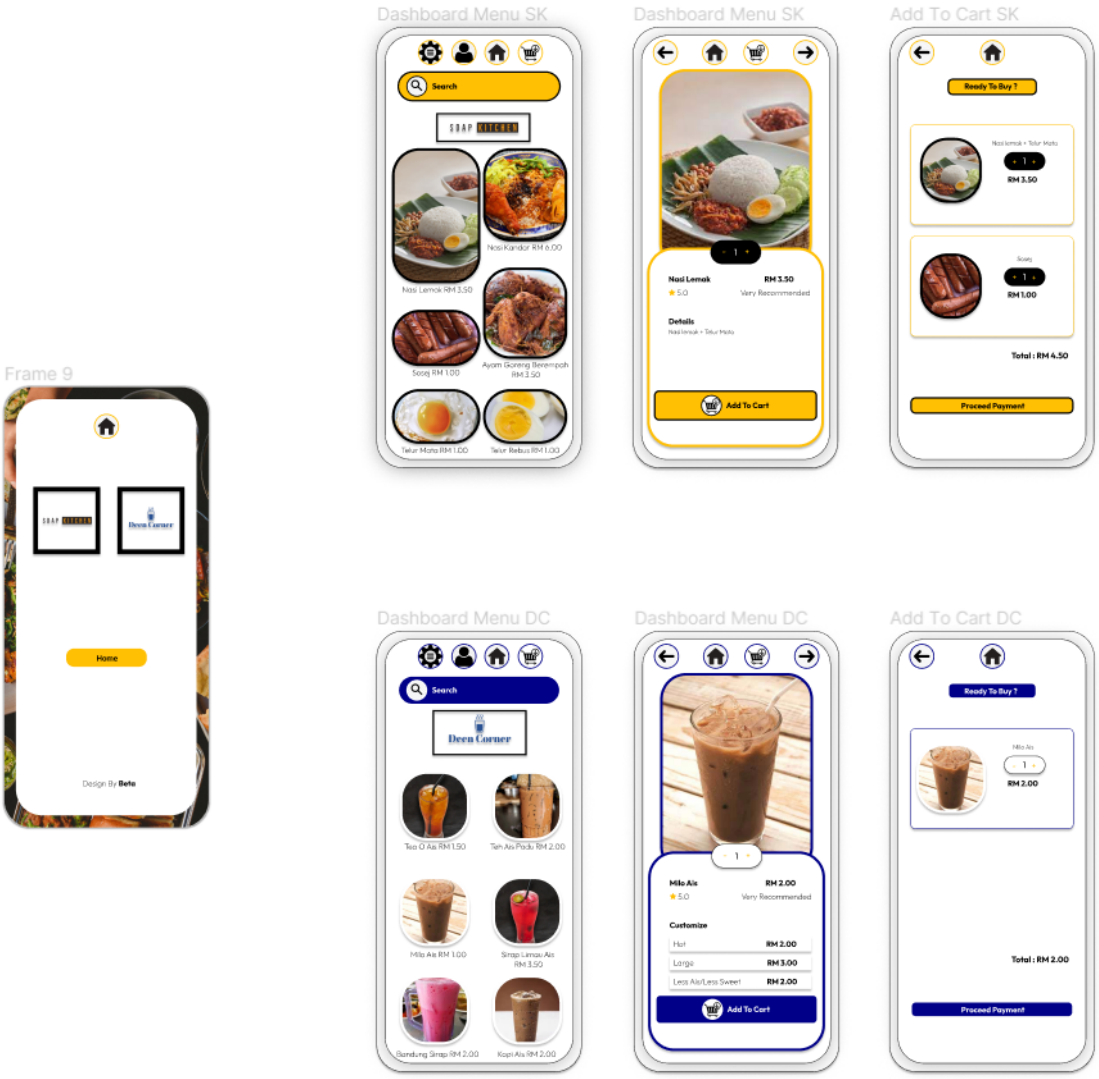


Figure 6.6: Interface for < Placing Order >

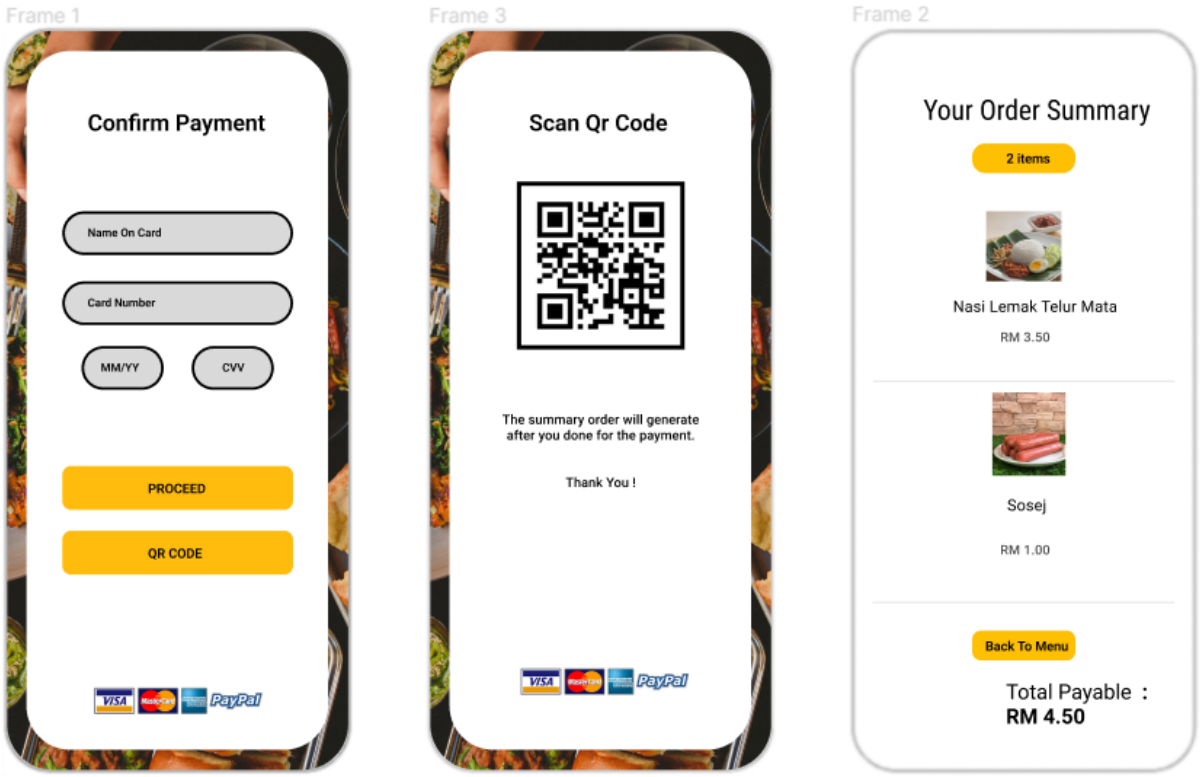


Figure 6.7: Interface for < Payment Process >

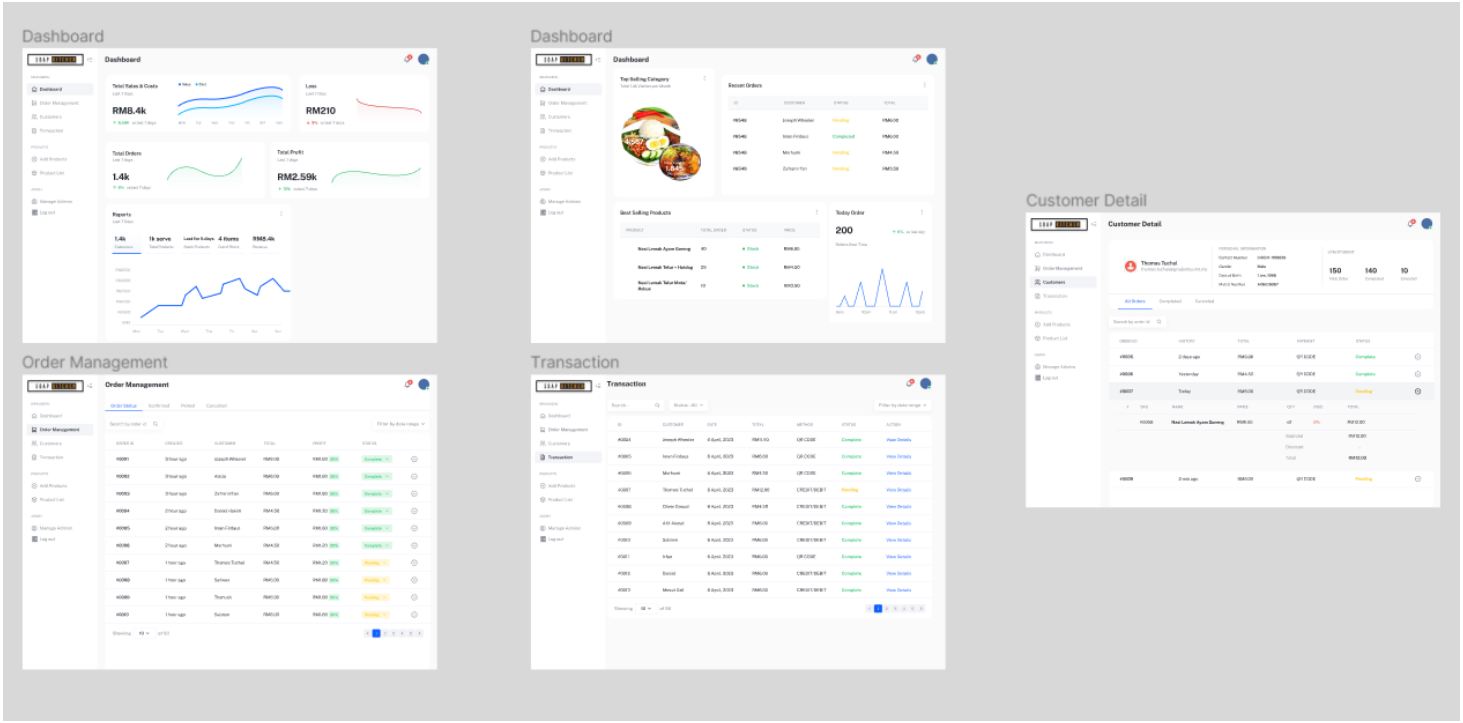


Figure 6.8: Interface for < Handle Order >