UNIVERSITETI® METROPOLITAN TIRANA

MetroMensa

Online On-demand University Canteen Application.
System Documentation

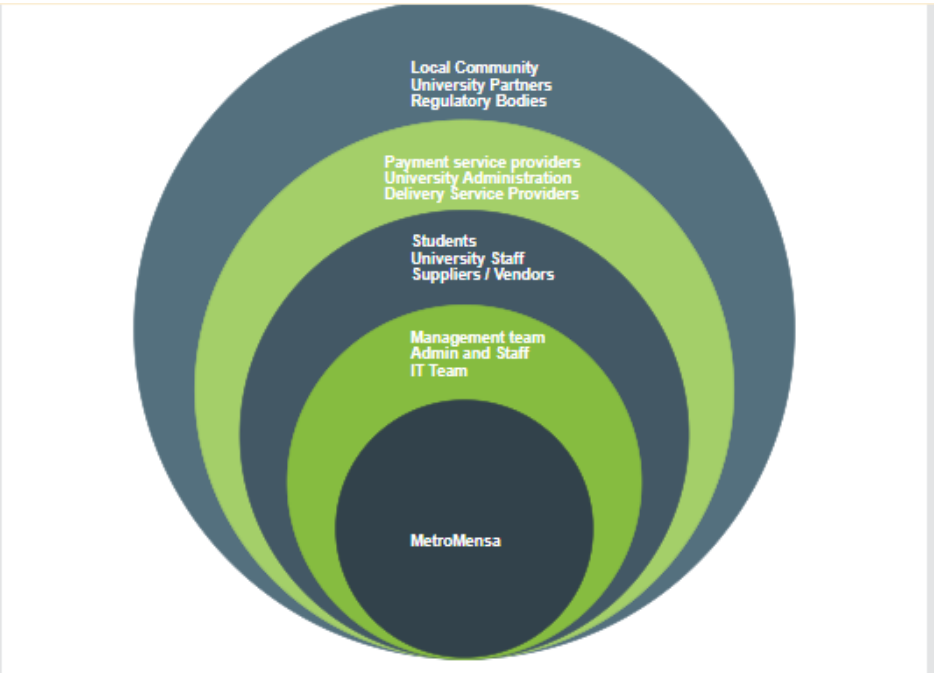
Group Members :
Mateo Hysa
Leandro Leka
Anxhelo Belegu

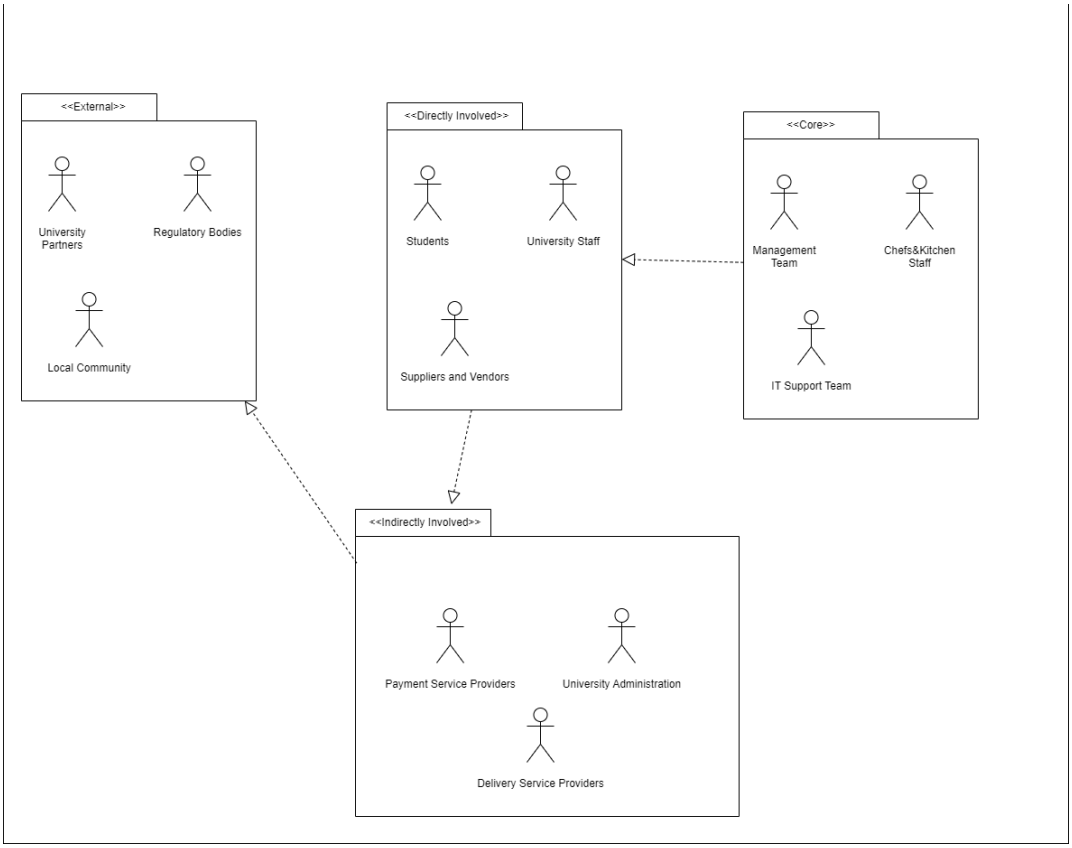
Accepted By : Dr.Igli Hakrama

Table of contents

    i. **State Diagram**
    j. **Entity relation diagram**
    k. **Object Diagram**
    l. **Data flow Diagram**
    m. **Component Diagram**
    n. **Package Diagram**
    o. **Deployment Diagram**
5. **Design Patterns**
    a. **Design Patterns**
6. **User interface**
    a. **User interface**

1. # Summary
    a. ## Overview.
    Metromensa is a comprehensive digital platform designed to enhance the dining experience in our campus and around. It focuses on connecting users with a wide range of dining options, tailored to their preferences and location. The platform integrates advanced features such as personalized recommendations, seamless reservation systems, and user reviews. Metromensa aims to streamline the decision-making process for diners and support local eateries by boosting their visibility and accessibility. Its user-friendly interface and real-time updates ensure a convenient and enjoyable dining experience for urban food enthusiasts.
    b. ## Project purpose.
    The purpose of Metromensa is to revolutionize the urban dining scene by providing a one-stop digital solution for discovering and accessing diverse culinary options. It aims to simplify the decision-making process for diners through personalized recommendations and easy reservation features. The scope of the project encompasses the integration of real-time restaurant data, user preferences, and a robust review system, catering to both casual diners and food connoisseurs. Ultimately, the project aspires to create a dynamic and interactive dining ecosystem that benefits both consumers and food establishments.
    c. ## Stakeholders





2. Product description

a. Context

Metromensa is a dynamic digital platform tailored for urban dining, offering personalized restaurant discovery and reservation services. It caters to a wide user base, from students to casual diners to food enthusiasts, providing an intuitive interface for a seamless dining experience. The platform emphasizes user-centric features, ensuring a tailored and enjoyable culinary exploration.

b. User characteristics

Metromensa is primarily tailored for students in urban areas, offering a streamlined platform for discovering and enjoying diverse dining experiences. It addresses the unique preferences and constraints of student life, such as budget-friendly options, proximity to campuses, and quick meal solutions. The platform is user-friendly, catering to tech-savvy students who are familiar with digital apps and seek convenience in their culinary choices. Metromensa facilitates social dining experiences, making it easier for students to explore new eateries and organize group meals, enhancing their overall university life experience.

c. Assumptions

Metromensa assumes that its primary users, predominantly students, possess basic digital literacy and have access to smartphones or computers with internet connectivity. The platform is designed with the understanding that students are familiar with navigating similar digital applications, making it intuitive and user-friendly. It also assumes that students are seeking quick, budget-friendly dining options, aligning with their lifestyle and preferences. Additionally, Metromensa presumes a need for social connectivity, facilitating group dining experiences and social interactions among the student community.

d. Constraints

Metromensa is segmented into various sections, each tailored for different user interactions, and assumes a basic level of familiarity with similar digital platforms. The design of the system limits user customization to maintain stability and prevent anomalies in its operation. Essential features, such as reservation making and accessing detailed restaurant information, depend on specific criteria like real-time restaurant availability. Furthermore, the performance and responsiveness of Metromensa are influenced by the overall user traffic.

e. Dependencies

Metromensa's core operations are closely tied to the university's canteen system, relying on their menu updates, food availability, and service hours to provide accurate information to students. The platform's ability to process food orders and provide estimated waiting times is dependent on the canteen's real-time order flow and capacity. Future expansions to include other local restaurants hinge on establishing partnerships and integrating their systems into Metromensa. Additionally, the app's performance and user experience are influenced by the volume of student usage, especially during peak dining hours.

3. Requirements

a. Functional requirements

- **User Account Management:** Enable users to create, edit, and manage their personal accounts, including setting preferences and dietary restrictions.

- **Menu Display and Updates:** Automatically update and display the current menu from the university canteen, including prices, dish descriptions, and nutritional information.

- **Order Placement System:** Allow users to select items from the menu, specify quantities, and place orders for pickup or dine-in.

- **Payment Processing:** Integrate secure payment options for users to pay for their orders online via credit/debit cards.

- **Order Status Tracking:** Provide real-time updates to users on the status of their orders, from preparation to ready-for-pickup or delivery.

- **Feedback and Rating System:** Enable users to rate and review dishes and services, contributing to a feedback system for quality improvement.

- **Dietary and Allergen Information:** Provide detailed information about allergens and dietary categories (e.g., vegan, gluten-free) for each menu item.

- **Order History and Reordering:** Maintain a history of past orders for users and allow quick reordering of favorite items.

- **Social Sharing:** Enable users to share their favorite dishes or reviews on social media platforms directly from the app.

- **Integration with University Systems:** Seamlessly integrate with existing university systems for student identification and meal plan validation.

- **Multilingual Support:** Offer the app's interface and menu descriptions in multiple languages to cater to a diverse student population.

b. Non-functional requirements

i. Product Requirements

1. User Interface Requirements.

a. **Browser Accessibility:** In addition to a dedicated mobile app, Metromensa should be accessible via a web browser, ensuring that users can interact with the service on a desktop or laptop computer. The web interface should offer a comparable level of functionality and user experience to the mobile app, including intuitive navigation, real-time updates, and a visually appealing design. This accessibility ensures that all students, regardless of their preferred device, can conveniently use Metromensa's services.

b. **Intuitive and User-Friendly Design:** The interface should be straightforward and easy to navigate, similar to UberEats, allowing users to quickly find and order food. Key features should be accessible within a few taps, and the app should offer a clean, uncluttered layout with clear call-to-action buttons.

c. **Responsive and Adaptive Layout:** The app's design should be responsive, ensuring a seamless experience across various devices and screen sizes, including smartphones and tablets. It should adapt its layout and functionality to different devices while maintaining consistency in user experience.

d. **Visually Appealing Presentation:** The app should employ an aesthetically pleasing design with high-quality images of menu items, mirroring UberEats' approach to visually engaging users. This includes using appropriate color schemes, typography, and iconography that align with the university's branding and appeal to the student demographic.

2. **Usability**

a. **Ease of Learning and User Onboarding:** The Metromensa platform should be designed for ease of learning, enabling new users to quickly become proficient in using the app. This could include an onboarding tutorial for first-time users, demonstrating key functionalities like menu navigation, order placement, and tracking. The goal is to minimize the learning curve, making the app accessible even to those not familiar with similar platforms.

b. **Accessibility for All Users:** The app should be universally accessible, catering to users with varying abilities. This includes compliance with accessibility standards like screen reader compatibility, voice commands, and options for larger text. The design should consider color contrast for better visibility and offer alternative navigation methods for those with limited dexterity. Ensuring accessibility broadens the

app’s usability, making it inclusive for all students in the university community.

3. **Efficiency**
    a. **High Performance and Low Latency:** Metromensa should be optimized for high performance, ensuring swift loading times and minimal lag in user interactions, especially during peak dining hours. This includes efficient data handling and optimized server responses to handle multiple concurrent user requests, ensuring that the app remains responsive and reliable at all times, similar to the performance standards set by local industry leaders.

    b. **Resource Optimization on Devices:** The app should be designed to minimize the use of device resources such as battery life and data consumption. This involves optimizing image sizes, caching frequently accessed data, and ensuring that the app's background processes are not overly demanding on the user’s device. This consideration is particularly crucial for students who may rely on the app throughout the day and may not always have access to charging facilities or Wi-Fi.
4. **Security.**
    a. **Simplified User Authentication:** Implement a straightforward yet secure authentication system, like password-based logins combined with security questions. This approach balances security with the ease of implementation for a smaller team.

    b. **Basic Data Encryption:** Focus on implementing standard SSL/TLS encryption for data transmission, particularly for sensitive data like personal details and order information. This is a fundamental and essential security measure that is manageable for a small team.

    c. **Periodic Security Reviews**: Schedule regular but manageable security reviews to ensure the app's integrity.

This can include basic vulnerability scanning and keeping abreast of security best practices, which are crucial yet achievable tasks for a small development team.

5. **Technology Stack**

   For the user interface of MetroMensa, utilizing React for web application development and React Native for mobile application development would be highly beneficial. React's component-based architecture facilitates the development of dynamic and responsive user interfaces, essential for an engaging user experience in the canteen system. It allows for seamless updates and state management, which is crucial for real-time menu updates and order management. React Native extends these advantages to mobile platforms, providing a native-like experience while enabling code reuse across web and mobile applications, thus optimizing development time and effort. For handling real-time data, such as order statuses and menu updates, integrating a solution like Socket.IO with React would enable efficient bidirectional communication between the web clients and the server.

   On the backend, Node.js, with its event-driven, non-blocking I/O model, would be an excellent choice for building a fast and scalable server. It's particularly well-suited for applications that require real-time capabilities, like processing food orders and user feedback in MetroMensa. For the database, using MongoDB, a NoSQL database, would allow for flexible data storage and easy scaling. MongoDB's schema-less nature is ideal for storing diverse data types that the canteen system might handle, such as user profiles, menu items, orders, and feedback. Moreover, integrating an ORM (Object-Relational Mapping) tool, like Mongoose for MongoDB, would simplify database interactions and data validation in Node.js. This stack, known as the MERN stack (MongoDB, Express.js, React, Node.js), is widely recognized for its performance, scalability, and developer productivity, making it a suitable choice for the MetroMensa system.

ii. Organizational Requirements.

1. Environmental Requirements.

a. MetroMensa will utilize a cloud-based infrastructure, ensuring efficient use of computing resources and minimizing energy wastage. This approach allows for scalable resource allocation, adapting to varying demand without the need for excessive physical infrastructure. The system's design emphasizes sustainability and operational efficiency, aligning with environmental conservation goals.

2. Operational Requirements.

a. MetroMensa's operational framework is designed to facilitate efficient communication and coordination across the university canteen's functionalities. The system's architecture allows for seamless data conversion and integration, enabling users to easily navigate and utilize various features for ordering food, managing preferences, and tracking orders. This operational design ensures that MetroMensa efficiently meets the diverse needs of the university student body, enhancing the overall dining experience on campus.

c. Domain Requirements

1. MetroMensa is a web-based application but also a mobile designed to cater to a diverse user base, primarily students, with features and access privileges varying based on user roles. The platform offers specialized access: students can browse menus, place orders, and track their order status, while administrative users from the university can manage the canteen's offerings, update menus, and analyze order data. This domain-specific design ensures that MetroMensa effectively addresses the unique needs of both the student community and the university's administrative staff, facilitating efficient food service operations within the campus environment.

d. External Requirements.

i. Regulatory requirements.

1. For MetroMensa, the regulatory requirements primarily focus on food service and safety. The platform must comply with Albanian food safety regulations, ensuring all food handling, storage, and preparation meet health standards. Additionally, the university

canteen and potential future partners are required to undergo regular health inspections and maintain necessary safety certifications, with MetroMensa responsible for verifying ongoing compliance.

ii. Ethical requirements.

1. **Transparency and Honesty:** MetroMensa should provide clear, accurate, and comprehensive information about food items, including ingredients, nutritional values, and sourcing. Prices and any additional fees should be clearly stated to avoid misleading users.
2. **User Privacy and Data Protection:** Respect users' privacy by implementing strict data protection measures, only collecting necessary information, and never sharing personal data without consent. Users should be informed about what data is collected and how it is used.

e. Legislative Requirements.

i. **General Data Protection Regulation (GDPR):**
Consent: Obtain clear and explicit consent from users before collecting, processing, or storing their personal data. Right to Access: Ensure users can access their personal data and understand how it's being used. Data Portability: Allow users to obtain and reuse their personal data across different services. Right to be Forgotten: Enable users to have their personal data deleted upon their request. Data Protection: Implement measures to securely store and transfer personal data, preventing unauthorized access and breaches.

ii. **Albanian Legislation (Law No. 9887, Date 10.3.2008 on "Protection of Personal Data"): Data Controller and Processor Compliance:**
Ensure that the data controller and processors are adhering to the Albanian laws regarding personal data protection. Notification of Data Collection: Inform the Information and Data Protection Commissioner and individuals about the collection of personal data and its purpose. Data Security Measures: Implement appropriate technical and organizational measures to protect personal data against accidental or unlawful destruction, loss, alteration, unauthorized disclosure, or access. Cross-Border Data Transfer: Adhere to regulations governing the transfer of personal data to countries outside Albania, ensuring they provide an adequate level of protection.

4. Software Design

a. User Scenarios.

Scenario Title: Student Orders a Meal

- A student opens the MetroMensa app on their smartphone.
- They browse through the canteen's menu and select their desired items.
- After reviewing their order, the student confirms and pays using their preferred payment method.
- The system sends a confirmation and an estimated time for order readiness.

Scenario Title: Admin Updates the Canteen Menu

- The canteen admin logs into the MetroMensa system through a web browser.
- They navigate to the menu management section and update the menu items, including prices and dish descriptions.
- Once the updates are saved, the system immediately reflects these changes in the app for students to view.

Scenario Title: Student Leaves Feedback on a Meal

- After eating, a student logs back into the MetroMensa app.
- They navigate to their past orders, select the recent one, and leave a rating and review for the meal.
- The system updates the meal's rating and makes the review visible to other users.

Scenario Title: Admin Analyzes Sales Data

- The canteen admin wants to understand the most popular dishes for the week.
- They log into the MetroMensa system and access the sales analytics dashboard.

- The dashboard presents a breakdown of the week's most ordered dishes, helping the admin make informed decisions on menu planning.

Scenario Title: Student Sets Dietary Preferences

- A student with specific dietary needs (e.g., vegan, gluten-free) uses the MetroMensa app.
- They navigate to their profile settings and set their dietary preferences.
- The app filters and highlights menu items that meet their dietary requirements during future orders.

Scenario Title: Group Order Placement

- A group of students decides to order lunch together using MetroMensa.
- One student creates a group order and invites others to add their selections.
- Each member of the group adds their items, and the group leader places the final order for everyone.

Scenario Title: Urgent Menu Update

- A dish runs out unexpectedly in the canteen.
- The canteen admin quickly accesses the MetroMensa system to mark the dish as unavailable.
- Students are immediately notified of the change, preventing further orders of the unavailable dish.

b. Use cases extended.

Use Case ID	UC-MM-01
Use Case name	Login User
Scope	MetroMensa Application
Level	User Goal
Primary Actor	Student/User
Stakeholders and interests	Student/User wants to securely access their account to order food, view history etc. Canteen Staff needs to verify user identity to process orders accurately.
Preconditions	The user must be registered with the MetroMensa System
Postconditions	The user is logged on to the system and has access to application features.
Main Success Scenario	User opens the application. Enters the username and password. The system verifies the credentials. The user is granted access to their account.
Extensions	1. If the user enters the incorrect credentials a message to be displayed. 2. If the user has forgotten their password the user can reset it.
Minimum guarantees	The user is informed of any login failures.
Success guarantees	User accesses the system with correct credentials.
Trigger	User initiates a login attempt
Frequency of occurrence	Potentially multiple times per day.

Use Case ID	UC-MM-02
Use Case name	Place Order
Scope	MetroMensa Application
Level	User Goal
Primary Actor	Student/User
Stakeholders and interests	Student / User wants to order food conveniently through the app. Canteen Staff want to receive and fulfill orders.
Preconditions	User must be logged in to the MetroMensa system.
Postconditions	The user’s order is placed and pending confirmation from the canteen.
Main Success Scenario	1. The user selects the 'Order Food' option. 2. The user browses the menu and selects their meal. 3. The user adds the meal to the cart and applies any customizations. 4. The user reviews their order and confirms the details. 5. The user completes the payment process. 6. The system confirms the order and sends it to the canteen staff.
Extensions	1. If the selected meal is not available, the system notifies the user to select a different option. 2. If the user wants to add special dietary requests, the system allows customizations. 3. If the payment fails, the system allows the user to retry.
Minimum guarantees	The user is informed if the order cannot be placed.
Success guarantees	The user's order is placed successfully and acknowledged by the system.
Trigger	User decides to place an order for food.
Frequency of occurrence	Multiple times a day, especially during mealtimes.

Use Case ID	UC-MM-03
Use Case name	Update Canteen Menu
Scope	MetroMensa System
Level	User Goal
Primary Actor	Canteen Admin/Canteen Staff
Stakeholders and interests	1. Canteen Admin: Wants to easily update the menu to reflect the current offerings. 2. Students/Users: Interested in viewing the latest menu to make their meal choices. 3. Canteen Staff: Needs to know the updated menu to prepare the meals accordingly.
Preconditions	Canteen Admin must be logged in and have admin access to the menu management system.
Postconditions	The canteen menu is updated and immediately viewable by the users.
Main Success Scenario	1. Admin logs into the MetroMensa system. 2. Admin navigates to the menu management section. 3. Admin selects the option to update the menu. 4. Admin makes necessary changes to the menu items, including prices, descriptions, and availability. 5. Admin submits the changes for review. 6. The system validates the updated information. 7. The system confirms the menu update. 8. The updated menu is published to the MetroMensa system.
Extensions	1. Admin decides to add a new menu item: a. Admin selects the option to add a new item. b. Admin enters details for the new item.
Minimum guarantees	The system should reflect the updated menu in real-time without the need for users to refresh their app or webpage.

Success guarantees	The feedback provided by the user is accurately recorded and made available for review by the canteen staff, contributing to the improvement of the services and meal quality.
Trigger	The user's completion of a meal ordered through the MetroMensa system
Frequency of occurrence	Daily, as menu offerings may change daily.

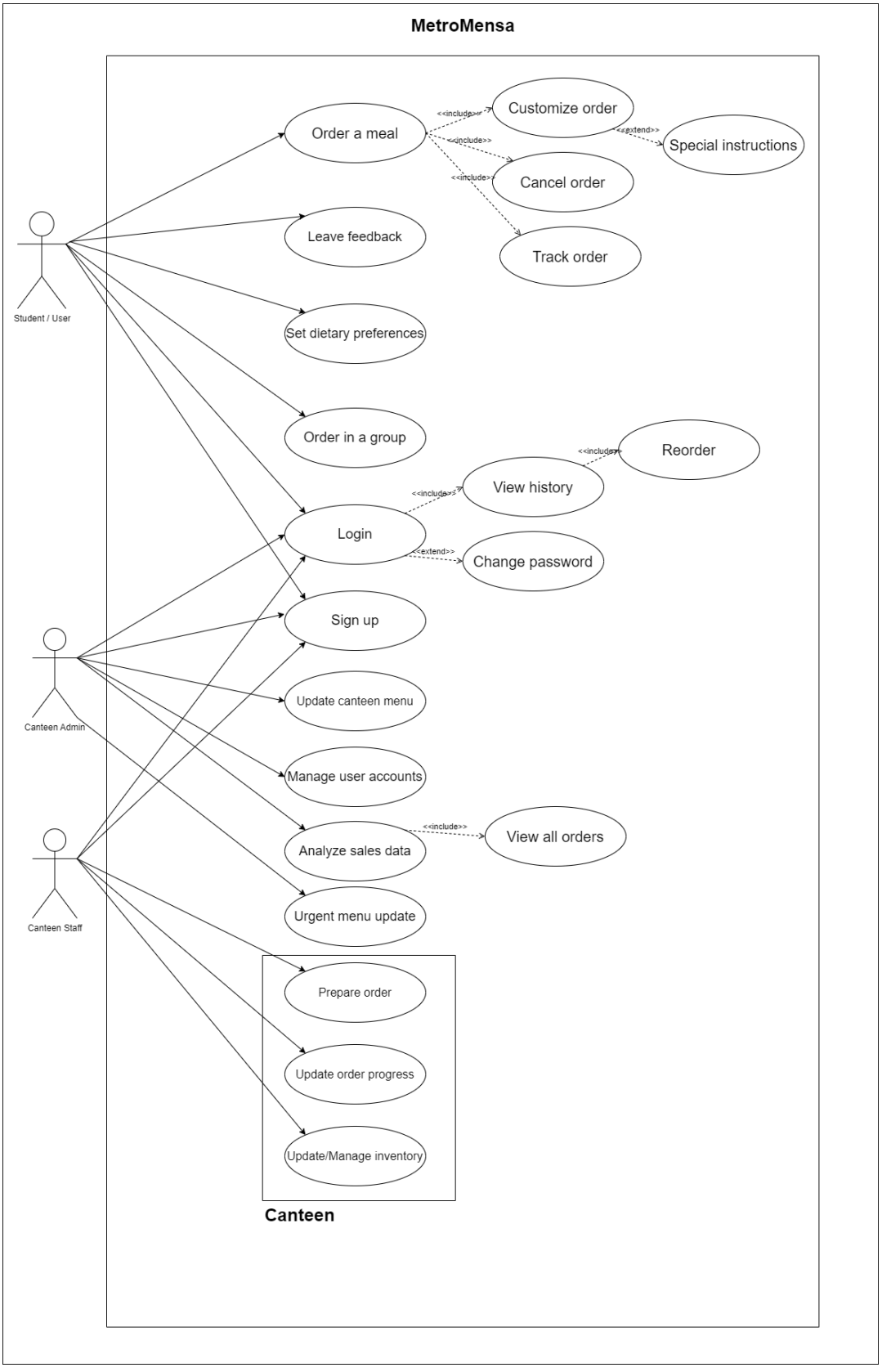
Use Case ID	UC-MM-04
Use Case name	View Menu
Scope	MetroMensa Application
Level	User Goal
Primary Actor	Student/User
Stakeholders and interests	- Student/User: Wants to view the current canteen menu to make informed meal choices. - Canteen Staff: Needs to ensure the menu displayed is up-to-date and accurate.
Preconditions	The user must be logged into the MetroMensa system. The menu must be current and updated by the canteen admin.
Postconditions	The user views the current menu available in the canteen.
Main Success Scenario	1. User accesses the MetroMensa application. 2. User navigates to the 'View Menu' section. 3. The system displays the current menu, including items and prices. 4. User browses through the menu items.

Extensions	If certain items on the menu are not available, the system indicates their unavailability. If the user has dietary preferences set in their profile, the menu highlights suitable items.
Minimum guarantees	The user is informed if the menu cannot be displayed.
Success guarantees	The user is able to view the up-to-date menu.
Trigger	User opens the 'View Menu' section in the app.
Frequency of occurrence	Several times a day, depending on user meal preferences and timings.

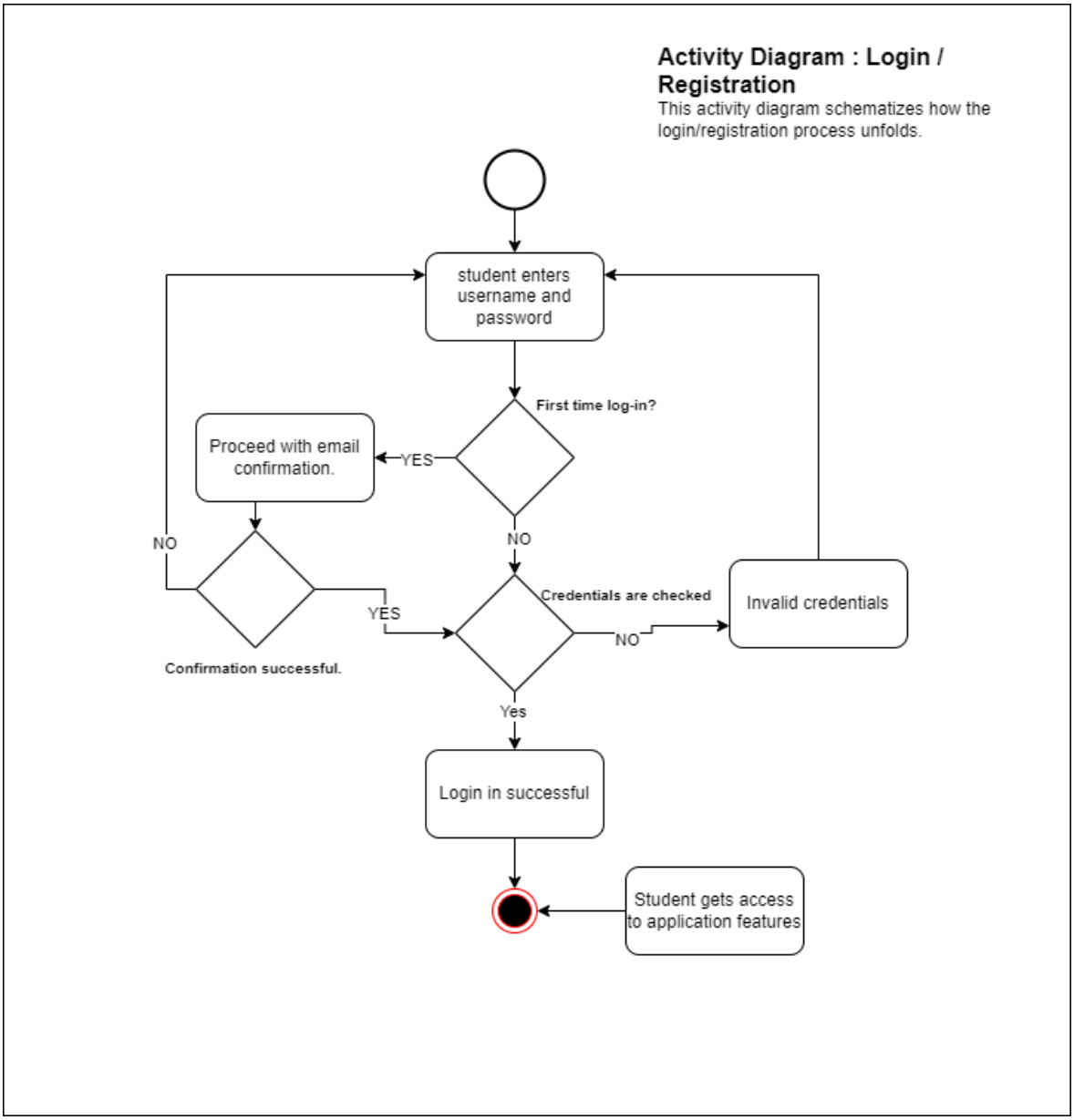
Use Case ID	UC-MM-04
Use Case name	Provide Feedback and Rating
Scope	MetroMensa Application
Level	User goal
Primary Actor	Student/User
Stakeholders and interests	- Student/User: Wants to provide feedback on meals and service for continuous improvement. - Canteen Staff: Interested in receiving feedback to enhance service quality.
Preconditions	The user must have completed at least one order through the MetroMensa system.
Postconditions	User feedback and rating are submitted and recorded in the system.
Main Success Scenario	1. User accesses the 'Feedback' section in the app after completing an order. 2. User rates the meal and/or service based on their experience.

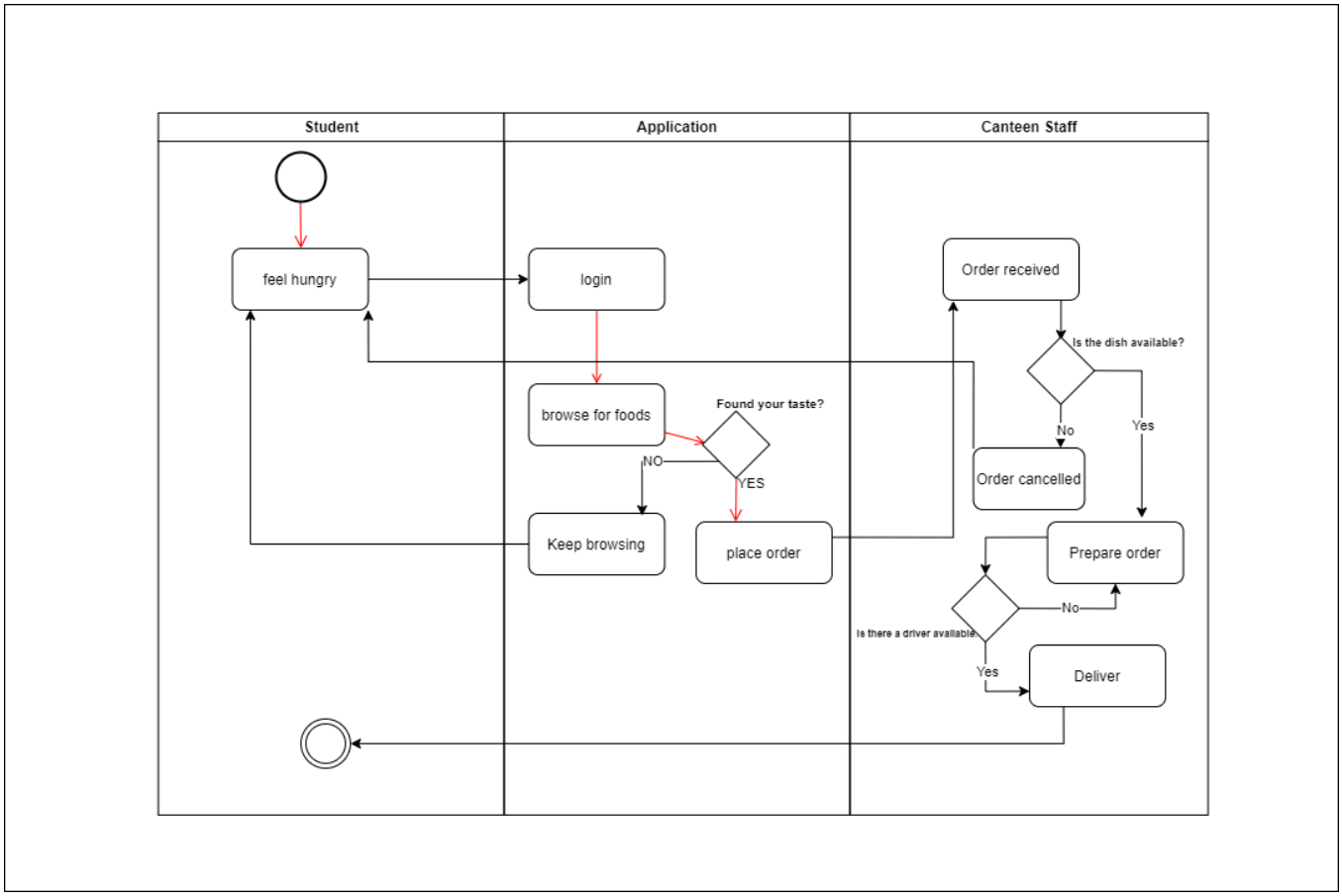
	3. User provides additional comments if desired. 4. User submits the feedback. 5. The system records the feedback and acknowledges the submission to the user.
Extensions	If the user chooses not to provide additional comments, the system still accepts the rating. 3b. If the user encounters technical issues while submitting, the system prompts to resubmit.
Minimum guarantees	The user is able to access the feedback section and leave a rating.
Success guarantees	The feedback is successfully submitted and recorded.
Trigger	Completion of a meal order by the user.
Frequency of occurrence	As often as the user completes an order and chooses to provide feedback.

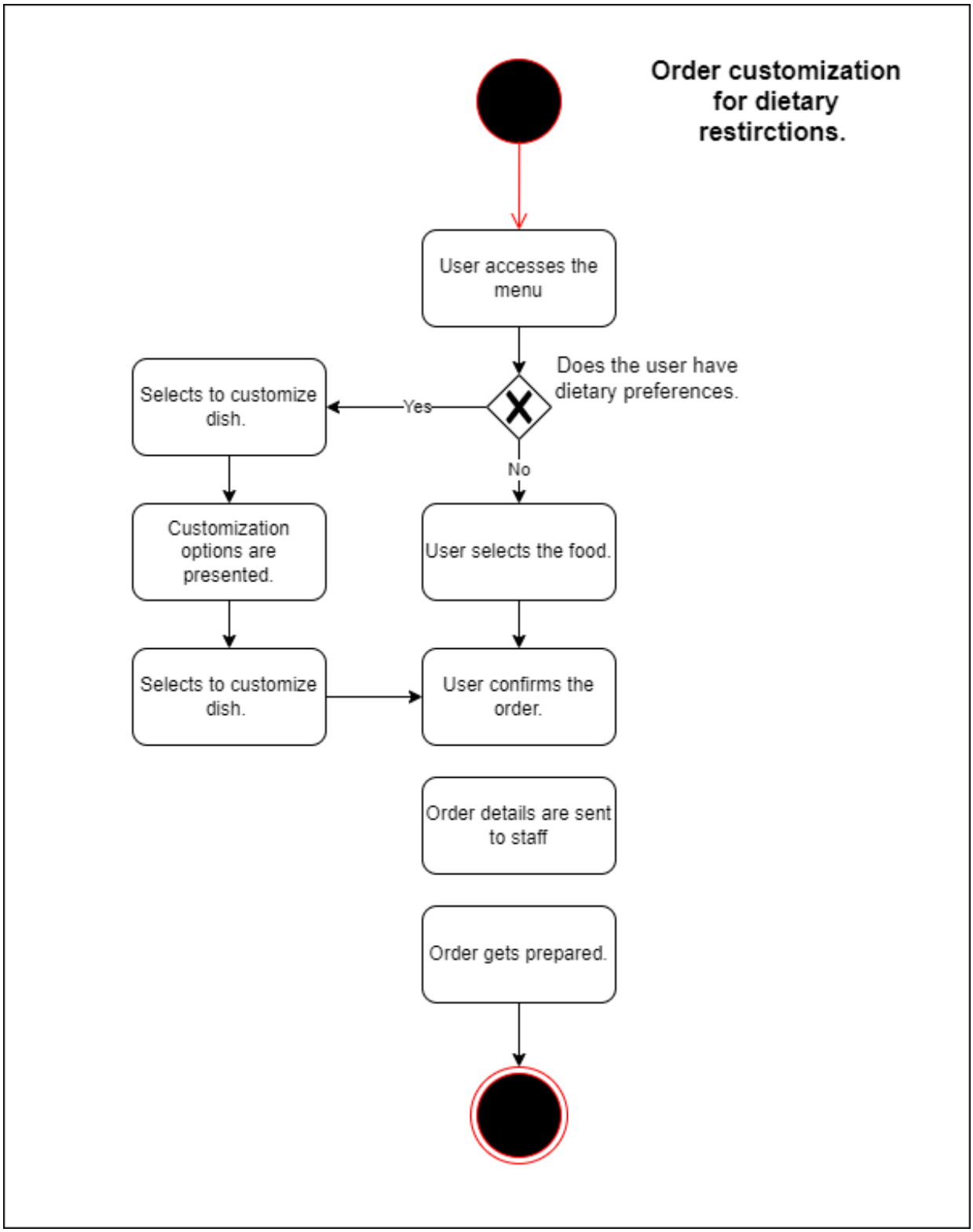
c. Use Case Diagram.

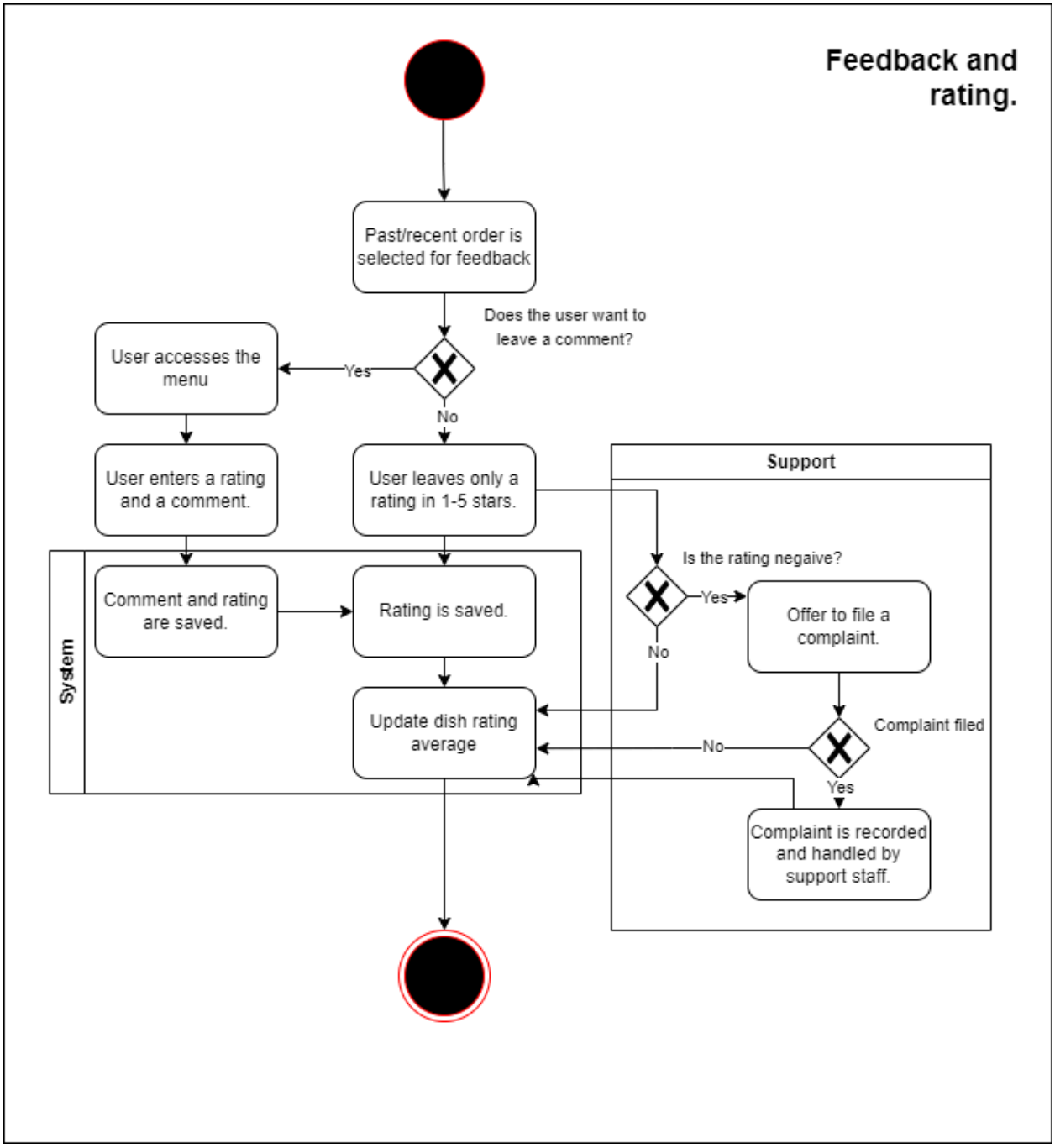


d. Activity Diagram



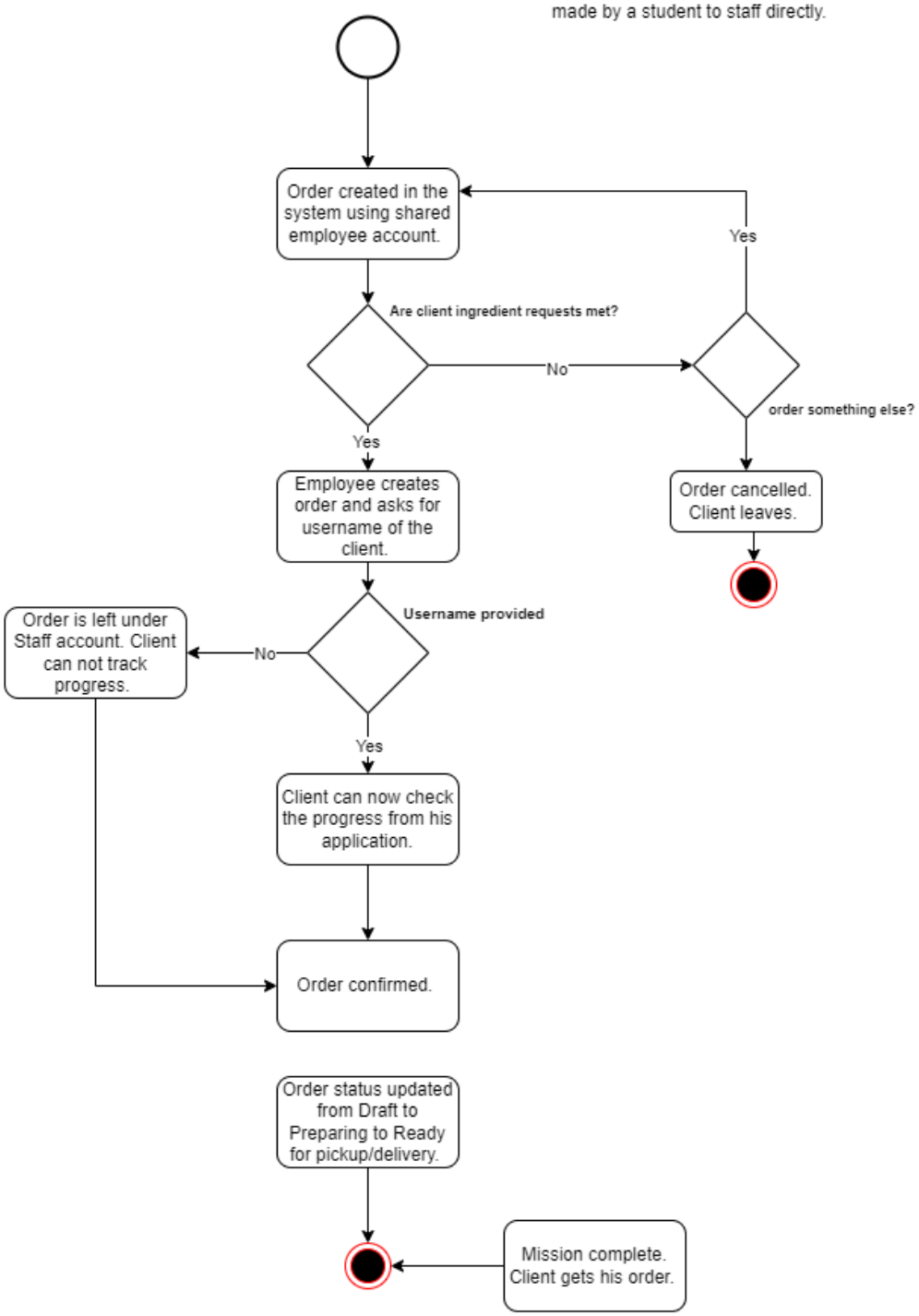

Student
Application
Canteen Staff
feel hungry
login
Order received
Is the dish available?
No
Yes
browse for foods
Found your taste?
NO
YES
Order cancelled
Keep browsing
place order
Prepare order
No
Is there a driver available
Yes
Deliver


Order customization for dietary restirctions.
User accesses the menu
Does the user have dietary preferences.
Yes
No
Selects to customize dish.
Customization options are presented.
Selects to customize dish.
User selects the food.
User confirms the order.
Order details are sent to staff
Order gets prepared.

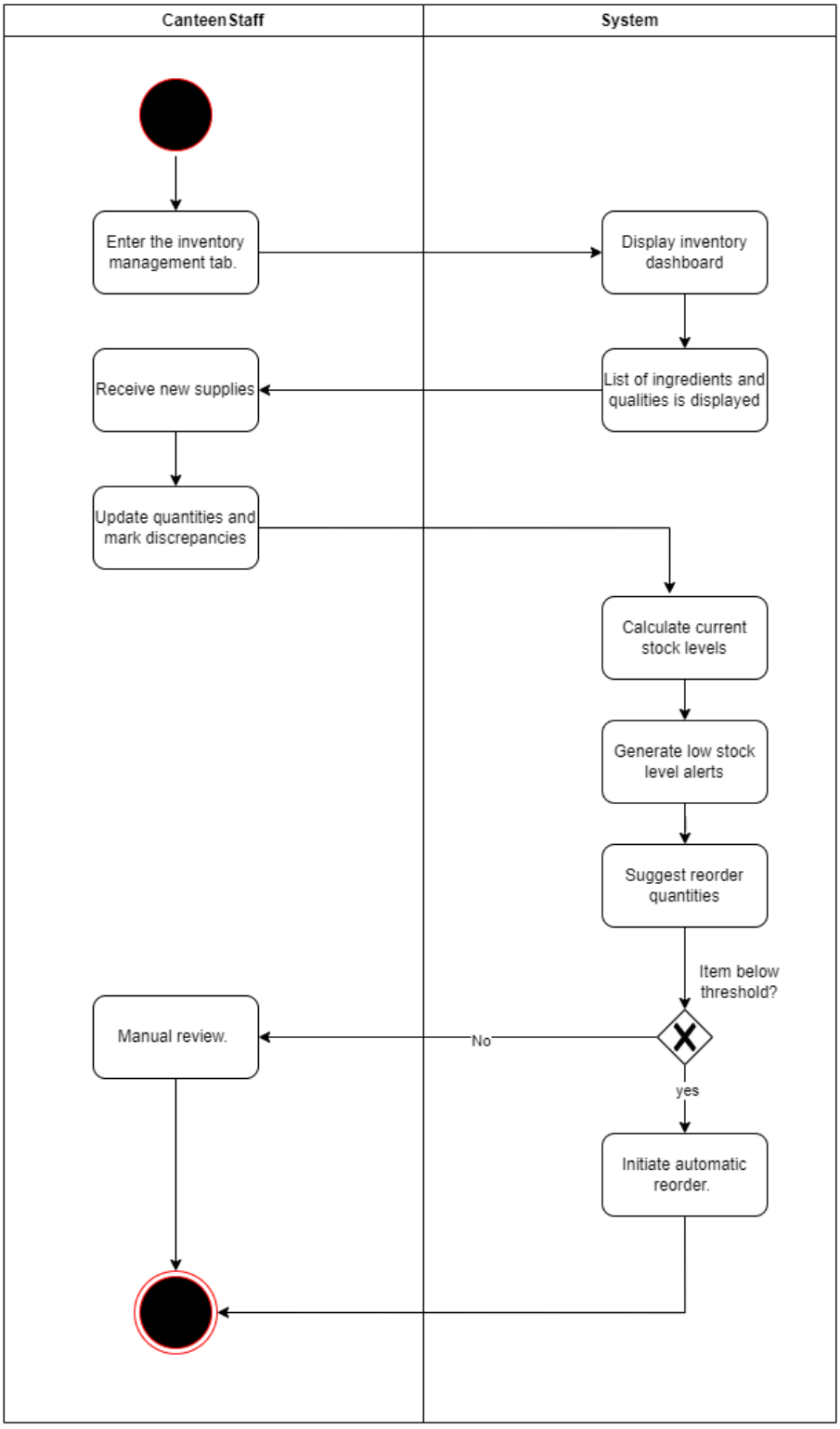

Feedback and rating.
Past/recent order is selected for feedback
Does the user want to leave a comment?
Yes
User accesses the menu
No
User enters a rating and a comment.
User leaves only a rating in 1-5 stars.
Support
System
Is the rating negaive?
Yes
Offer to file a complaint.
No
Comment and rating are saved.
Rating is saved.
Update dish rating average
Complaint filed
No
Yes
Complaint is recorded and handled by support staff.

Activity Diagram : Over the counter order.

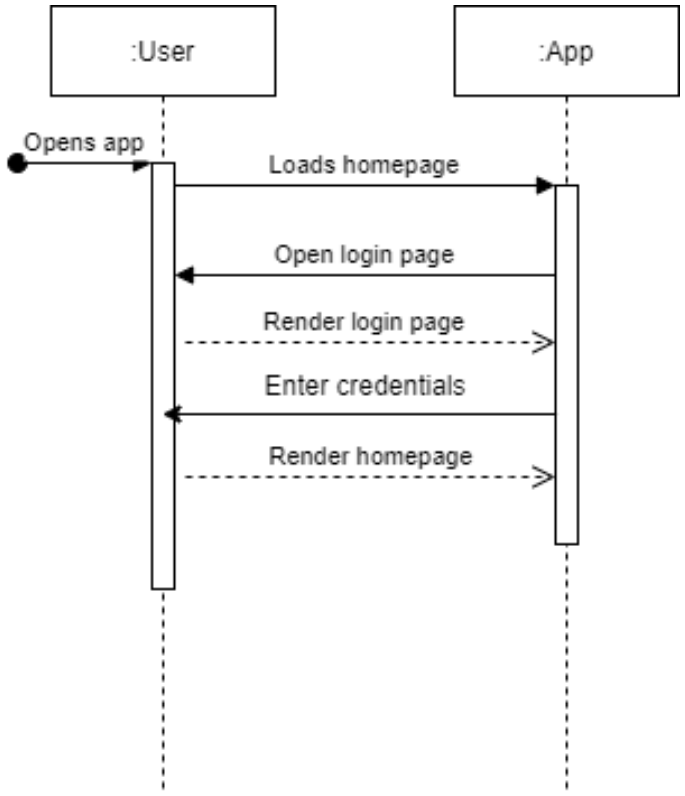
Activity diagram for an over the counter order made by a student to staff directly.

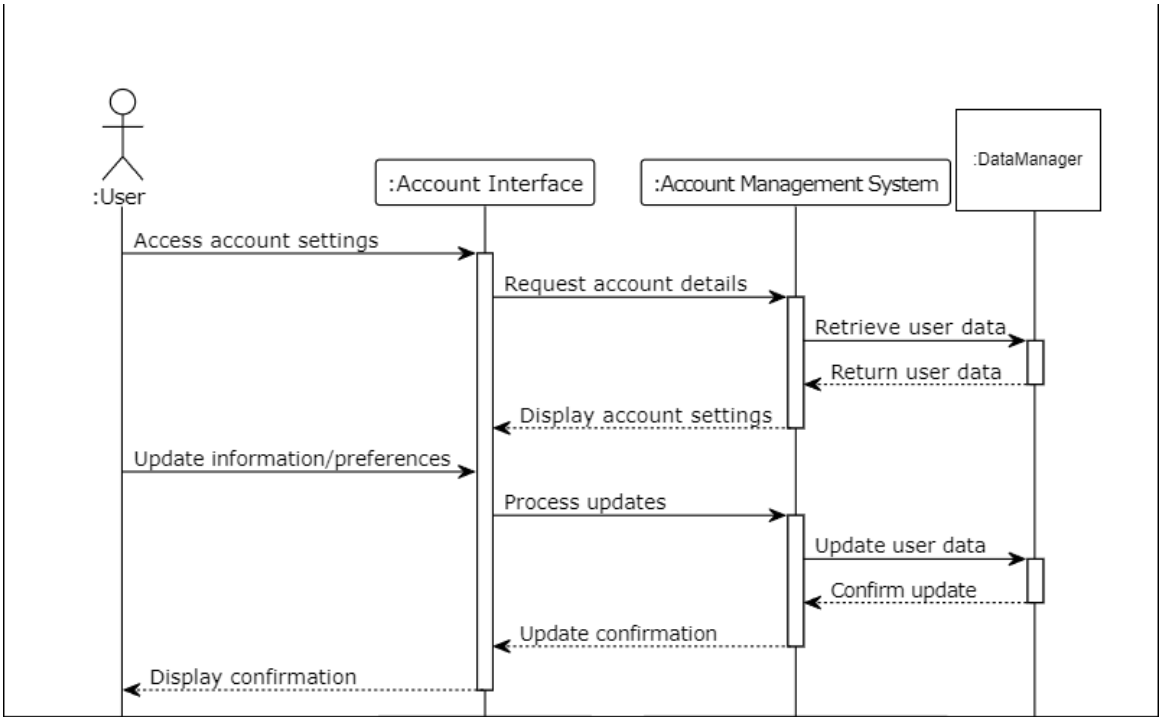


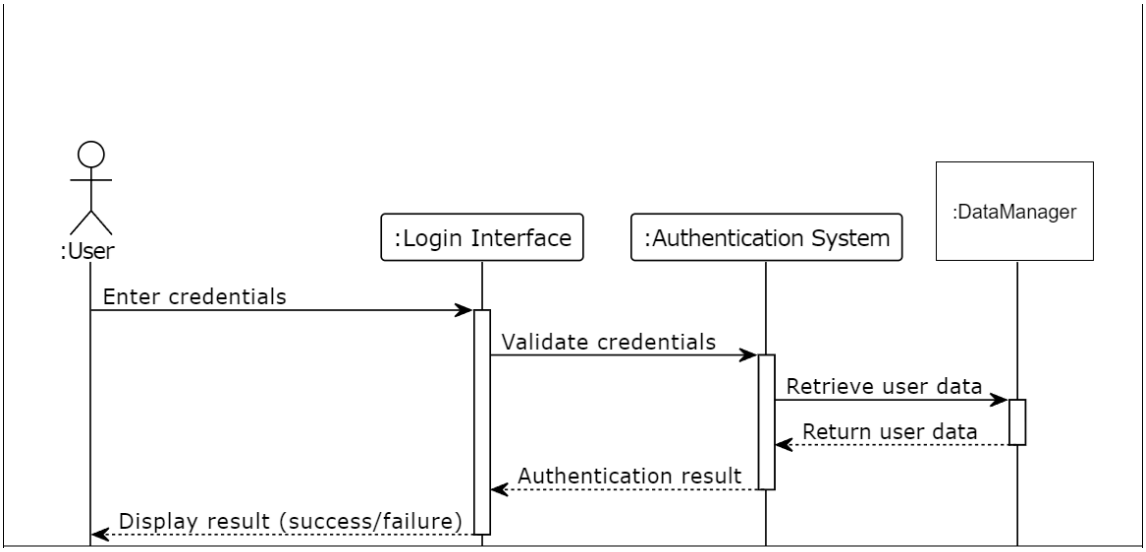
Inventory checking

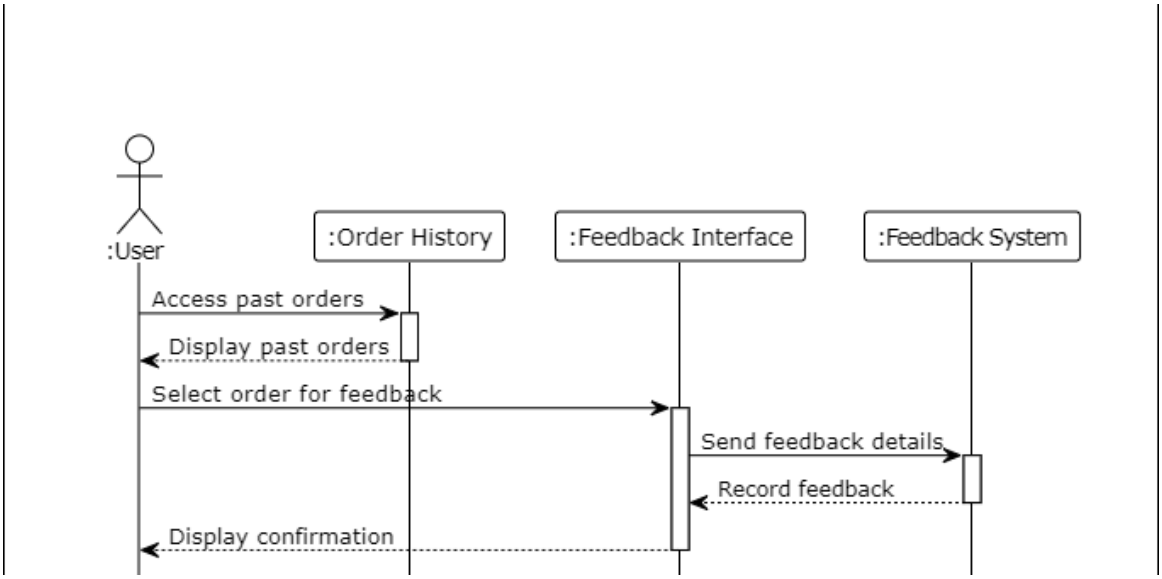


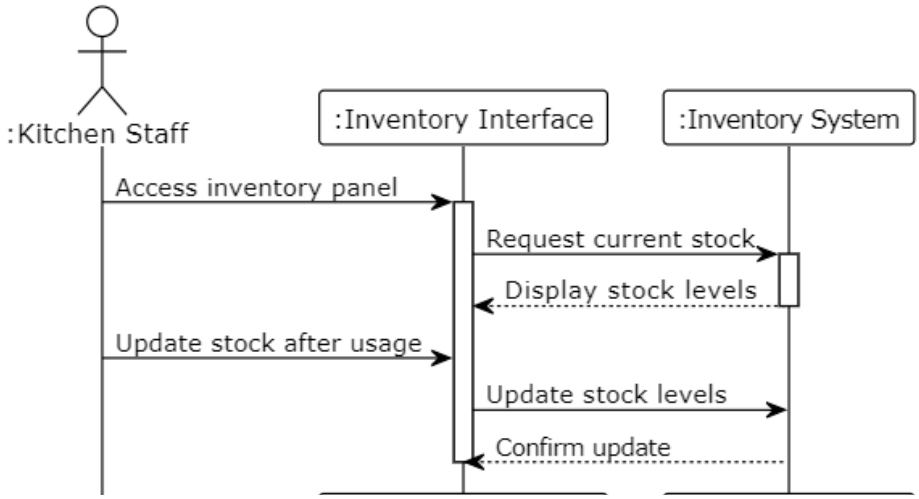
e. Sequence Diagram

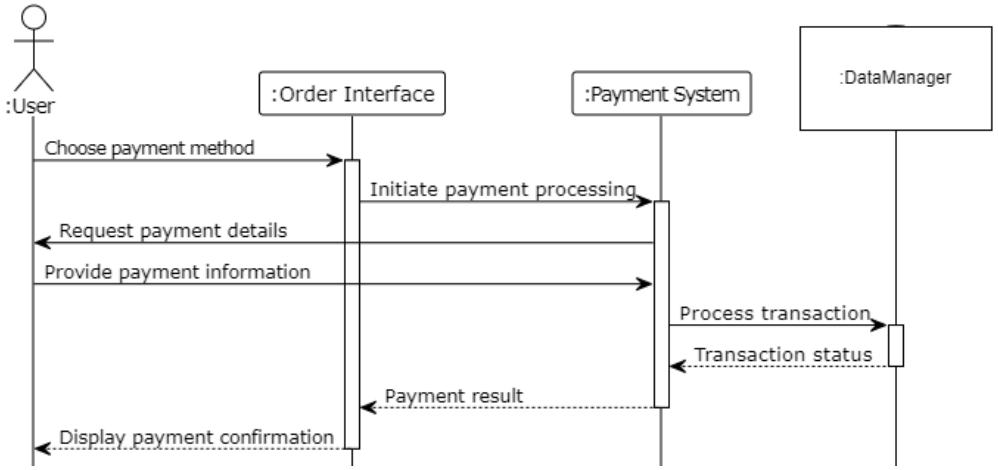


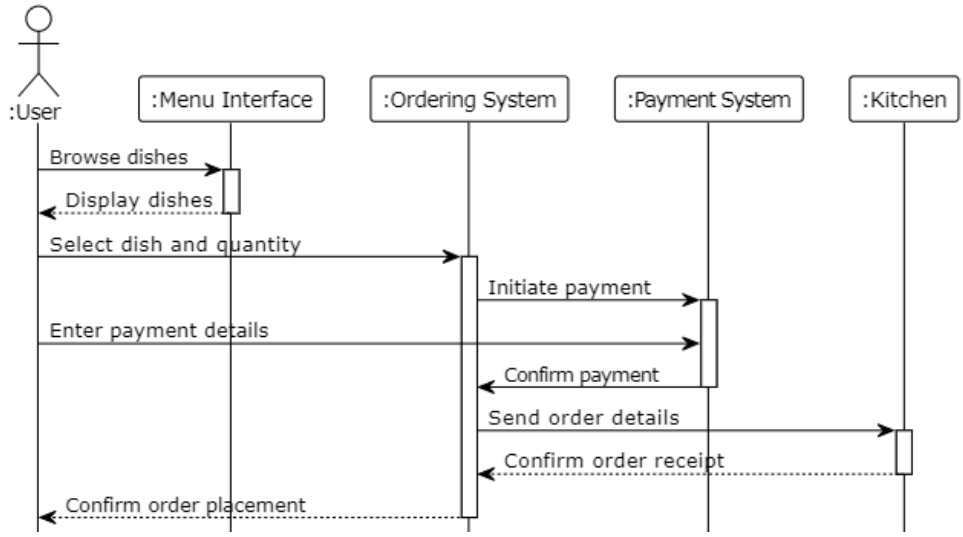



:User
:Login Interface
:Authentication System
:DataManager
Enter credentials
Validate credentials
Retrieve user data
Return user data
Authentication result
Display result (success/failure)

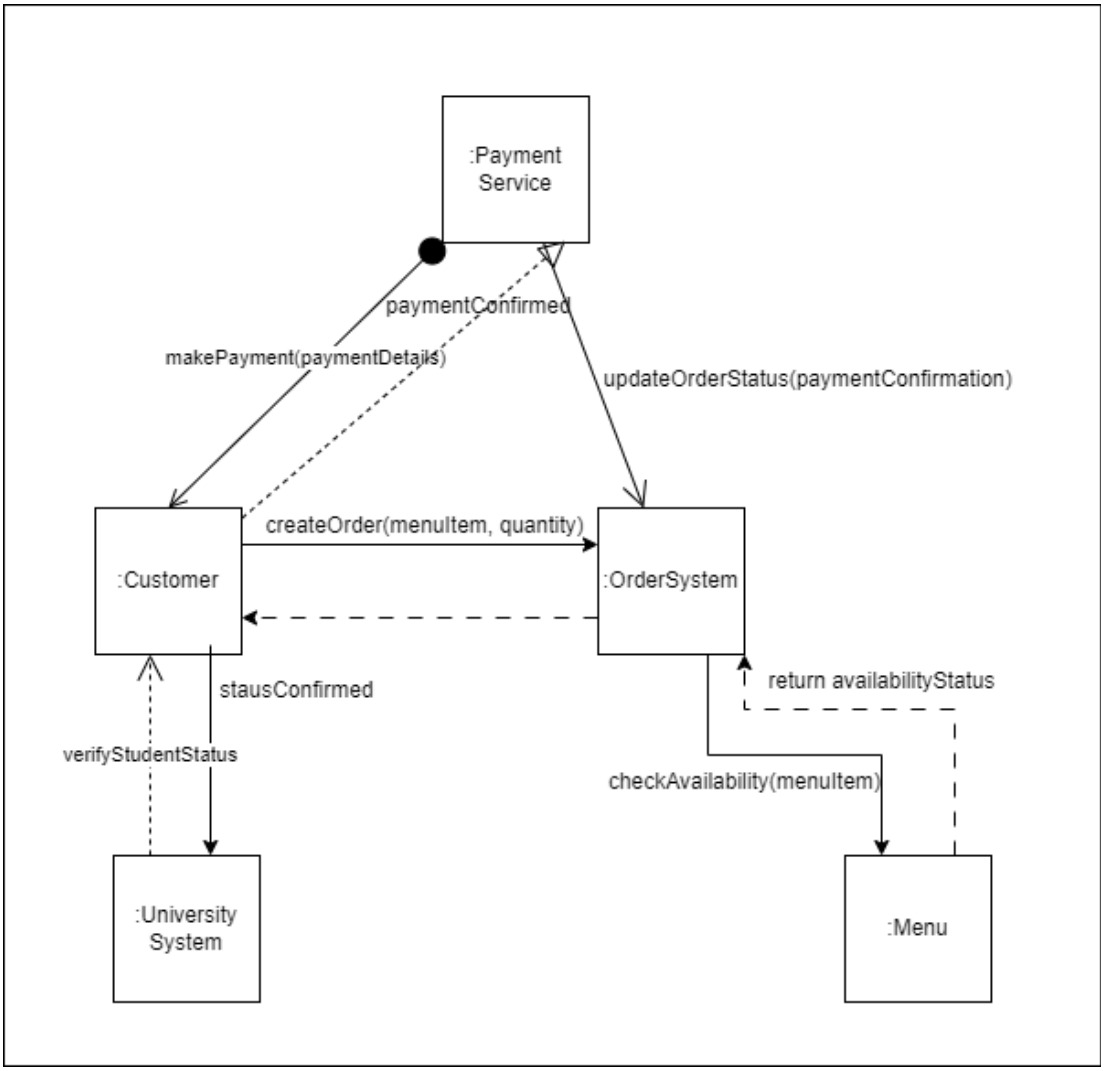

:User
:Order History
:Feedback Interface
:Feedback System
Access past orders
Display past orders
Select order for feedback
Send feedback details
Record feedback
Display confirmation

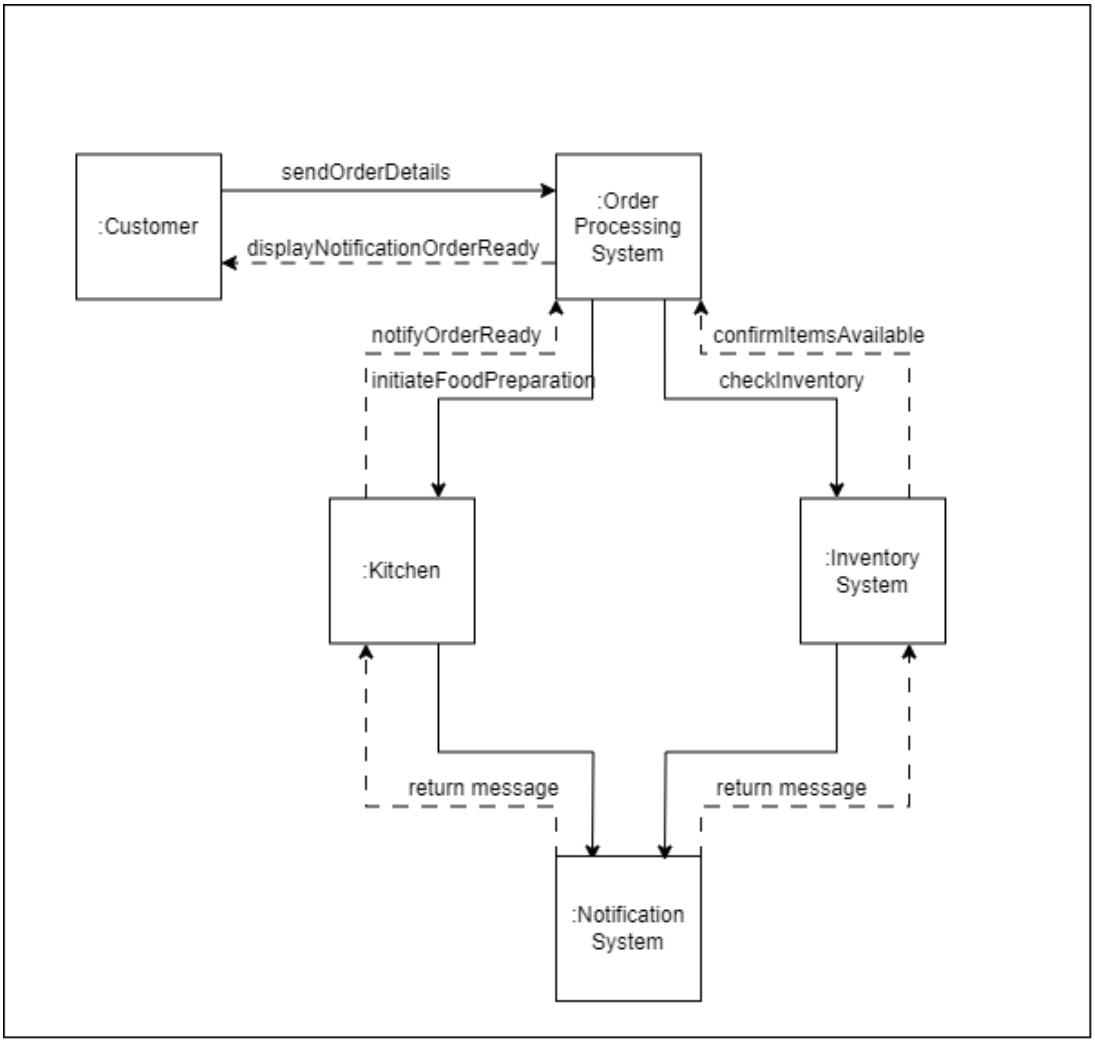

:Kitchen Staff
:Inventory Interface
:Inventory System
Access inventory panel
Request current stock
Display stock levels
Update stock after usage
Update stock levels
Confirm update

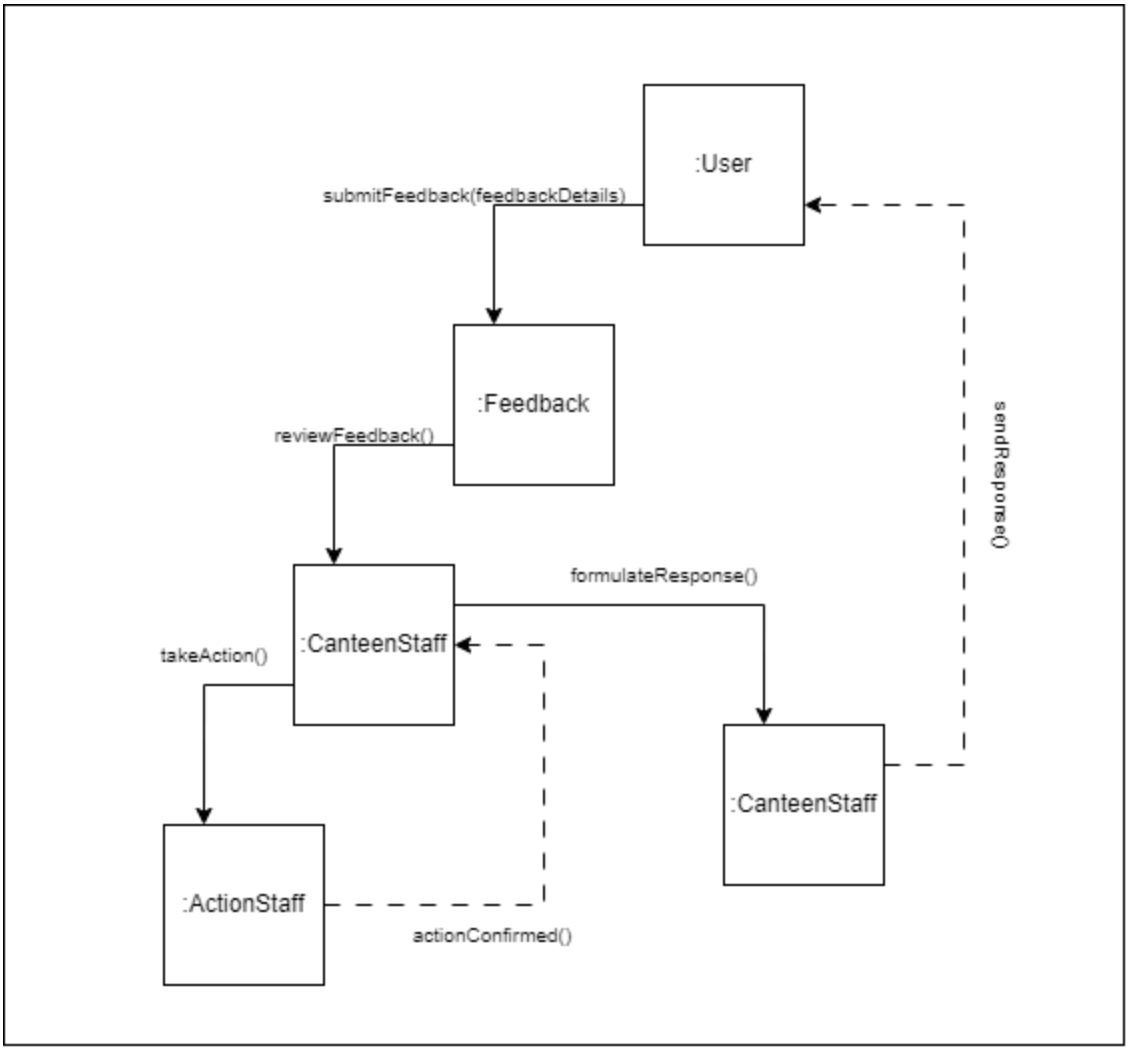

:User
:Order Interface
:Payment System
:DataManager
Choose payment method
Initiate payment processing
Request payment details
Provide payment information
Process transaction
Transaction status
Payment result
Display payment confirmation

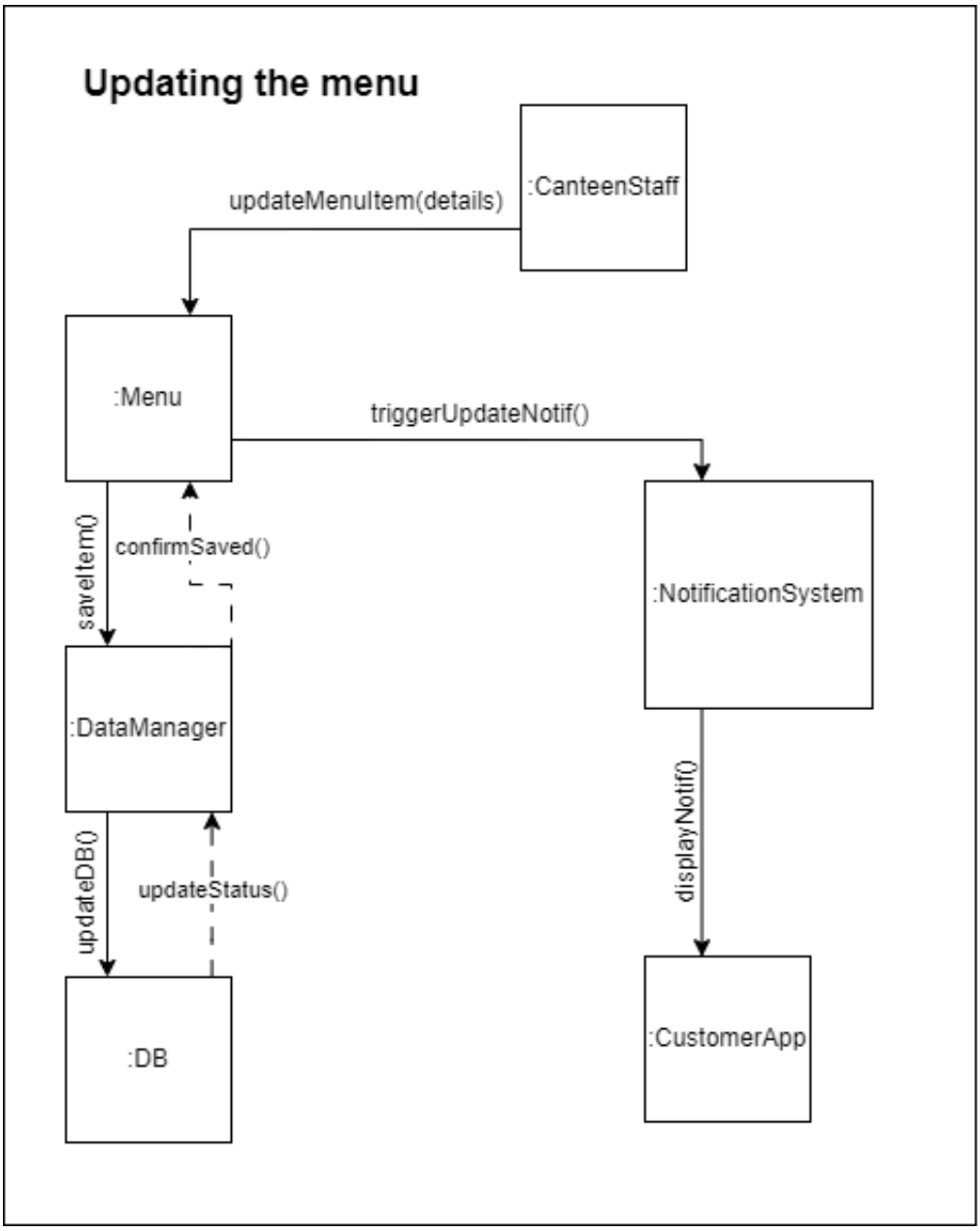

:User
:Menu Interface
:Ordering System
:Payment System
:Kitchen
Browse dishes
Display dishes
Select dish and quantity
Initiate payment
Enter payment details
Confirm payment
Send order details
Confirm order receipt
Confirm order placement

f. Collaboration Diagram

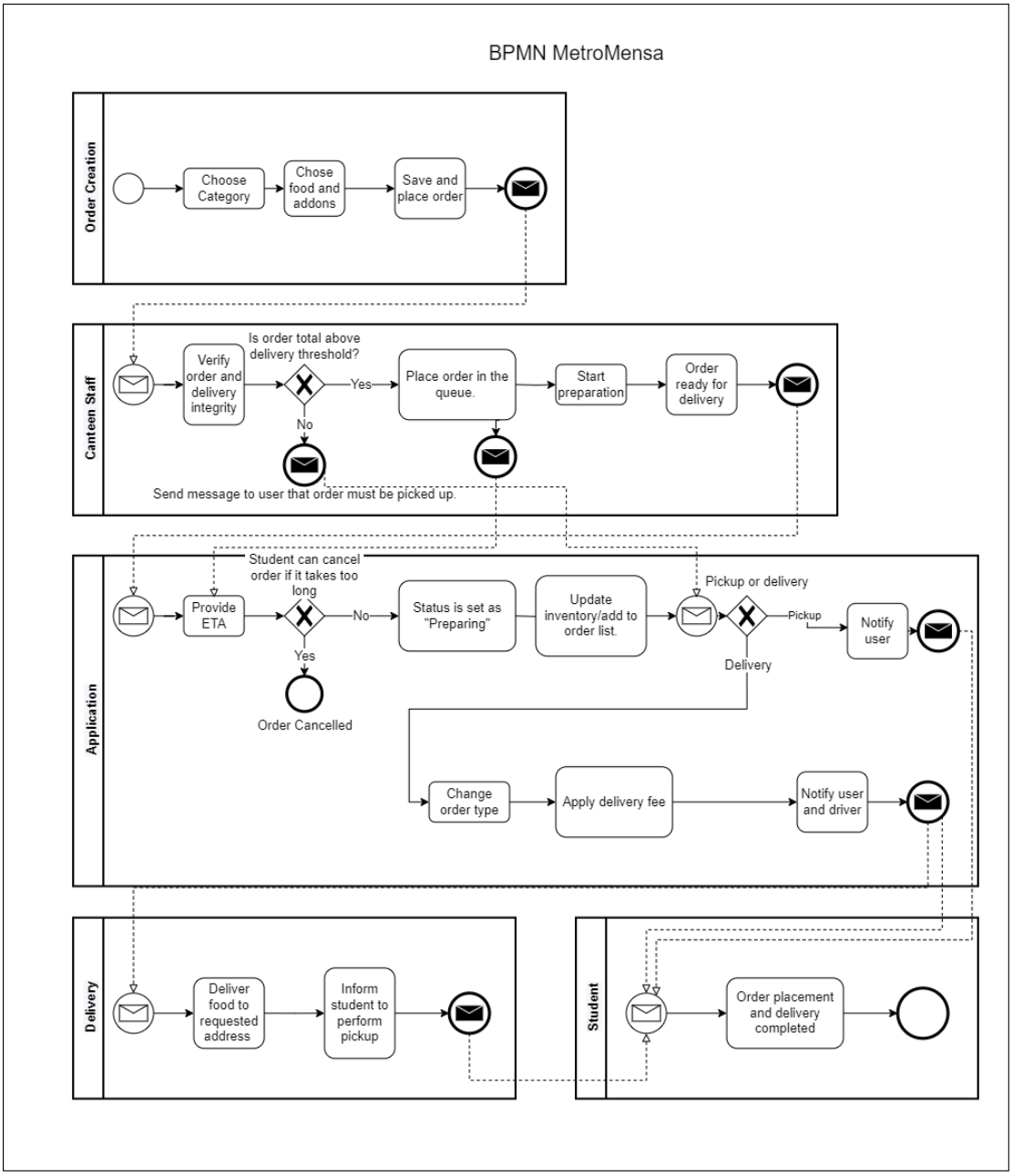



:Customer
sendOrderDetails
displayNotificationOrderReady
:Order Processing System
notifyOrderReady
initiateFoodPreparation
confirmItemsAvailable
checkInventory
:Kitchen
:Inventory System
return message
return message
:Notification System

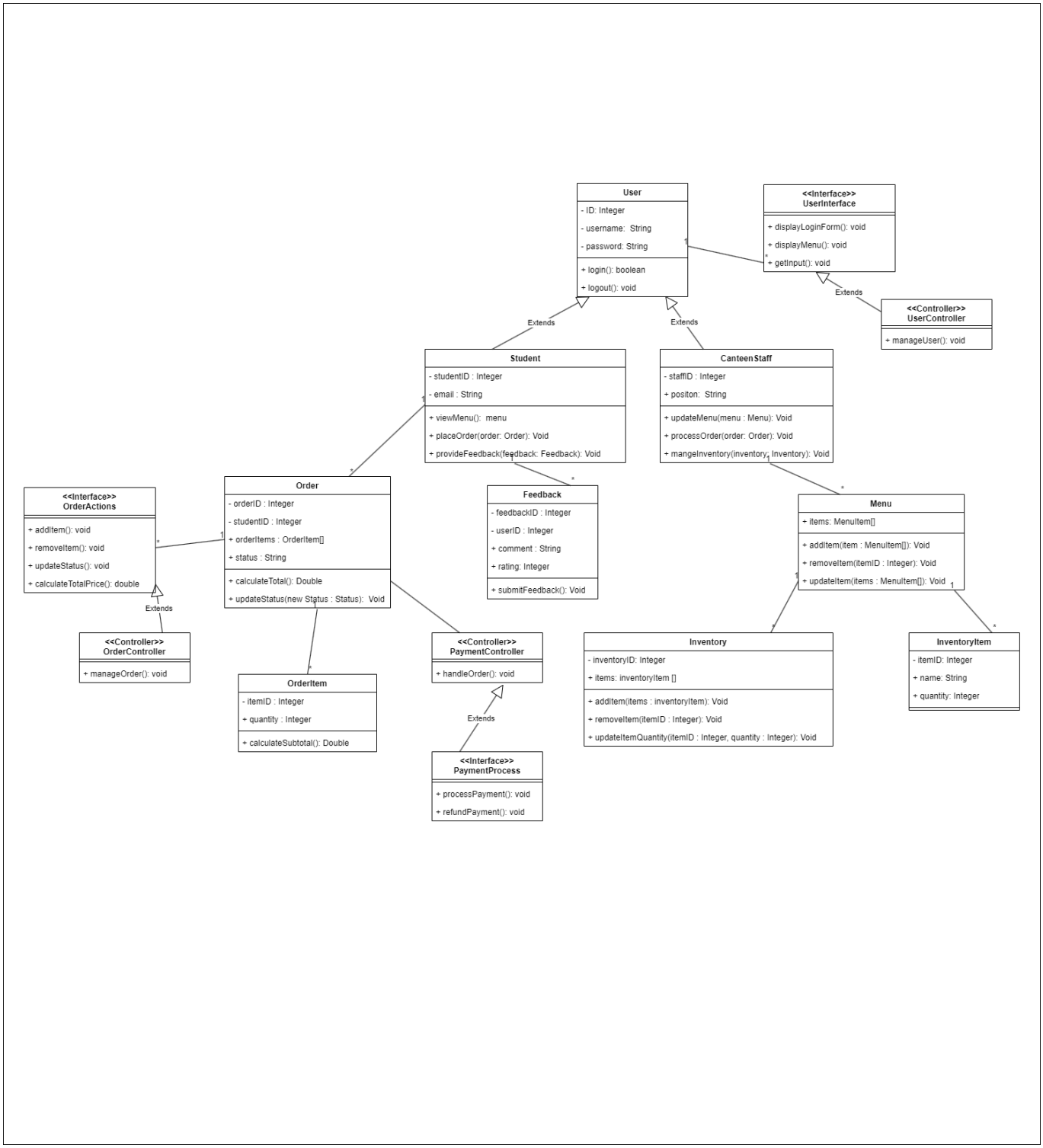

:User
submitFeedback(feedbackDetails)
:Feedback
reviewFeedback()
sendResponse()
formulateResponse()
:CanteenStaff
takeAction()
:CanteenStaff
:ActionStaff
actionConfirmed()

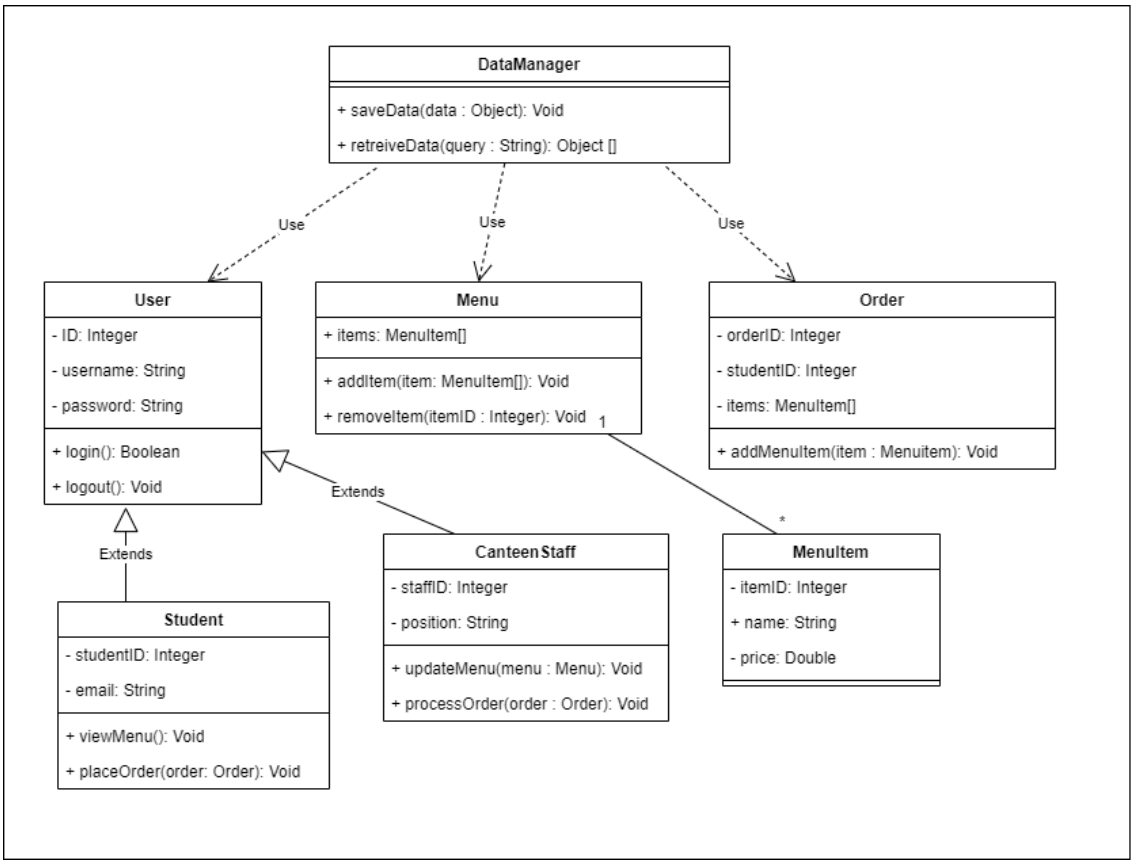

Updating the menu
:CanteenStaff
updateMenuItem(details)
:Menu
triggerUpdateNotif()
saveItem()
confirmSaved()
:NotificationSystem
:DataManager
displayNotif()
updateDB()
updateStatus()
:DB
:CustomerApp

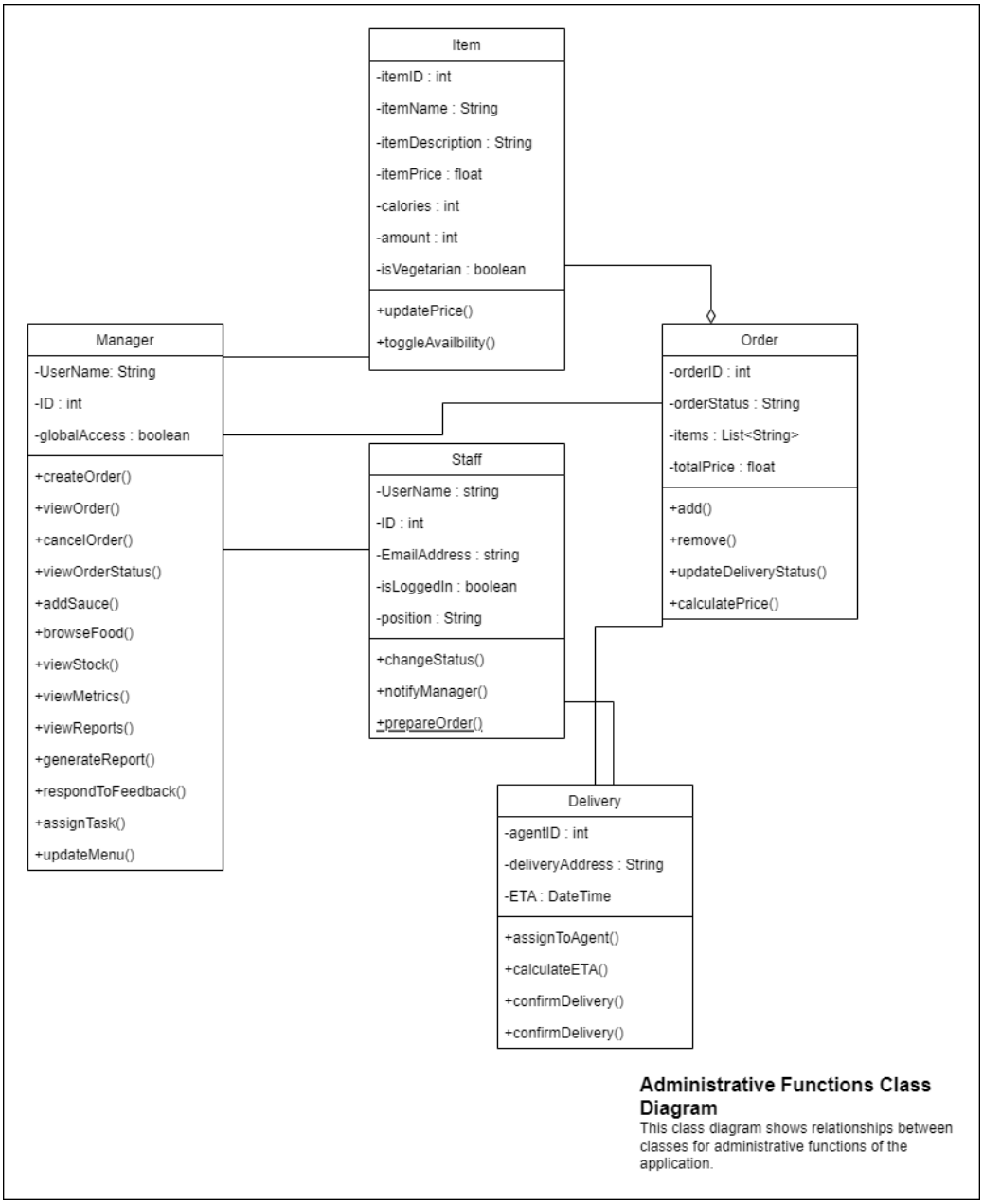
g. BPMN Diagram

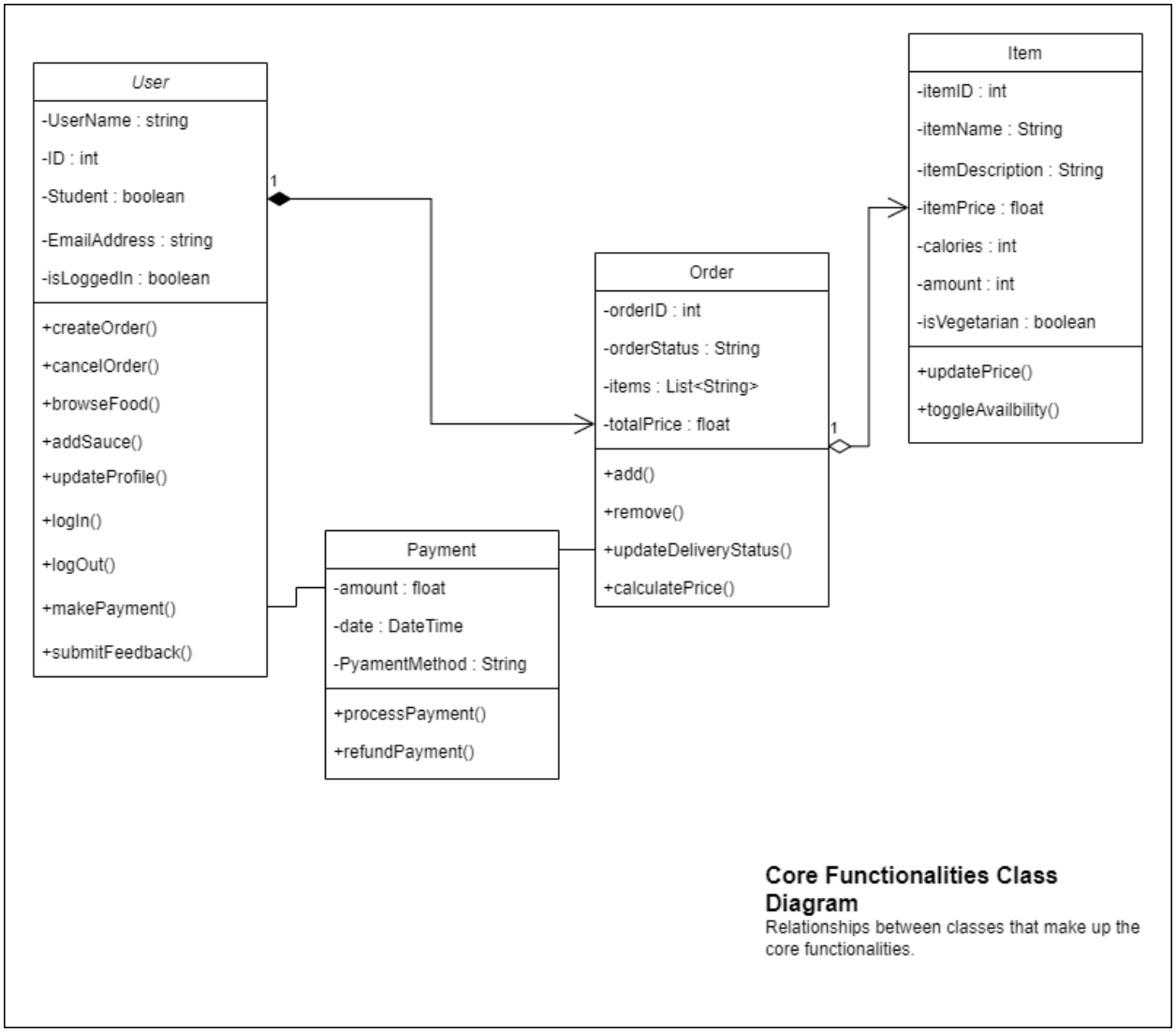


h. Class Diagram

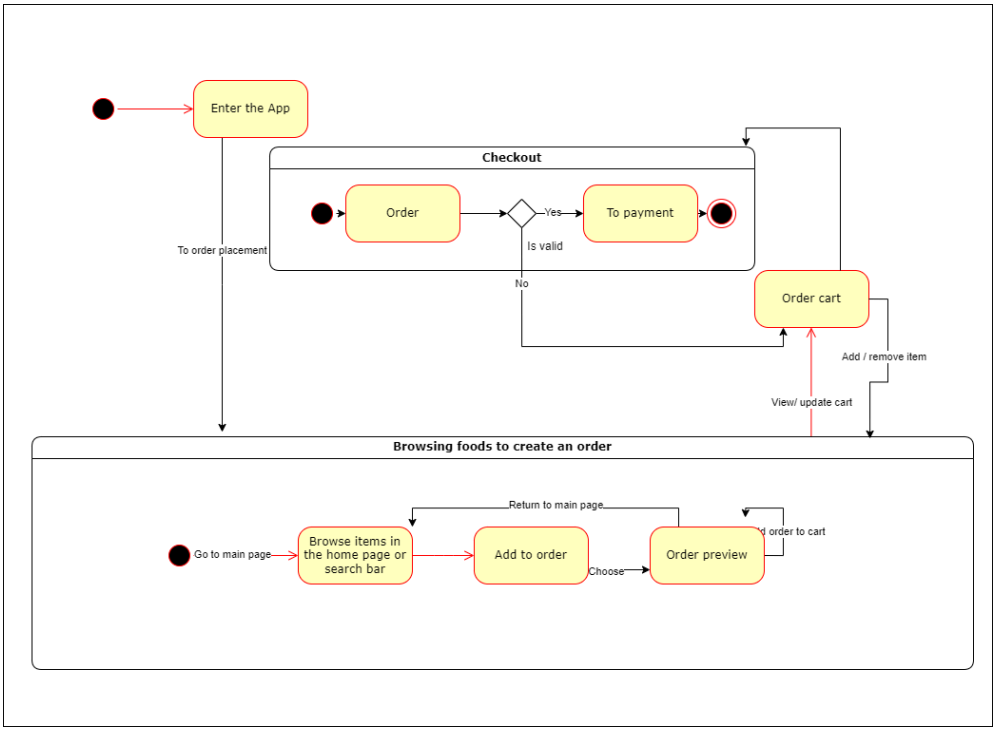


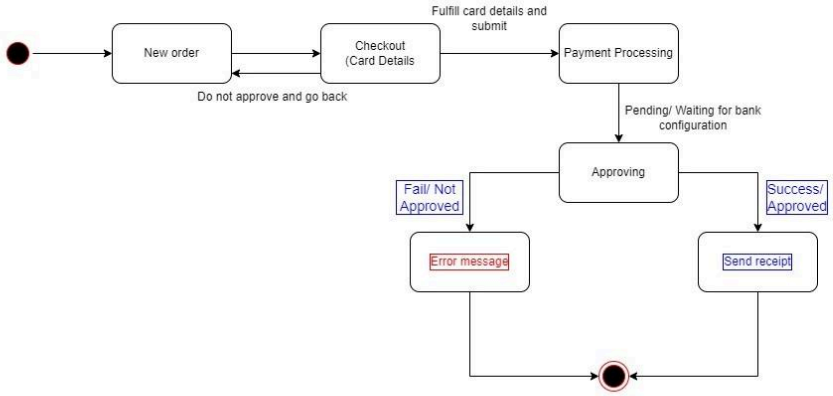

DataManager
+ saveData(data : Object): Void
+ retreiveData(query : String): Object []
Use
Use
Use
User
- ID: Integer
- username: String
- password: String
+ login(): Boolean
+ logout(): Void
Menu
+ items: MenuItem[]
+ addItem(item: MenuItem[]): Void
+ removeItem(itemID : Integer): Void
Order
- orderID: Integer
- studentID: Integer
- items: MenuItem[]
+ addMenuItem(item : Menuitem): Void
Extends
Extends
1
*
Student
- studentID: Integer
- email: String
+ viewMenu(): Void
+ placeOrder(order: Order): Void
CanteenStaff
- staffID: Integer
- position: String
+ updateMenu(menu : Menu): Void
+ processOrder(order : Order): Void
MenuItem
- itemID: Integer
+ name: String
- price: Double


Item
-itemID : int
-itemName : String
-itemDescription : String
-itemPrice : float
-calories : int
-amount : int
-isVegetarian : boolean
+updatePrice()
+toggleAvailbility()
Manager
-UserName: String
-ID : int
-globalAccess : boolean
+createOrder()
+viewOrder()
+cancelOrder()
+viewOrderStatus()
+addSauce()
+browseFood()
+viewStock()
+viewMetrics()
+viewReports()
+generateReport()
+respondToFeedback()
+assignTask()
+updateMenu()
Staff
-UserName : string
-ID : int
-EmailAddress : string
-isLoggedIn : boolean
-position : String
+changeStatus()
+notifyManager()
+prepareOrder()
Order
-orderID : int
-orderStatus : String
-items : List<String>
-totalPrice : float
+add()
+remove()
+updateDeliveryStatus()
+calculatePrice()
Delivery
-agentID : int
-deliveryAddress : String
-ETA : DateTime
+assignToAgent()
+calculateETA()
+confirmDelivery()
+confirmDelivery()
Administrative Functions Class Diagram
This class diagram shows relationships between classes for administrative functions of the application.

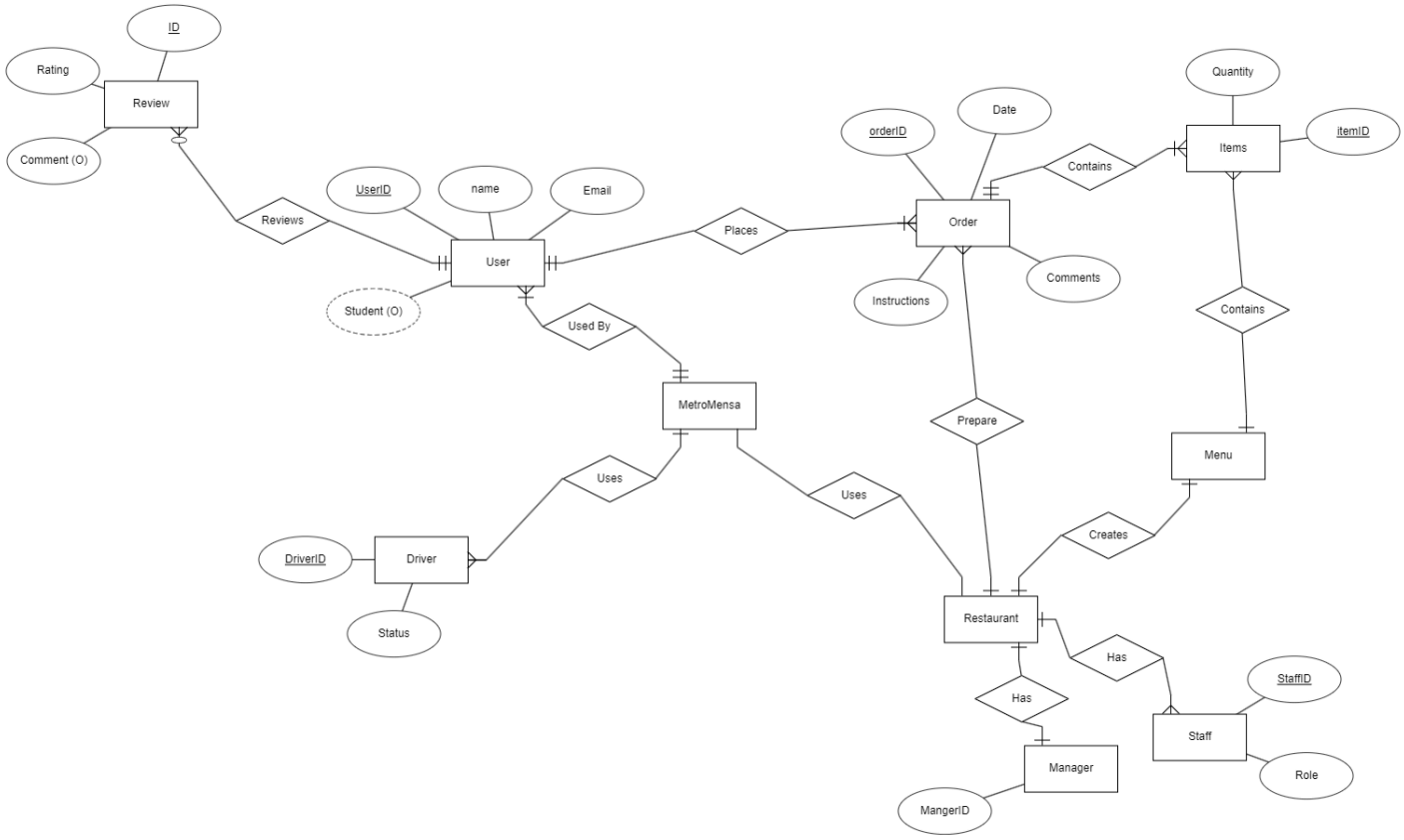

User
-UserName : string
-ID : int
-Student : boolean
-EmailAddress : string
-isLoggedIn : boolean
+createOrder()
+cancelOrder()
+browseFood()
+addSauce()
+updateProfile()
+logIn()
+logOut()
+makePayment()
+submitFeedback()
1
Item
-itemID : int
-itemName : String
-itemDescription : String
-itemPrice : float
-calories : int
-amount : int
-isVegetarian : boolean
+updatePrice()
+toggleAvailbility()
Order
-orderID : int
-orderStatus : String
-items : List<String>
-totalPrice : float
+add()
+remove()
+updateDeliveryStatus()
+calculatePrice()
1
Payment
-amount : float
-date : DateTime
-PyamentMethod : String
+processPayment()
+refundPayment()
Core Functionalities Class Diagram
Relationships between classes that make up the core functionalities.

i. State Diagram

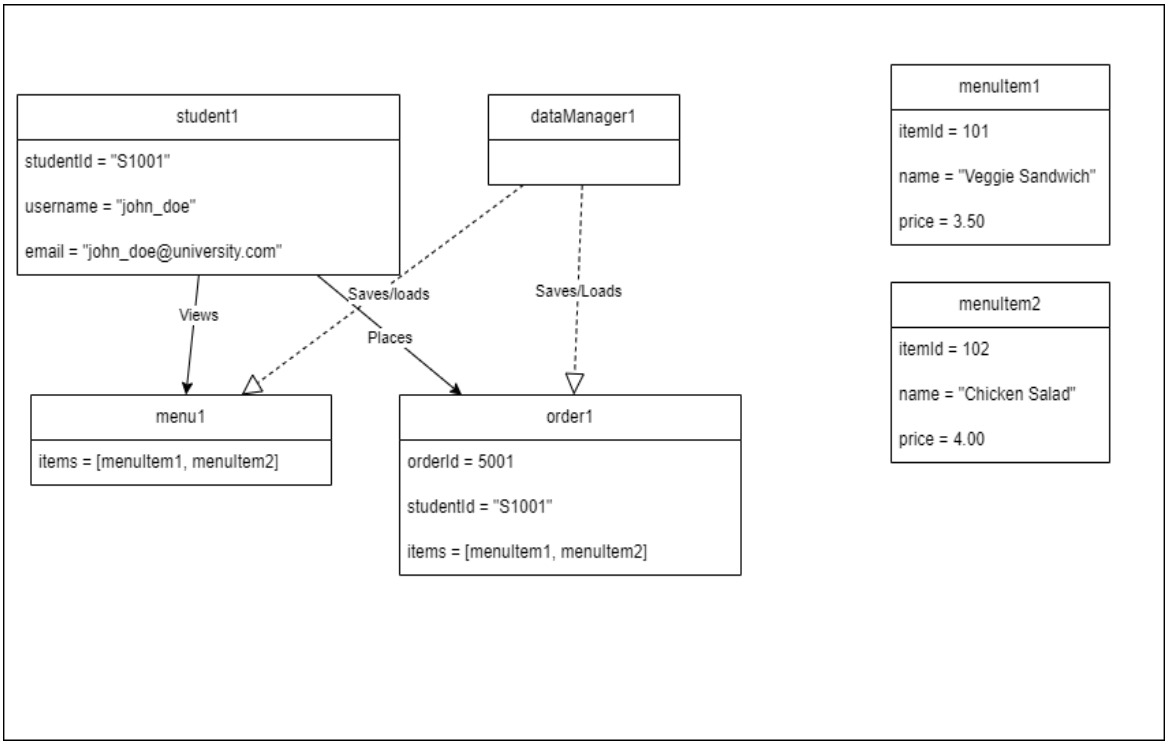


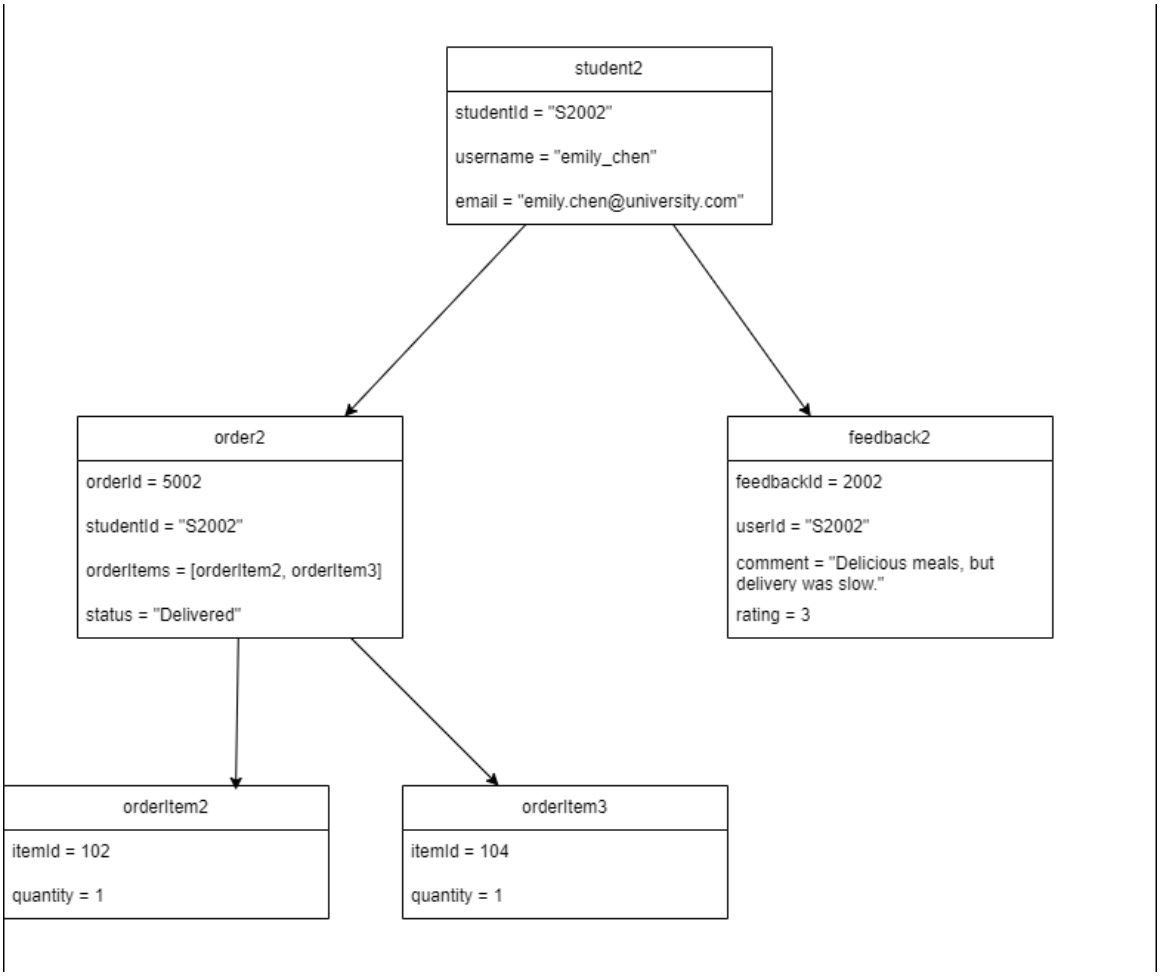


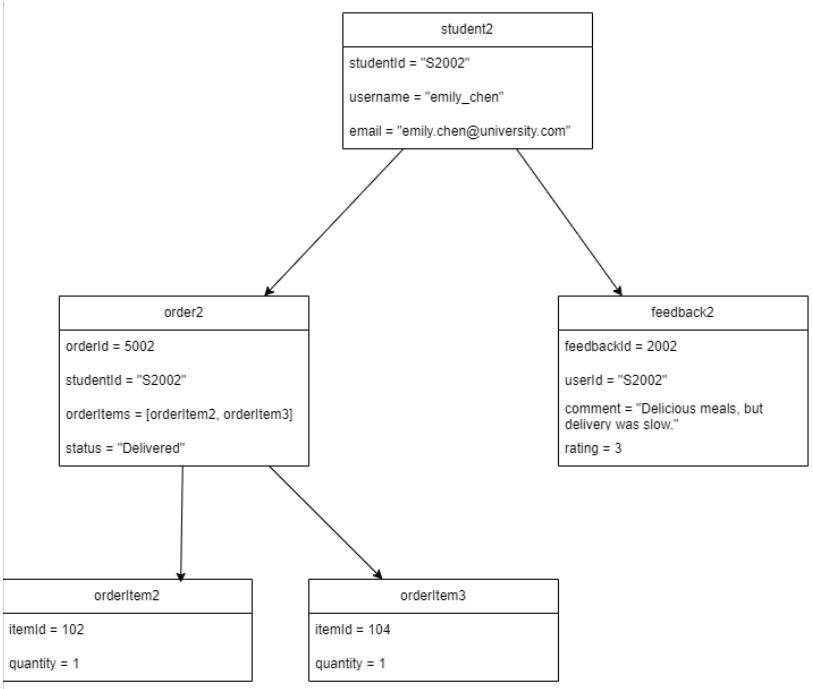
j. Entity Relation Diagram

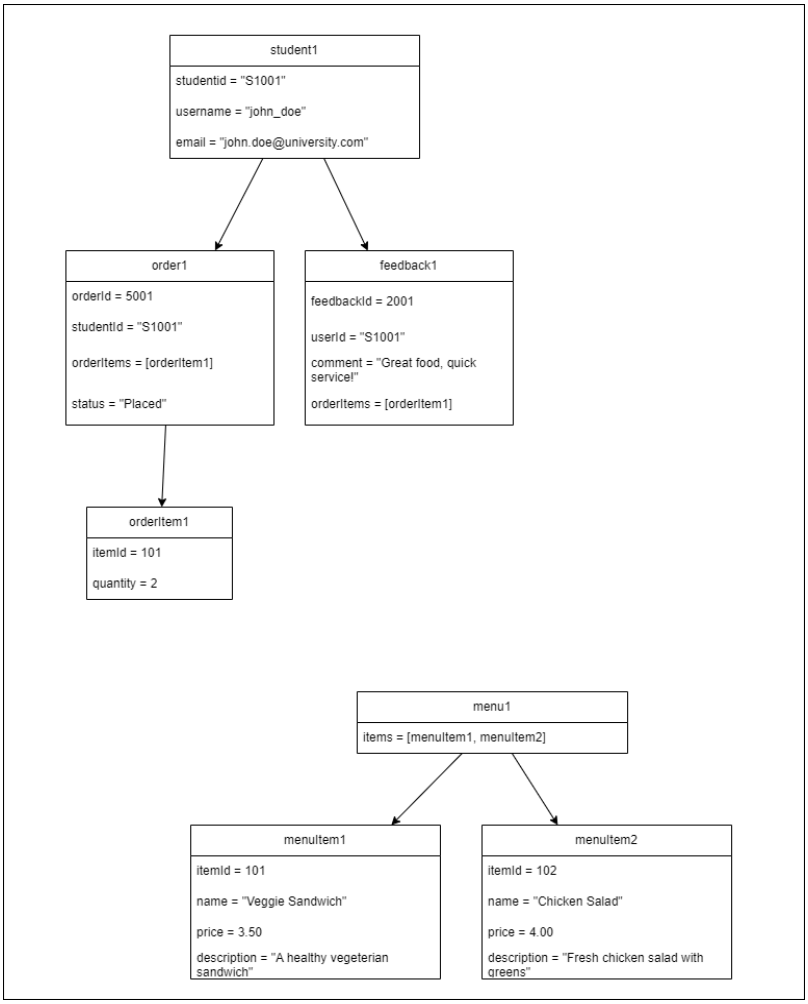


k. Object Diagram

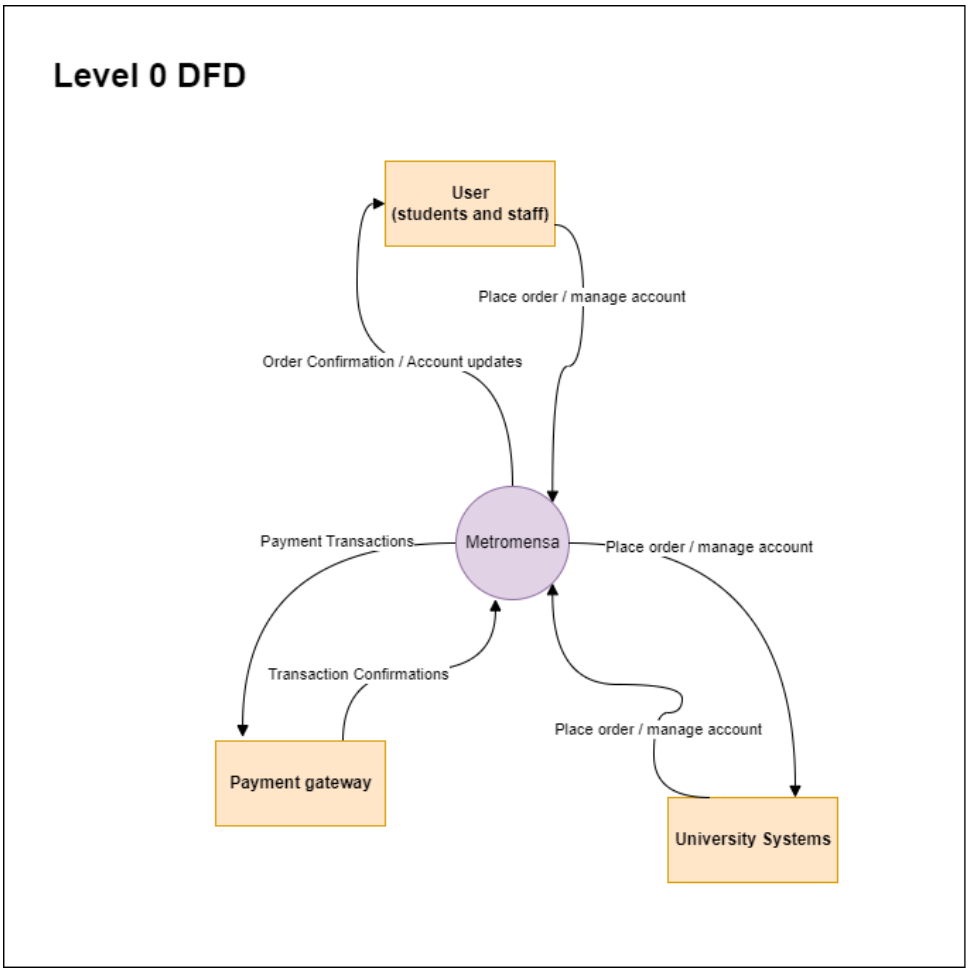


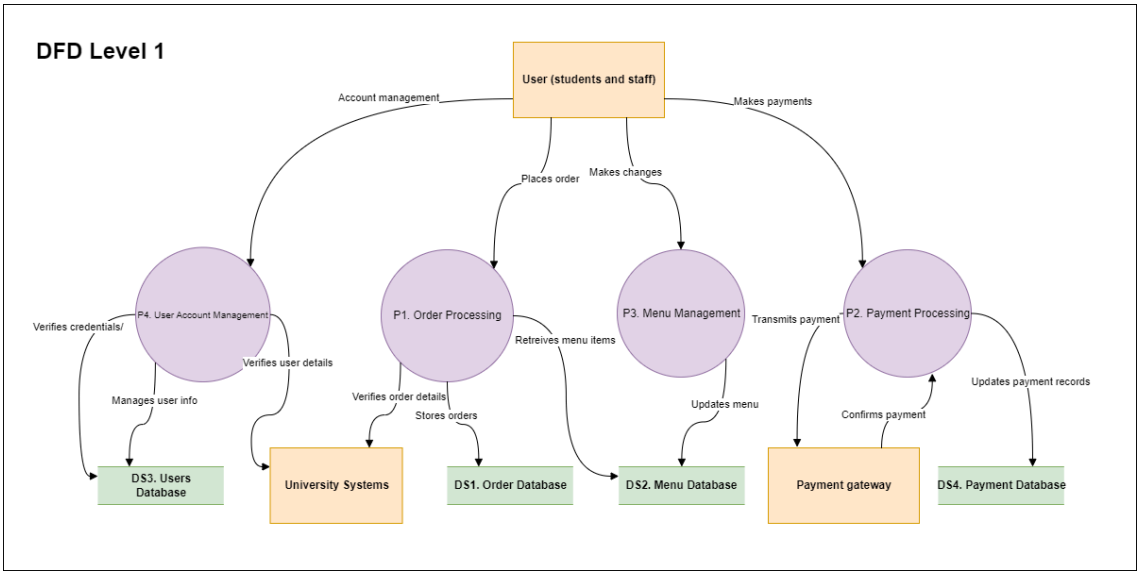

student2
studentId = "S2002"
username = "emily_chen"
email = "emily.chen@university.com"
order2
orderId = 5002
studentId = "S2002"
orderItems = [orderItem2, orderItem3]
status = "Delivered"
feedback2
feedbackId = 2002
userId = "S2002"
comment = "Delicious meals, but delivery was slow."
rating = 3
orderItem2
itemId = 102
quantity = 1
orderItem3
itemId = 104
quantity = 1


student2
studentId = "S2002"
username = "emily_chen"
email = "emily.chen@university.com"
order2
orderId = 5002
studentId = "S2002"
orderItems = [orderItem2, orderItem3]
status = "Delivered"
feedback2
feedbackId = 2002
userId = "S2002"
comment = "Delicious meals, but delivery was slow."
rating = 3
orderItem2
itemId = 102
quantity = 1
orderItem3
itemId = 104
quantity = 1

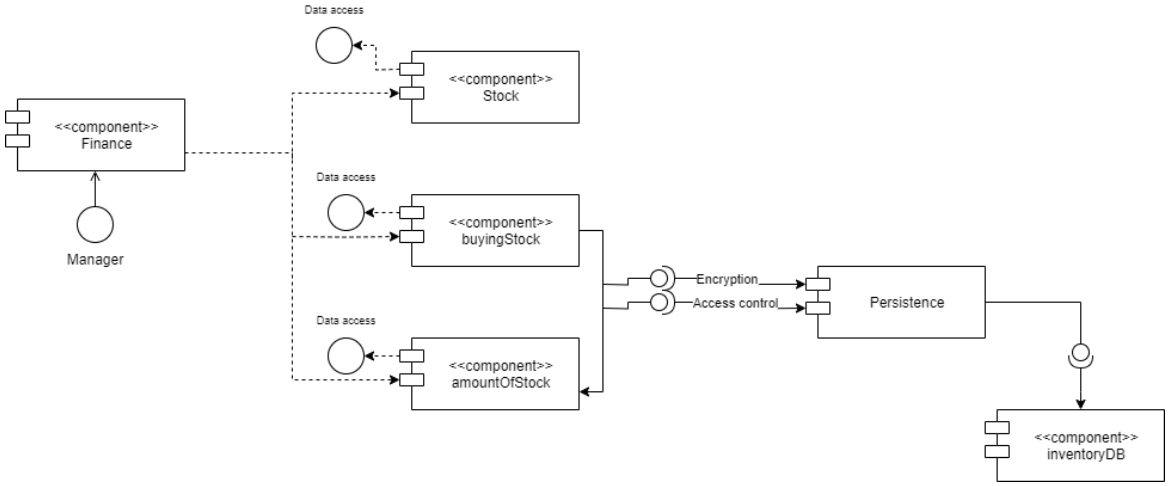

student1
studentid = "S1001"
username = "john_doe"
email = "john.doe@university.com"
order1
orderId = 5001
studentId = "S1001"
orderItems = [orderItem1]
status = "Placed"
feedback1
feedbackId = 2001
userId = "S1001"
comment = "Great food, quick service!"
orderItems = [orderItem1]
orderItem1
itemId = 101
quantity = 2
menu1
items = [menuItem1, menuItem2]
menuItem1
itemId = 101
name = "Veggie Sandwich"
price = 3.50
description = "A healthy vegeterian sandwich"
menuItem2
itemId = 102
name = "Chicken Salad"
price = 4.00
description = "Fresh chicken salad with greens"

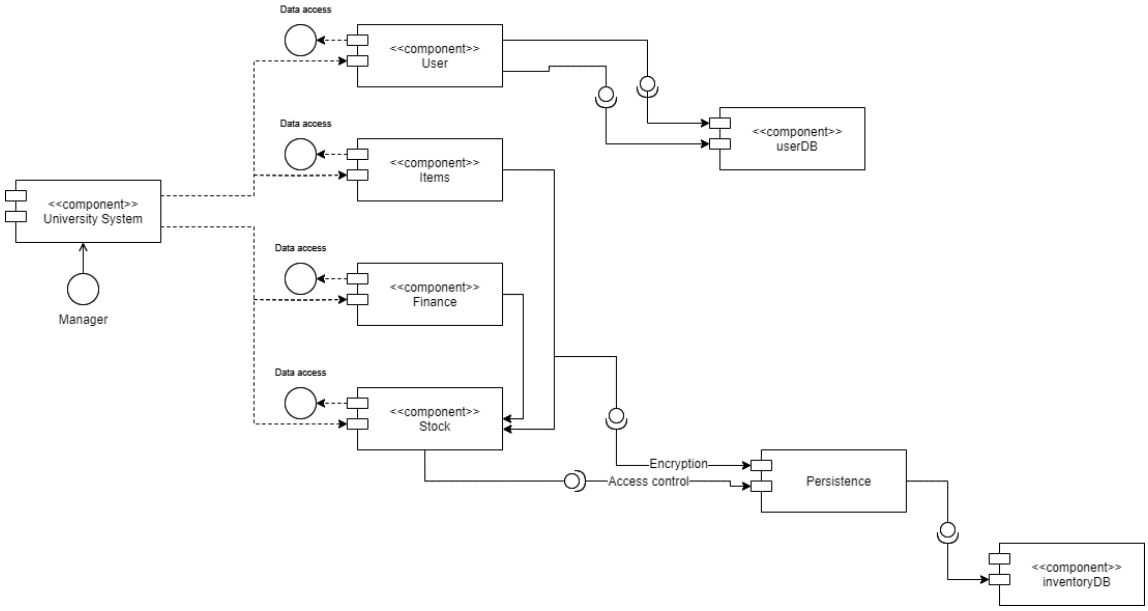
I. Data Flow Diagrams



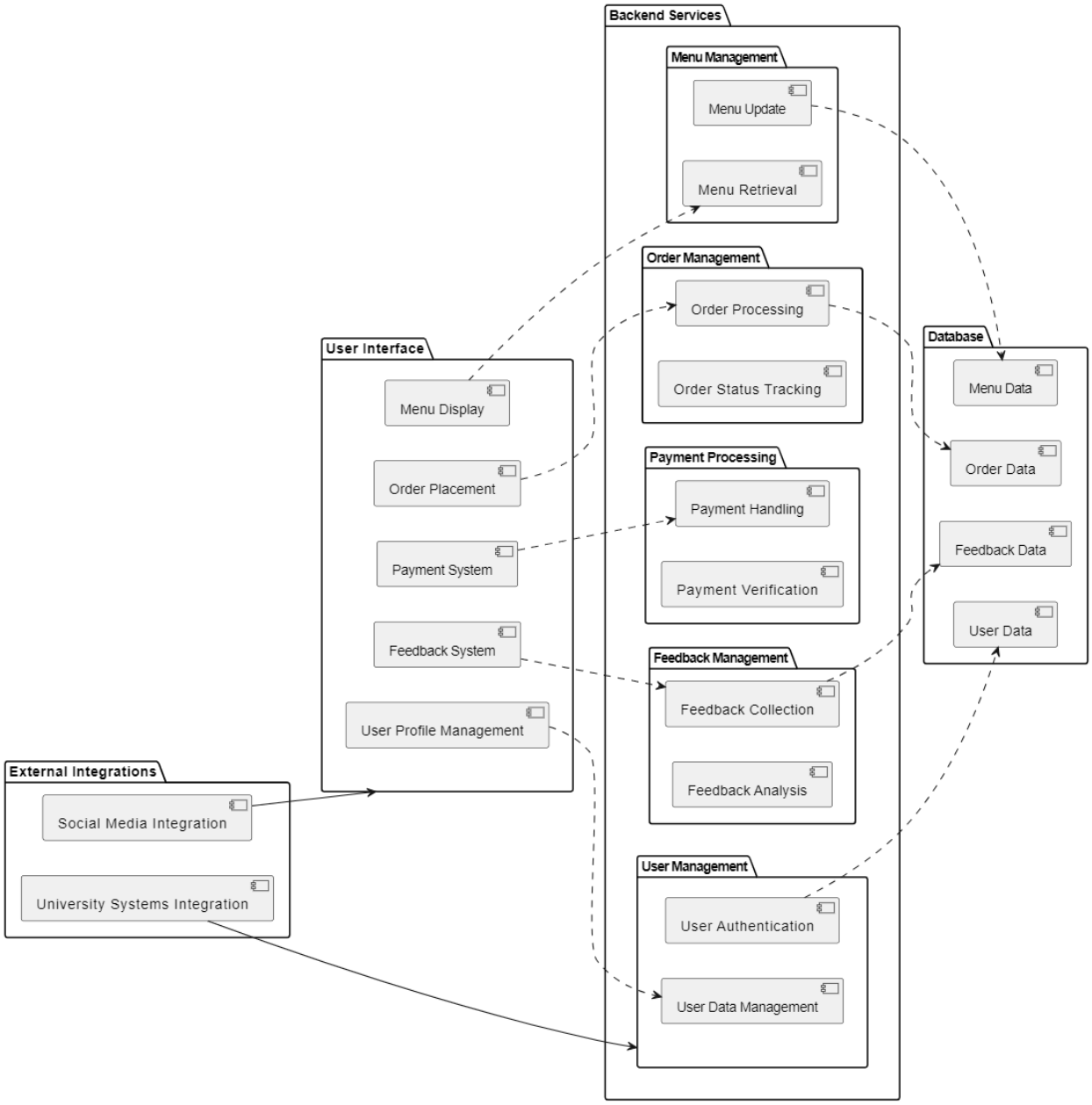


m. Component Diagram

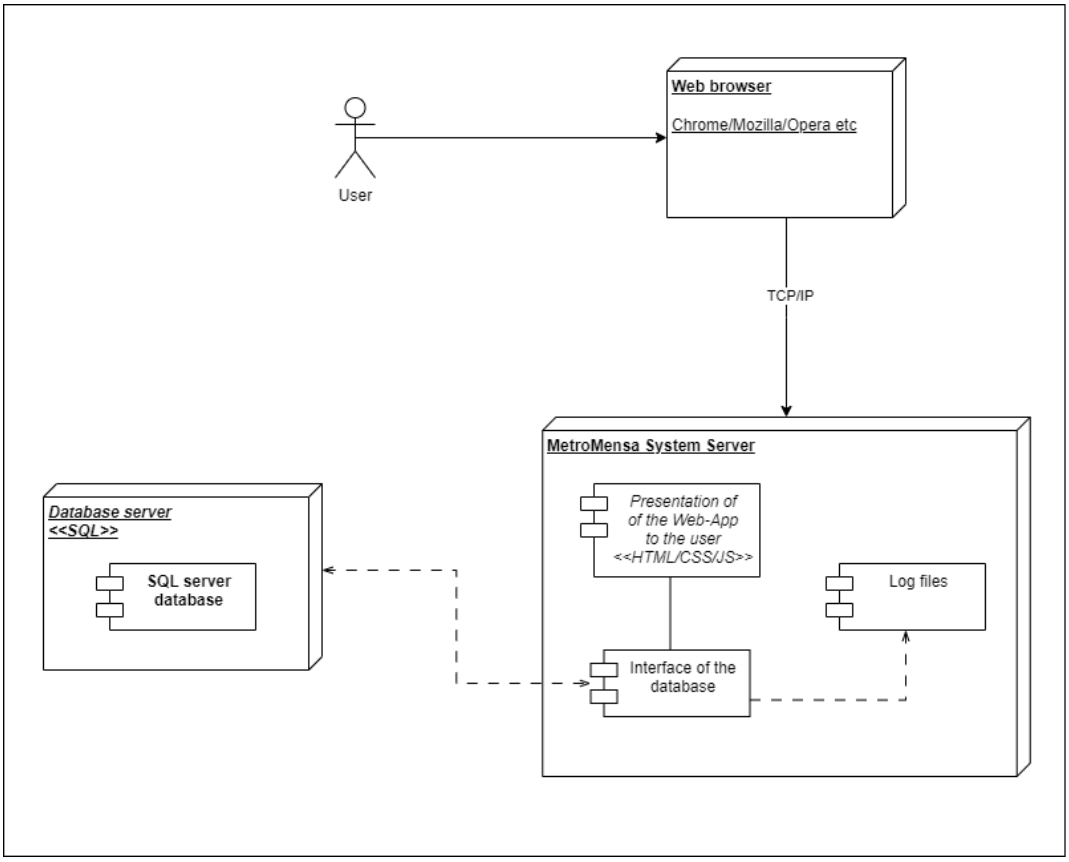


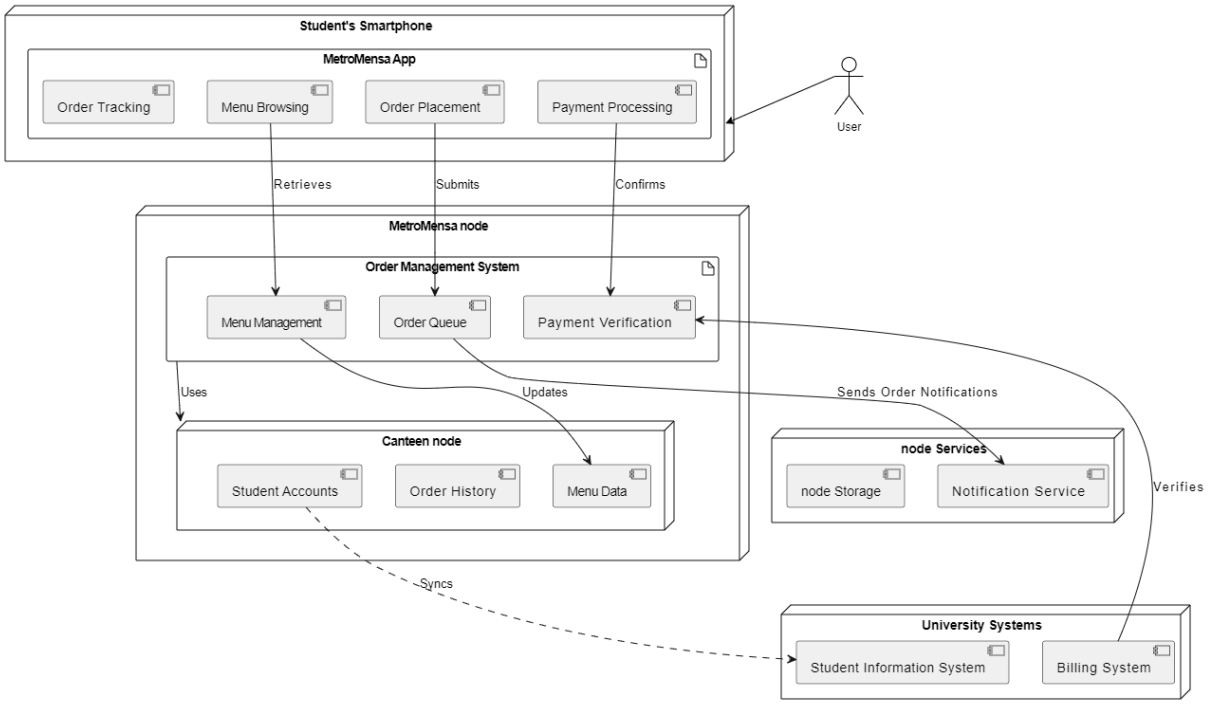


n. Package Diagram



o. Deployment Diagram





5. Design patterns.

a. Overview of DAO Pattern

The Data Access Object (DAO) pattern is a structural pattern that provides an abstract interface to a database or other persistence mechanisms. By mapping application calls to the persistence layer, DAOs provide some specific data operations without exposing details of the database. This separation of concerns makes the application code independent of the data access strategies.

Application in MetroMensa

In the context of MetroMensa, the DAO pattern can be particularly beneficial for handling all interactions with the database that stores information about menus, orders, users, feedback, and inventory. Here's how it can be integrated:

Structure

- MetroMensaDAO Interface: Defines the standard operations to be performed on the MetroMensa models (like Student, Order, MenuItem, etc.).

- StudentDAO, OrderDAO, MenuItemDAO, etc.: Concrete classes implementing the MetroMensaDAO interface, each responsible for handling database operations for their respective models.

Operations

- CRUD Operations: Each DAO class implements CRUD (Create, Read, Update, Delete) operations relevant to its model. For example, OrderDAO might have methods like addOrder(), getOrderById(), updateOrder(), and deleteOrder().
- Custom Queries: For complex queries, specific methods can be defined, such as getOrdersForStudent() in OrderDAO or getAvailableMenuItems() in MenuItemDAO.

Benefits

Decoupling Application and Persistence Logic: This separation allows changes in the database schema or switching to a different database provider without affecting the business logic.

Reusability and Maintainability: Common data access logic is centralized in DAO classes, promoting cleaner code and easier maintenance.

Testability: Business logic can be tested independently from the persistence layer.

Example Implementation

```java
public class StudentDAOImpl implements StudentDAO {
  public List<Order> getOrdersForStudent(String studentId) {
    // Code to query the database and return a list of orders for a given
student
  }
}

// Usage in Application
public class OrderService {
  private StudentDAO studentDAO = new StudentDAOImpl();

  public List<Order> getOrderHistoryForStudent(String studentId) {
    return studentDAO.getOrdersForStudent(studentId);
  }
}
```

B. Overview of Observer Pattern

The Observer Pattern is a behavioral design pattern that defines a one-to-many dependency between objects. When one object (the subject) changes its state, all its dependents (observers) are notified and updated automatically. This pattern is typically used for implementing distributed event handling systems and is crucial in scenarios where an object should be able to notify other objects without making assumptions about who those objects are.

Application in MetroMensa

In MetroMensa, the Observer Pattern can be effectively used to manage notifications related to order status updates, menu changes, or promotional events. This ensures that different parts of the system are updated in real-time as changes occur.

Structure

- Subject Interface: Defines the methods for attaching, detaching, and notifying observers.
- Concrete Subject (e.g., Order, Menu): Implements the subject interface and maintains the state. Notifies observers when changes occur.
- Observer Interface: Defines the method for receiving updates from the subject.

- Concrete Observer (e.g., User Interface, Email Notification System): Implements the observer interface. Each observer registers with a concrete subject to receive updates.

Operations

- Update Notifications: When the state of the subject (e.g., an Order) changes, it automatically notifies all attached observers, allowing them to update their state or behavior accordingly.
- Dynamic Subscription: Users or components of the system can subscribe or unsubscribe to the subject dynamically, allowing for flexible notification management.

Benefits

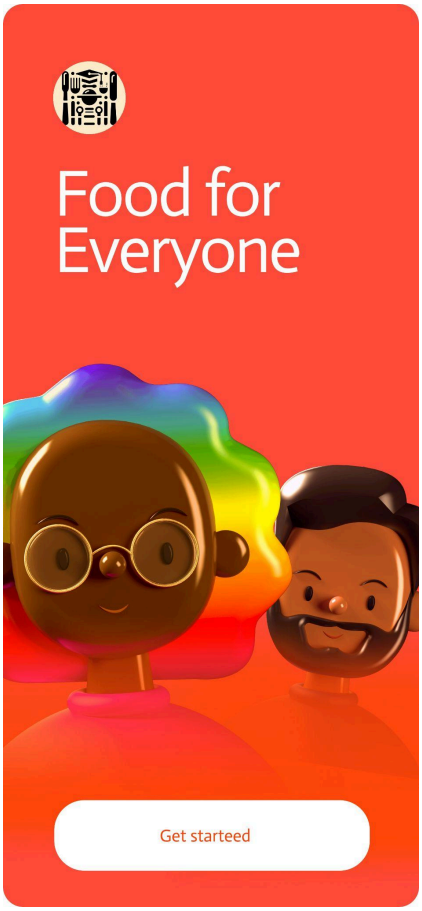
- Loose Coupling: The subject doesn't need to know anything about the observers, promoting loose coupling between the notifying system and the receiving system.
- Real-time Updates: Changes in the subject are propagated to all observers in real-time, ensuring the system remains consistent and up-to-date.
- Scalability: New observers can be added at any time without modifying the subject, making the system scalable.

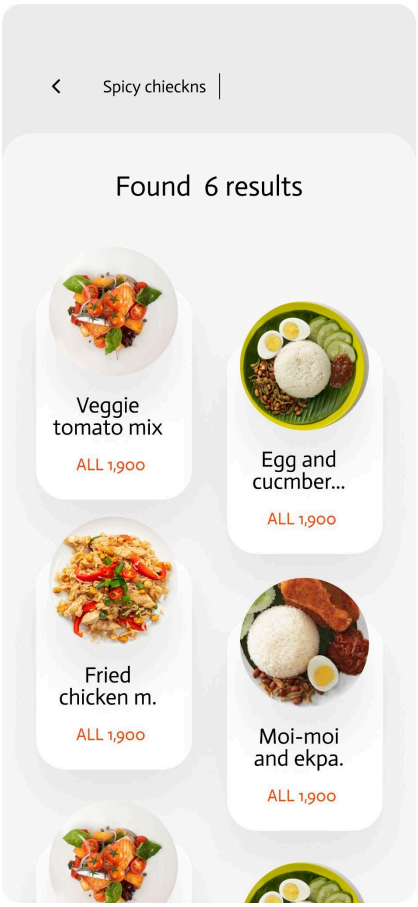
Example Implementation.

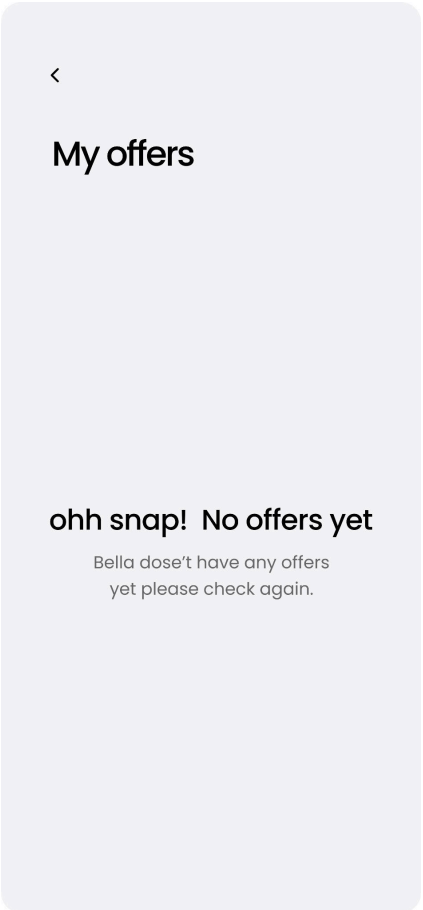
```java
public interface OrderObserver {
  void update(Order order);
}

public class Order implements Subject {
  private List<OrderObserver> observers = new ArrayList<>();
  private String status;

  public void setStatus(String status) {
    this.status = status;
    notifyObservers();
  }

  public void attach(OrderObserver observer) {
    observers.add(observer);
  }

  public void detach(OrderObserver observer) {
    observers.remove(observer);
  }

  public void notifyObservers() {
```

```
    for (OrderObserver observer : observers) {
      observer.update(this);
    }
  }
}

public class UserInterface implements OrderObserver {
  public void update(Order order) {
    // Code to update the user interface based on the order status
  }
}

// Usage
Order order = new Order();
UserInterface ui = new UserInterface();
order.attach(ui);

// When the order status changes
order.setStatus("Preparing");
```

6. User Interface.







Checkout

Payment

Payment method

Card

Please note

DELIVERY TO DOOR
ALL 100

IN PERSON PICKUP
FREE

Cancel | Proceed

Total 23,000

Proceed to payment

Veggie tomato mix

ALL 230

Delivery info

Delivered between monday aug and thursday 20 from 8pm to 91:32 pm

Return policy

All our foods are double checked before leaving our stores so by any case you found a broken food please contact our hotline immediately.

Add to cart

My profile

Information

Student Name
studentemail@umt.edu.al
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum.

Payment Method

Card
UMT Credit
Paypal

Updatet

No internet Connection

Your internet connection is currently not available please check or try again.

Try again

Spicy chieckns

Item not found

Try searching the item with a different keyword.

Login | Sign-up

Email address
studentemail@umt.edu.al

Password
****

Forgot password?