



ADDIS ABABA UNIVERSITY

FACULTY OF NATURAL AND COMPUTATIONAL
SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Enbla food delivery and recommendation system

COSC 4123: FINAL PROJECT II

<u>NAME OF THE STUDENT</u>	<u>ID NO.</u>
----------------------------	---------------

1. ABAYSEW TEKLE	UGR/8264/12
2. ABDU MOHAMMED	UGR/3032/12
3. ABEL HAILEMICHAEL	UGR/2301/12
4. ABIRHAM YOHANNES	UGR/9887/12

DECLARATION

This is to declare that this project work which is done under the supervision of Instructor Ashenafi and having the title Enbla food delivery and recommendation is the sole contribution of:

1. ABAYSEW TEKLE
2. ABDU MOHAMED
3. ABEL HAILEMICHAEL
4. ABIRHAM YOHANNES

No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this

declaration is proven.

Date: _____

Group Members:

Full Name	Signature
_____	_____
_____	_____
_____	_____
_____	_____

CERTIFICATE

I certify that this BSc final project report entitled Enbla food delivery and
recommendation system by:

- | | |
|----------------------|-------------|
| 1. ABAYSEW TEKLE | UGR/8264/12 |
| 2. ABDU MOHAMED | UGR/3032/12 |
| 3. ABEL HAILEMICHAEL | UGR/2301/12 |
| 4. ABIRHAM YOHANNES | UGR/9887/12 |

Is approved by me for submission. I certify further that, to the best of my knowledge, the
report represents work carried out by the students.

Date	Name and Signature of Supervisor

Acknowledgment

First of all, we would like to thank the Almighty 'God' who helped us in all aspects and gave us good health to do our works effectively and to complete this project documentation successfully. Secondly, we would like to express our deepest appreciation and gratitude to our advisor Mr. Ashenafi Kassahun for his advice on our project what we have to do timely and efficiently. We have very much appreciated his support, his numerous advices throughout the project, and his careful proofreading. And thirdly we would like to thank Computer Science academic staff for their guideline preparation which helps us as a framework to develop our project.

Acronym and Abbreviation

UI - User Interface

UX - User Experience

API - Application Programming Interface

REST API - REpresentational State Transfer Application Programming Interface

COCOMO - Constructive Cost Model

PM - Persons per Month

KLOC - Thousands of Lines of Code

EAF - Effort Adjustment Factor

E - Effort Applied

E_i - External Input

RAM - Random Access Memory

GB - GigaBytes

KBPS - Kilobytes per Seconds

GPS - Global Positioning System

SQL - Structured Query Language

NoSQL - Not Only Structured Query Language

OTP - One Time Password

SSL - Secure Sockets Layer

TLS - Transport Layer Security

EnblaFoods- Enbla Delivery and Recommendation System

CMS - Content Management System

Table of Contents

Table of Contents	1
1. CHAPTER ONE – PROJECT INTRODUCTION	5
1.1. Introduction	5
1.2 Statement of the Problem	6
1.3 Objective	6
1.3.1 General Objective	6
1.3.2 Specific Objective	6
1.4 Scope	7
1.5 Limitations	7
1.5 Methodology	8
1.5.1 System Development Methodology	8
1.5.2 System Development Tools	9
1.6 Significance of Enbla Food Delivery and Recommendation Application	9
1.7 Beneficiaries	10
1.8 Time Schedule	11
2. CHAPTER TWO - REQUIREMENT ANALYSIS	13
2.1 Introduction	13
2.2 Current System	13
2.2.1 Problems in Existing System	14
2.3. Requirement Gathering	14
2.3.1. Requirement Gathering Method	14
2.3.2. Results Found	15
2.4. Proposed System	15
2.4.1. Overview	15
2.4.2. Functional Requirements	16
2.4.3. Non – Functional Requirement	17
2.4.3.1. User Interface and Human Factors	17
2.4.3.2. Documentation	17
2.4.3.3. Hardware Consideration	18
2.4.3.4. Performance Characteristics	18
2.4.3.5. Error Handling and Extreme Conditions	18
2.4.3.6. Quality Issues	19
2.4.3.7. System Modification	19
2.4.3.8. Physical Environment	19
2.4.3.9. Security Issues	19
2.4.3.10. Resource Management	20
2.4.3.11. Constraints or Pseudo Requirements	20
2.5. System Model	20
2.5.1. Use case Model	20

2.5.1.1. Use Case Diagram	20
2.5.1.2 Description of Selected Use Case Model	22
2.5.2 Scenario	36
2.5.3 Object Model	41
2.5.3.1 Data Dictionary	41
2.5.3.2. Class Diagram	46
2.5.4. Dynamic Modelling	47
2.5.4.1. Sequence Diagram	47
2.5.4.2 Activity Diagram	57
2.5.4.3 State Machine Diagram	60
3. CHAPTER THREE - SYSTEM DESIGN	65
3.1 Introduction	65
3.2 Design Goals	65
3.3 Proposed software architecture	67
3.3.1 Overview	67
3.3.2 Subsystem Decomposition	68
3.3.3 Hardware/software mapping	76
3.9 Persistent Data Management	77
3.3.5 Access Control and Security	82
3.3.6 Subsystem services	88
4. CHAPTER FOUR: OBJECT DESIGN DOCUMENT	90
4.1 Detailed Class Diagram	90
4.1.1 Transaction Class Diagram	90
4.1.2 Food Recommendation Class Diagram	91
4.1.3 Manage Menu Item Class Diagram	92
4.1.4 Admin Class Diagram	93
4.1.5 Restaurant Admin class Diagram	95
4.1.6 Customer Class Diagram	96
4.1.7 Account Class Diagram	97
4.1.8 Orders Class Diagram	98
4.1.9 Payment Class Diagram	99
4.1.10 Notification Service Class Diagram	100
4.1.11 Restaurant Class Diagram	101
4.1.12 Delivery Service Class Diagram	102
4.2 Packages	103
5. CHAPTER FOUR - IMPLEMENTATION AND TESTING	106
5.1 INTRODUCTION	106
5.2 Mapping Models to code	106
5.3.1 Mapping Association	106
5.2 Source code of Major Classes	112
5.3 SCREENSHOTS	117

5.3.1 Login -Client Side	117
5.3.2 Sign up Page	118
5.3.3 Admin Panel	118
5.3.4 Home Screen	118
5.3.5 Order screen -Admin side	118
5.3.6 Order screen - Client side	119
5.3.7 Cart screen - Client side	119
5.3.8 Dish screen - Admin side	119
5.3.9 Dish screen - Client side	120
5.3.10 Account screen - Admin side	120
5.3.11 Food Category - Admin side	120
5.3.12 Restaurant List - Admin side	120
5.4 TESTING	120
References	129

List of Figures

Figure 1- 1 Time Schedule	7
Figure 2- 1 Use Case Diagram	14
Figure 2- 2 Class Diagram	36
Figure 2- 3 Cancel Order Sequence Diagram.....	48
Figure 2- 4 Update Order Sequence Diagram	48
Figure 2- 5 Manage Account Sequence Diagram	49
Figure 2- 6 Add Restaurant Sequence Diagram	49
Figure 2- 7 Update delivery status sequence diagram.....	50
Figure 2- 8 Choose Restaurant sequence Diagram.....	50
Figure 2- 9 View Favorite List.....	51
Figure 2- 10 View Order History Sequence Diagram	52
Figure 2- 11 Delete restaurant Sequence Diagram.....	53
Figure 2- 12 Add Menu Item Sequence Diagram	54
Figure 2- 13 Delete Menu Item Sequence Diagram	55
Figure 2- 14 Track Delivery Sequence Diagram	55
Figure 2- 15 View Transaction Sequence Diagram.....	56
Figure 2- 16 Activity Diagram For Delete Restaurant	57
Figure 2- 17 Activity Diagram for Deleting Order	58
Figure 2- 18 Activity Diagram for Food Recommendation.....	59
Figure 2- 19 State Chart Diagram For Order	60

Figure 2- 20 Landing Page for EnblaFoods	61
Figure 2- 21 Login Interface for EnblaFoods	62
Figure 2- 22 Choose menu For EnblaFoods.....	63
Figure 2- 23 Order Details For Enbla Foods	64
Figure 3- 1 System Decomposition	69
Figure 3- 2 Food Recommendation Subsystem.....	70
Figure 3- 3 Order Management subsystem	71
Figure 3- 4 User Management subsystem	72
Figure 3- 5 Inventory Management subsystem.....	73
Figure 3- 6 Notification Service	73
Figure 3- 7 Payment Service	74
Figure 3- 8 Delivery Service	75
Figure 3- 9 Hardware/software mapping	79
Figure 4.1 Transaction Class Diagram	90
Figure 4.2 Food Recommendation Class Diagram.....	91
Figure 4.3 Manage Menu Item Class Diagram	92
Figure 4.4 Admin Class Diagram	94
Figure 4.5 Restaurant Admin class Diagram.....	95
Figure 4.6 Customer Class Diagram	96
Figure 4.7 Account Class Diagram.....	97
Figure 4.8 Order Class Diagram	98
Figure 4.9 Payment Class Diagram	99
Figure 4.10 Notification Service Class Diagram	100
Figure 4.11 Restaurant Class Diagram	101
Figure 4.12 Delivery Service Class Diagram	102
Figure 4.13 Package diagram	104

Figure 4.14 Packages dependency diagram	105
---	-----

1. CHAPTER ONE – PROJECT INTRODUCTION

1.1. Introduction

Online Food delivery system is a system which helps restaurants to optimize and control over their restaurants. For the waiters it's making life easier because they don't have to go to the kitchen and give the order to the chef easily. And it makes the manager control the restaurant by having the records of the employee and orders.

And surprisingly the automated nature of the order-taking procedure makes the system significantly reduce the workload for the restaurant and consumers.

Back in 2010 or so in Ethiopia, the food delivery system was in its infancy. But at the end of the year 2015, Addis Ababa-based companies were introduced to deliver take-away restaurant food, groceries, beverages, flowers and books like Feleg and brought changes to the food delivery system.

In today's age of fast food and take out, many restaurants as well as customers have chosen to focus on quick preparation and speedy delivery of orders rather than offering a rich dining experience.

Existing System

Even Though there are a few food delivery service providers in Ethiopia still now delivery orders were placed in person or over phone mostly, and only delivered to cities or near cities.

In the existing system the food delivery services process orders, communicate with the restaurant and place customers' orders. The customer will visit their page or application, register with their credentials then go through a list of restaurants and foods. Once getting the preferred one then customers place an order and pay with a limited number of payment options.

The order placed then the food delivery service contact with the restaurant and the order will be delivered.

1.2 Statement of the Problem

We have Identified the following problems based on the current food delivery system in Ethiopia

- The need for more Food delivery service providers in sub cities and branch out to countryside
- The need for restaurant and foods recommendation incase customers doesn't know where and what to order
- The need for including low income restaurants like “mother bet” and be part of the system
- Inconvenience of the customer needing to have tangible copy of the menu
- Lack of visual confirmation that the order was placed correctly because humans are up to receive order
- Pain of customer waiting for the food and wasting valuable time
- The necessity for the restaurant to have an employee answering the phone and taking orders
- The need for delivering and promoting Ethiopian food overseas to many countries

1.3 Objective

1.3.1 General Objective

The general objective of the project is focused on providing a simple, time saving and smooth food ordering system for both consumers and restaurants in Ethiopia and promoting our food overseas.

1.3.2 Specific Objective

In order to achieve our end goal of developing this full fledged application we have the following specific objectives

- To analyze, wireframe, design, develop and test a mobile based application
- Developing an app that will surely satisfy the customer service.

- To design a system able to accommodate huge amounts of orders at a time.
- To improve the communication between the client and the server and minimize the time of ordering.
- To automatically compute the bill.
- To Pay from the comfort of their home using their smartphone or laptop.

1.4 Scope

Enbla is a mobile app that will connect a vast variety of food establishments with customers. The application will offer food recommendations, allow customers to place orders and offer multiple payment options. The application will also partner with local delivery services to provide reliable and efficient delivery services to customers. The scope of the solution covers all aspects of the food delivery experience, from browsing and ordering to delivery and payment, making it a one-stop-shop for customers' food delivery needs.

1.5 Limitations

- May cause compromise with food quality in some cases.
- It is only for Addis Ababa
- Food can get cold due to irresponsible delivery or traffic crowds.
- Restaurants can't design the menu UI to fit their brand or taste
- Customers may face late or incorrect orders
- It will not have real-time delivery tracking system on our first version
- Restaurants have less control over customer's experience with delivery, for instance if the delivery arrives cold or poorly presented after the trip it could negatively impact a customer's opinion of the restaurant as they have fewer opportunities to turn a bad experience around it will reflect badly on its rating and reviews.
- More room for error since there will be more points of contact between the order being prepared and the food ending up in the hands of the customer.

1.5 Methodology

1.5.1 System Development Methodology

The development of the food delivery mobile app will follow the Agile methodology, which emphasizes collaboration, flexibility, and customer satisfaction. The methodology will include the following steps:

- **Requirements Gathering:** The development team will work closely with stakeholders to gather and analyze customer requirements, restaurant requirements, and delivery requirements.
- **Design and Planning:** Based on the requirements, the team will design the user interface, database, and system architecture. The team will also create a detailed project plan that outlines the tasks, timelines, and resources required for the project.
- **Development:** The development team will start building the app, using agile techniques such as sprints, continuous integration, and continuous deployment. The team will also conduct regular code reviews and testing to ensure the app meets the requirements and is of high quality.
- **Testing and Deployment:** The app will be thoroughly tested for functionality, performance, and security before deployment. The team will also conduct user acceptance testing with a group of customers to ensure the app meets their expectations.
- **Maintenance and Support:** After deployment, the team will provide ongoing maintenance and support to ensure the application remains reliable and secure. The team will also regularly update the app to incorporate customer feedback and new technologies.

The Agile methodology will ensure that the development of the food delivery mobile app is customer-centric and iterative, allowing for frequent feedback and adjustments along the way.

1.5.2 System Development Tools

The following tools will be used for the development of the food delivery mobile app:

1. **Project Management:** Trello for project management, task tracking, and collaboration.
2. **Design:** Figma for UI/UX design, creating wireframes, and prototyping.
3. **Development:** React Native and Sanity CMS for cross-platform mobile app development
4. **Database:** Firebase, MongoDB for real-time data storage , retrieval and authentication purposes.
5. **API Development:** Node.js for API development, allows to seamlessly integrate the delivery service providers and restaurant partners.
6. **Testing:** Appium for unit testing and automated testing, and TestFlight or Google Play Beta for user testing and beta deployment.

These tools will provide the necessary infrastructure for the development of a high-quality and scalable food delivery app that offers efficient and reliable services to customers.

1.6 Significance of Enbla Food Delivery and Recommendation Application

Significance of our application :

- **Customer Convenience:** The Application provides customers with a quick and convenient solution for ordering food and tracking delivery. And its food recommendations feature also saves customers time and effort in choosing what to order.
- **Increased Sales for Restaurants:** By partnering with the application, restaurants will have access to a new customer base and an efficient platform for receiving and processing orders, leading to increased sales and revenue.
- **Partnership with Delivery Services:** The applications partnership with local delivery services provides a reliable and efficient delivery solution for customers, improving the overall delivery experience.

- **Improved Food Delivery Industry:** The application will improve the food delivery industry by providing a comprehensive solution that integrates recommendations, ordering, delivery, and payment into a single platform.
- **Technological Advancements:** The application will utilize cutting-edge technologies, such as machine learning and real-time data storage and retrieval, to provide customers with a personalized and efficient food delivery experience.

Our project is significant in terms of providing customers with a convenient and efficient solution for ordering food and improving the food delivery industry. The application's use of technology and partnerships with local delivery services will drive innovation and advancements in the food delivery space.

1.7 Beneficiaries

Enbla makes things easier for both restaurants and customers by facilitating the process on both sides. Beneficiaries of our system include:-

1. **Customers:** Customers will benefit from the convenience of ordering food and tracking delivery through the application, it also provides personalized food recommendations feature.
2. **Restaurants:** Restaurants will benefit from the increased sales and revenue generated through partnerships with the application and the efficient platform for receiving and processing orders.
3. **Delivery Services:** Delivery services will benefit from the partnership with the application, providing them with a steady stream of delivery requests and improving their delivery experience.
4. **Developers:** Developers will benefit from the opportunity to work on a cutting-edge project utilizing the latest technologies in the food delivery space.
5. **Investors:** Investors will benefit from the potential return on investment from the successful development and deployment of the Application.

Our project will provide benefits to a wide range of stakeholders, including customers, restaurants, delivery services, developers, and investors. By improving the food delivery experience for customers and generating increased sales for restaurants and delivery services, the application has the potential to drive innovation and growth in the food delivery industry.

1.8 Time Schedule

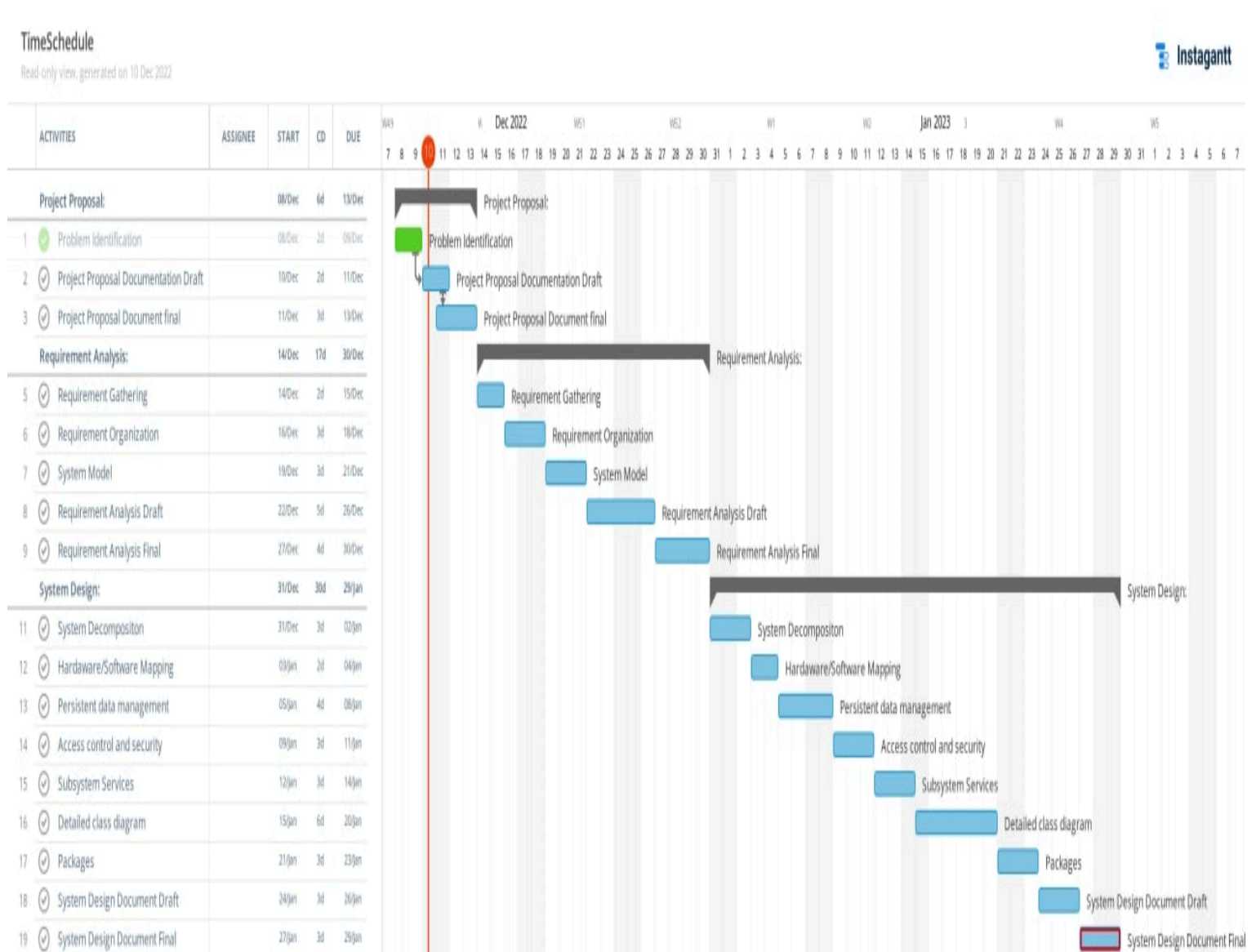


Figure 1-1 Time Schedule

Effort (cost) estimation

For cost estimation, we used the Constructive Cost model (COCOMO model). It is a well-documented and non-proprietary estimation model which is used to estimate effort needed for a project in terms of persons per month (PM).

The overall size of this system is estimated to be 4KLOC. This project falls under the organic project since it is a simple and small project which can be handled by a small team. In addition, it is relatively straightforward with few rigid requirements. Therefore, the initial effort estimated is:

$$E_i = 3.2 * (4)1.05 = 13.72 \text{ PM}$$

From the requirements, the ratings of the different cost driver attributes chosen for this project are:

- | | | |
|-----------------------------------|------|------|
| • Database size | High | 1.08 |
| • Programming language experience | Low | 1.07 |

From the above, the effort adjustment factor is:

$$EAF = 1.08 * 1.07 = 1.16$$

And the final effort will be:

$$E = E_i * EAF = 15.9 \text{ PM}$$

Therefore, this project needs 16 persons per month.

2. CHAPTER TWO - REQUIREMENT ANALYSIS

2.1 Introduction

Enbla foods, delivery and food recommendation System, is a food delivery, recommendation and advertisement system which directs to overcome problems in the existing system.

So we make a cross platform mobile application for customers and restaurants through which a customer can easily access all the things belonging to a restaurant while sitting anywhere instead of going to a restaurant and avoid facing rush and wasting a lot of time. Our system provides customers with the most recommended dishes based on the customer choice and the customer's preference. We enhance communication between customers and restaurants by providing restaurants a way to brief themselves about their services and our system also create a job opportunity for delivery organizations

The purpose of working on this project is to provide an ease to customers so that they can order anything, anytime while sitting anywhere and it can change the existing system into a better food delivery system.

2.2 Current System

The current food ordering system is a mix of both the traditional food ordering and food ordering by food delivery service providers. In traditional food ordering, customers will choose for the restaurant and call for delivery of a food if they have the contact of the restaurant. Or on the other hand customers need to go to the restaurants and order the food in person. After waiting till the food is fully served and the restaurant delivers the food on their location. In contrast food delivery service providers like 'tikus Delivery', 'beU Delivery' and so on, receive orders from customers via their smartphone and after the customer pays for the food, they contact the ride providers and serve the food based on the customers location.

2.2.1 Problems in Existing System

We have Identified the following problems based on the current food delivery system in Ethiopia

1. The need for more food delivery service providers in sub cities and branch out to countryside
2. The need for restaurant and foods recommendation incase customers does not know where and what to order
3. The need for including low income restaurants like “mother bet” and be connecting it to the the system
4. Inconvenience of the customer needing to have tangible copy of the menu
5. Lack of visual confirmation that the order was placed correctly because humans are up to receive order
6. Pain of customer waiting for the food and wasting their valuable time
7. The necessity for the restaurant to have an employee answering the phone and taking orders
8. The need for delivering and promoting Ethiopian food oversea to many countries

Our system shall overcome the problems by smoothing the food order process and letting them know which restaurant and foods are recommended for them. The system allows customers to easily get the locations and dishes of the restaurant and enhance communication by giving suggestions for them, so that they improve their services.

2.3. Requirement Gathering

2.3.1. Requirement Gathering Method

We have followed different methods to provide us with some insights into the system we are developing.

- We had gone through all the food delivery systems and their websites and had a look at their services.
- We had a discussion with “mother bet” owners about how the current system is handling them and how we should solve the gap and organize the food delivery system.
- We had an informal interview with Ms. Betelhem Eshetu worker at beU Delivery and gain info on how they work
- We had a discussion with random users in and out of the campus on the methods they used to order food. And gain a suggestion on how the developed system should be.

2.3.2. Results Found

After gathering information, we found out that

- Customers are still using traditional food ordering by phone call or in person
- Current food delivery systems does not have food recommendation system

- Currents food delivery systems does not have details page about the restaurant and the service they have
- The customers still want to have their food from their home but they don't have much knowledge about the current food delivery systems, so there need to be an advertisement

Through the requirement gathering we have identified basic features that the customers and restaurants look for.

2.4. Proposed System

2.4.1. Overview

The proposed system is a mobile-based application which solves shortcomings of the current working trends in the current food order and delivery system.

2.4.2. Functional Requirements

In this section we shall define the functional requirement of our proposed system. The functional requirement is describing the relationship between the system and its environment.

FR 1: The app shall provide personalized food recommendations based on the user's food preferences and order history.

FR 2: The system shall allow users to be able to view and modify their food preferences in their profile.

FR 3 : The app shall provide a detailed menu for each restaurant, including descriptions of the dishes and images.

FR 4: The system users shall be able to place an order directly from the menu.

FR 5: The system shall allow users to be able to view the restaurant's ratings, reviews, and delivery time estimates.

FR 6: The system shall allow users to track the status of their delivery in real-time.

FR 7: The system shall allow users to receive notifications when the delivery is on its way and when it arrives.

FR 8: The app shall provide a secure payment processing system for users to place their orders and make payments.

FR 9: The app shall let the user choose multiple payment methods, such as credit cards, debit cards, and in cash payment services.

FR 10: The system shall allow users to view their order and payment history in their profile.

FR 11: The app shall allow users to be able to view their current and past orders in the app.

FR 12: The app shall let users be able to cancel or modify their orders before they are processed.

FR 13: The app shall integrate with a third-party delivery service for food delivery.

FR 14: The app shall integrate with a payment gateway for secure payment processing.

FR 15: An administrative panel shall manage restaurants, menu items, and deliveries.

FR 16: The admin panel shall provide real-time monitoring of delivery status and payment processing.

2.4.3. Non – Functional Requirement

Non-functional requirements describe features of the system that are not directly related to the functional behavior of the system. Instead, non-functional requirements deal with the quality of the application, describe the system's operation capabilities and constraints and attempt to improve its functionality. Accordingly, the non-functional requirements of the system are listed below.

2.4.3.1. User Interface and Human Factors

The user interface of the system shall be interactive, user friendly, and easy to learn and not confusing users. Anyone, who has an understanding of how smartphones and internet browsers work together, can access the system.

2.4.3.2. Documentation

The whole development process starting from the proposal shall be well documented. While implementing the system, the source code shall also be well commented and variables and functions are descriptive of what it does. so that maintainers shall be able to understand easily and maintain the whole system.

2.4.3.3. Hardware Consideration

The admin side of the application should work on all devices, including mobile devices, tablets, laptops and desktops. Whereas The client-side services shall work on all devices including both android and IOS. The system should interact with a server, so the smartphone should have an android version more than 7.0, a Random-Access Memory (RAM) of at least 1GB, GPS support and should be able to support 100 Kbps or faster connection between client computer and the server.

2.4.3.4. Performance Characteristics

The system shall be responsive, allowing for proper display on mobile and tablet. The system shall accommodate at least 50 users concurrently. The system shall not take more than one minute to process one request and pay for the order .

2.4.3.5. Error Handling and Extreme Conditions

The system shall be prepared to handle errors and exceptions. Improper data input validation, destruction of the data during processing and sanitization shall be controlled by the system. If the user makes mistakes, the system shall give an error message or alert box that is used to correct errors produced. Since it is an online solution, it is prone to various exceptions so that any exceptions occurring shall be shown to the users.

2.4.3.6. Quality Issues

The system should be available 24 hours a day and 7 days a week unless Internet connection or electric power goes down. It is accessed by multiple users at a time and this shall make sure that the operation and errors of one user shall not affect the other or the performance of the entire system. It allows concurrent access for resources at times.

2.4.3.7. System Modification

In the future the system can be extended to have more functionalities. The system should have a tracking system to know where exactly their order is. But changes made should first be approved by the development team before the system is updated. However, any professional shall be able to understand and modify the system with the help of the documentation.

2.4.3.8. Physical Environment

The system shall be deployed on the tele servers which shall have optimal air conditioning and 24/7 server support

2.4.3.9. Security Issues

Security is a huge issue of the system. Since it deals with payment systems, users and restaurants data, the system should only be accessed by authorized people who have accounts and their password should be encrypted with enforcing strong password policy by performing validations both on client and server sides. We do payment validation both on the client and servers in order to prevent frauds and sanitize inputs attempting for SQL injection or other cyber attacks

2.4.3.10. Resource Management

The system needs a laptop or mobile phone or desktop or a tablet and web browser in order to work. And also, due to an online access behavior of the system we are focusing on maximizing the communication bandwidth utilization and minimizing response time.

2.4.3.11. Constraints or Pseudo Requirements

The main requirement to deploy an Our System is better internet infrastructure. However, in our country most of the time the internet connection does not work in some parts of the country and is sometimes fully blocked by the government for political issues. The other constraint is that a user has to know how to use smart phones and browsers to use the system.

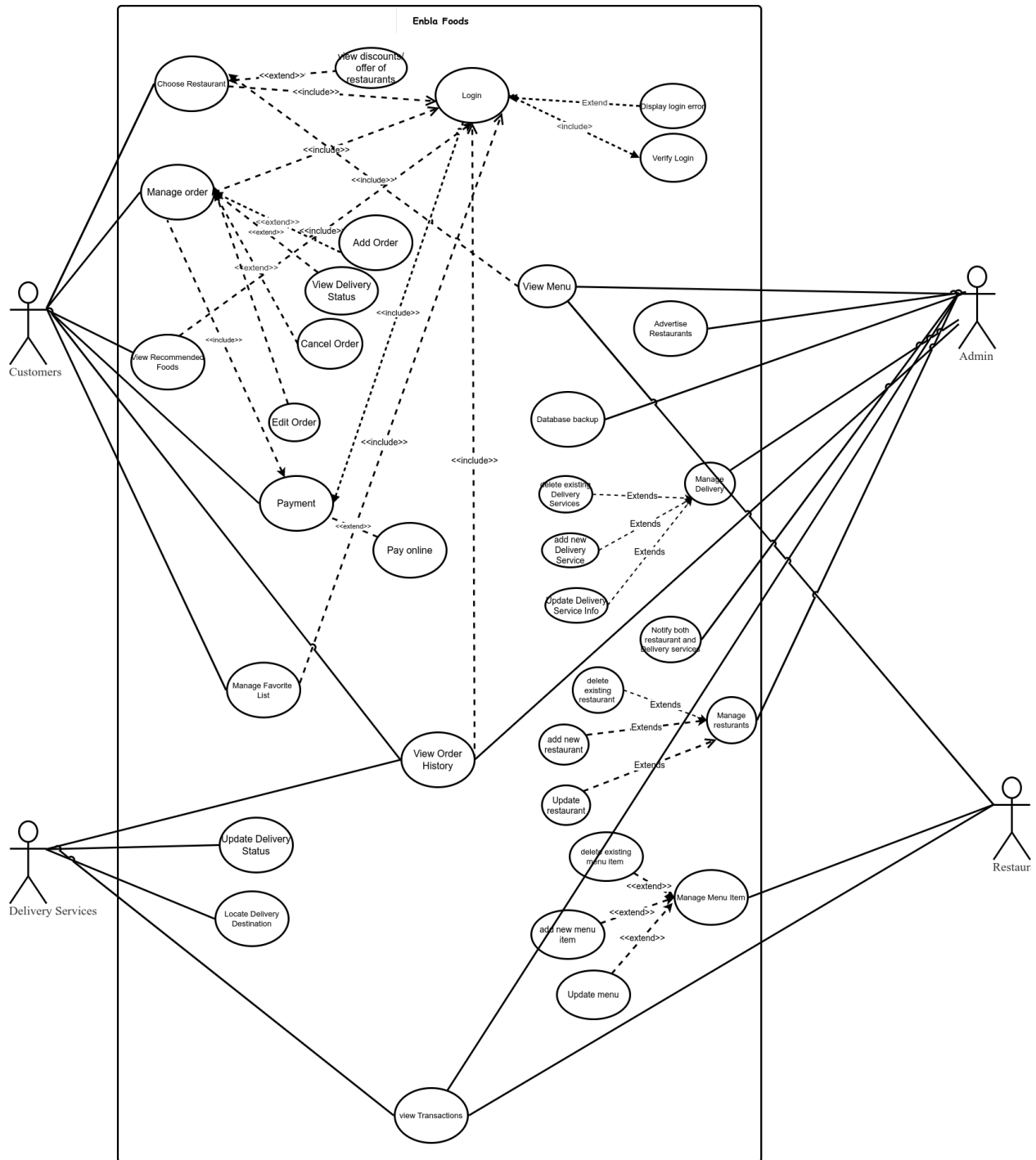
2.5. System Model

2.5.1. Use case Model

Use case is a technique for capturing the functional requirements of a system. Use cases work by describing the typical interaction between the users of a system and the system itself, providing a narrative of how a system is used.

2.5.1.1. Use Case Diagram

Figure 2.1 is used to illustrate how to capture the dynamic aspect of the system. It's representation of user's interaction with the system that shows the relationship



between the user and the different use cases in which the user is involved.

The use case model for our proposed system is described below:

Figure 2.1 use case diagram of EnblaFoods

2.5.1.2 Description of Selected Use Case Model

Description of Login Use Case

Use Case ID	001
Use Case Name	Login
Participating Actor	<ul style="list-style-type: none">• Customer• Delivery Service• Admin• Restaurant
Use Case Description	Allow users to access the system.
Flow Of Events	<ol style="list-style-type: none">1. The user navigates to the login page2. The app displays the login screen.3. The user fill the requited information and clicks login button [Alternate 1].4. The app opens where the user left off.
Alternate Flow	<ol style="list-style-type: none">3. [If the user fills the wrong username or password]<ol style="list-style-type: none">3.1. The page displays the error message dialogue and resets the login form to be empty.
Post condition	The user is logged into the system.

Table 2.1 Description of Login Use Case

Description of Manage User Account use case.

Use Case ID	002
Use Case Name	Manage User Account
Participating Actor	- Customer
Use Case Description	The customer can control their account
Pre-condition	The customer must be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. The user selects the settings icon on the top right corner of the display. 2. Then from the drop down menu select the Account option, this shall direct the customer to his account setting page. 3. If the customer wants to edit any information they can do so and select the save button. 4. If the customer wants to delete their account, there is a delete button at the button, which shall show a prompt if selected.
Alternate Flow	None
Post condition	Account information edited successfully.

Table 2.2 Description of Manage User Account use case.

Description of Choose Restaurant use case.

Use Case ID	003
Use Case Name	Choose Restaurant
Participating Actor	- Customer

Use Case Description	Lets the user choose which restaurant to order from.
Flow Of Events	<ol style="list-style-type: none"> 1. Customers select Choose Restaurants from the side menu. 2. The app displays a list of restaurants available. 3. The customer selects the restaurant of choice. 4. The menu of that restaurant shall be displayed with discounts offered by that restaurant displayed on top.
Alternate Flow	None
Post condition	The customers views the menu of the Restaurant that was selected

Table 2.3 Description of Choose Restaurant use case.

Description of Manage Order use case.

Use Case ID	004
Use Case Name	Manage Order
Participating Actor	- Customer

Use Case Description	Customers can update or cancel their food.
Pre-condition	The customer must be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Manage order option is selected from the side menu. 2. List of orders shall be displayed with estimated delivery time. [Alternate 2] 3. If the customer wants to cancel order they select the cancel option and that order shall be canceled 4. If the customer wants to update any order they can select the update option and they can remove items from the order or select add items options to be directed to restaurants list to add to their order.
Alternate Flow	<ol style="list-style-type: none"> 2. [If there were no orders made] <ol style="list-style-type: none"> 2.1. The page displays “No order has been placed.” message.
Post condition	The customers cancel or update their order.

Table 2.4 Description of Manage Order use case.

Description of view Order History

Use Case ID	005
Use Case Name	view Order History

Participating Actor	- Customer
Use Case Description	Displays all orders that were placed before.
Pre-condition	Customer should be logged in
Flow Of Events	<ol style="list-style-type: none"> 1. Customers select the Order History option from the side menu. 2. Customer is directed to a page where a list of orders made before is shown.[Alternate 2]
Alternate Flow	<ol style="list-style-type: none"> 2. [If no orders were made before] <ol style="list-style-type: none"> 2.1 The page displays a “ No order has been made yet” message.
Post condition	Customers view a list of orders they have made before.

Table 2.5 Description of view Order History

Description of View Delivery Status use case.

Use Case ID	006
Use Case Name	View Delivery Status
Participating Actor	- Customer
Use Case Description	Customers can track their orders during delivery.
Pre-condition	Customers should be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Customer selects the Track Delivery option. 2. Delivery Status is displayed

Alternate Flow	None
Post condition	The customers view the status of their order.

Table 2.6 Description of Track Delivery use case.

Description of Manage favorite list.

Use Case ID	007
Use Case Name	Manage Favorites List
Participating Actor	- Customer
Use Case Description	The customer can edit their favorites list.
Pre-condition	Customers must be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Customer selects the Favorites list from the side menu. 2. The customer is directed to a page that displays a list of dishes selected as favorites. [Alternate 2] 3. If the customer wants to remove an item, they can select the edit option and remove the item by selecting the remove button. 4. If the customer wants to change the order of their favorite list, they can select the edit option and drag the items in the order that suits them.
Alternate Flow	<ol style="list-style-type: none"> 2. [If the customer hasn't added any dishes to their favorites yet] <ol style="list-style-type: none"> 2.1. The page displays "No items have been added yet." message.

Post condition	Customer views list of favorites and can edit it.
----------------	---

Description of the Logout use case

Use Case ID	008
Use Case Name	Logout
Participating Actor	<ul style="list-style-type: none"> • Customer • Admin • Restaurant • Delivery Service
Use Case Description	Allows user to logout from the app.
Pre-condition	User should be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Select the logout option on the top right of the app. 2. User is logged out and directed to the homepage.
Alternate Flow	None.
Post condition	User is logged out

Table 2.7 Description of Manage favorite list.

Description of Update Delivery Status use case.

Use Case ID	009
Use Case Name	Update Delivery Status

Participating Actor	Delivery Service
Use Case Description	The delivery services update the status of the order.
Pre-condition	Delivery Service should be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Delivery Service enters updated status of order. 2. Delivery status is updated for the customer.
Alternate Flow	None
Post condition	Delivery service provides the delivery status of order when there is change.

Table 2.8 Description of Update Delivery Status use case.

Description of view recommended food use case.

Use Case ID	010
Use Case Name	view recommended food

Participating Actor	Customers
Use Case Description	The app presents personalized food recommendations based on their food preferences and order history.
Pre-condition	<ul style="list-style-type: none"> User has logged into the app
Flow Of Events	<ol style="list-style-type: none"> 1. User selects "Food Recommendations" option 2. System displays a list of recommended food items based on user preferences and order history 3. User selects a food item from the list 4. System displays details of the selected food item including ingredients, price, and estimated delivery time 5. User confirms the order 6. System displays a confirmation page with order details
Alternate Flow	none
Post condition	none

Table 2.9 Description of Locate Delivery Location use case.

Description of Manage Orders use case.

Use Case ID	011
Use Case Name	Manage Orders

Participating Actor	<ul style="list-style-type: none"> • Customer • Admin
Use Case Description	Orders list from customers are viewed and they can be canceled
Pre-condition	Customer or Admin should be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. Manage Orders option is selected from the side menu. 2. If the restaurant is the user he/she is directed to a page displaying a list of active orders. The newest orders shall be on the top and each order can be clicked to show the dish list, order number and notes left by the customer. Orders can be canceled by selecting the cancel button after clicking on the order. [Alternate 1] 3. If the Delivery Service is the user he/she is directed to a page displaying a list of active orders. The newest orders shall be on the top and each order can be clicked to show the order details, location of the restaurants and location of customer is shown. Orders can be canceled by selecting the cancel button after clicking on the order. [Alternate 2]

Alternate Flow	<p>2. [If there are no active orders]</p> <p>2.1. The page shall display “No active orders currently.” message.</p>
Post condition	The restaurant or the delivery service can view their order list with necessary details and cancel them if need be.

Table 2.10 Description of Manage Orders use case.

Description of View Transactions

Use Case ID	012
Use Case Name	View Transactions
Participating Actor	<ul style="list-style-type: none"> • Restaurant • Delivery Service
Use Case Description	The user can view their transactions that they have had so far.
Pre-condition	Restaurant or Delivery service should be logged in.
Flow Of Events	<ol style="list-style-type: none"> 1. The user selects the transactions option on the side menu 2. The user is directed to a page that displays all transactions that they had so far. [Alternate 2]

	3. If the user selects any transaction, details of the transaction shall pop up.
Alternate Flow	2. [If there were no transactions done yet.] 2.1. The page displays a “No transactions have been made yet.” message.
Post condition	The user views all their previous transactions.

Table 2.11 Description of View Transactions

Description of Manage Menu Item use case.

Use Case ID	013
Use Case Name	Manage Menu Items
Participating Actor	- Restaurant
Use Case Description	The restaurant can edit their menu by adding, deleting or updating the items in their list
Pre-condition	Restaurant should be logged in.

Flow Of Events	<ol style="list-style-type: none"> 1. User selects the menu option from the side menu in the app. 2. The app directs the user to a page that displays a list of items in their menu. [Alternate 2] 3. If the restaurant wants to add an item to their menu they can do so by selecting the add icon button and a pop up shall come allowing them to enter the details of the new dish to be added. 4. If the restaurant wants to delete an item from the menu, the user selects the edit option and then selects the delete option that belongs to the dish they want to remove. 5. If the restaurant wants to edit a dish, they first select the edit option then select the dish they want to update. A pop up shall appear with the details of the dish, the user can go ahead and change what is necessary. 6. After the user is done they select the save button and the new menu list is displayed. [Alternate 6]
----------------	---

Alternate Flow	<p>2. [If the restaurant hasn't added any item to their menu yet]</p> <p>2.1. The page displays a "No items have been added yet." message.</p> <p>6 [If the user tries to leave without saving]</p> <p>6.1. A warning that they haven't saved their changes shall be displayed.</p> <p>6.2. If the user selects leave anyway option all changes shall be reset to how they were before and the user can leave the page.</p> <p>6.3. If the user selects cancel they shall return where they were at.</p>
Post condition	The restaurant menu was successfully edited.

Table 2.12 Description of Manage Menu Item use case.

Description of Manage Restaurants use case

Use Case ID	014
Use Case Name	Manage Restaurants
Participating Actor	- Admin
Use Case Description	The admin can control restaurant accounts by deleting, adding and (updating) the restaurants.
Pre-condition	Admin should be logged in.

Flow Of Events	<ol style="list-style-type: none"> 1. Admin selects manage restaurants from the side menu 2. Admin is directed to a page with a list of restaurants operating actively in the app. [Alternate 2] 3. If the admin wants to add a new restaurant, he/she selects the pending button, then a restaurant deemed to be fit can be approved and admitted to the list of restaurants. 4. If the admin wants to remove a restaurant then he/she selects a particular restaurant, a pop up shall appear showing details about the restaurant. He/she then selects the delete button where a prompt shall appear confirming if the admin wants to delete the account, once the confirm button is selected the restaurant is deleted from the list and is removed from the restaurants list from the app. [Alternate 4]
Alternate Flow	<ol style="list-style-type: none"> 2. [If no restaurants have been added yet] <ol style="list-style-type: none"> 2.1. The page shall display a “No restaurants added yet.” message.

	4. [If the admin selects cancel] 4.1. The admin returned to the restaurant's detail pop up.
Post condition	The restaurants list is updated.

Table 2.13 Description of Manage Restaurants use case

2.5.2 Scenario

A scenario is simply a sequence of possible events of what people may do or experience on the app. In this section the following tools will be used for the development of the food delivery mobile app system is represented as a set of scenarios expressed as a flow of events and major functions of the system. The following selected scenarios define some real-life instances of the proposed system.

Login Scenario

Scenario Id	001
Scenario Name	Login
Participating Actor(s)	<ul style="list-style-type: none"> • Admin (Akram) • Customer (Rayan) • Restaurant (Romina) • Delivery Service (Lego)

Flow of Events	<ol style="list-style-type: none"> 1. The user selects the login button on the upper-right corner of their page. 2. The app directs the user to the login page. 3. User fills in their username and password and selects the login button. 4. The system checks if the username and password provided are both valid. 5. If the information checks out the user is redirected to their respective page. 6. Otherwise if there is any error the app denies the login attempt.
----------------	--

Table 2.14 Login Scenario

The following tools will be used for the development of the food delivery mobile app

Manage Favorites List Scenario

Scenario Id	002
Scenario Name	Manage Favorites List.
Participating Actor(s)	- Customer (Rayan)
Flow of Events	<ol style="list-style-type: none"> 1. Rayan selects the Favorites option from the side menu. 2. Rayan is directed to the favorites page. 3. To remove an item from their favorite list Rayan selects the edit option and delete buttons appear next to the items in their list 4. Rayan then selects the delete button and that item is removed from the list 5. Rayan then selects the save button so that their new list is updated.

Table 2.15 Manage Favorites List Scenario

Logout Scenario

Scenario ID	003
Scenario Name	Logout
Participating Actor(s)	<ul style="list-style-type: none"> • Customer (Rayan) • Admin (Akram) • Restaurant (Romina) • Delivery Service (Lego)

Flow of Events	<ol style="list-style-type: none"> 1. Once a user is ready to log out of the app, he/she selects the logout button on the upper-right corner of the page. 2. The app then redirects the user to the homepage of the app.
----------------	--

Table 2.16 Logout Scenario

Removing item from menu Scenario

Scenario Id	004
Scenario Name	Removing Item from Menu
Participating Actor(s)	- Restaurant (Romina)
Flow of Events	<ol style="list-style-type: none"> 1. Romina user selects the menu option from the side menu in the app, 2. The app directs the Romina user to a page that displays a list of items in their menu. 3. To delete an item from the menu, the Romina user selects the edit option and then selects the delete option that belongs to the dish they want to remove. 4. After the user is done they select save button and the new menu list is displayed.
Exit Condition	- Delete selected menu item successfully

Table 2.17 Removing item from menu Scenario

Removing a Restaurant

Scenario Id	005
Scenario Name	Removing a Restaurant
Participating Actor(s)	- Admin (Akram)
Flow of Events	<ol style="list-style-type: none"> 1. Akram selects manage restaurants from the side menu. 2. Akram is directed to a page with a list of restaurants operating actively in the app. 3. To remove a restaurant Akram selects the restaurant he wants, a pop up shall appear showing details about the restaurant. Akram then selects the delete button where a prompt shall appear confirming if the admin wants to delete the account, once the confirm button is selected the restaurant is deleted from the list and is removed from the restaurants list form the app,

Table 2.18 Removing a Restaurant

2.5.3 Object Model

2.5.3.1 Data Dictionary

Account

Field	Type	Login	Description	Example
userId	string	10	Unique Identification	Wowb01, Angla01
username	string	30	Login Name	Wow Burger
password	string	30	Login Password	W@wbr22!
role	string	30	Restrictions and permissions of users.	Customer User

Table 2.19 Account

View Transactions

Field	Type	Length	Description	Example
transactionId	string	30	Unique Identification	Tr011
transactionType	string	50	Type of transaction	Order payment
custId	string	10	Unique Identification of customer	cu575
deliveryServiceId	string	10	Unique Identification of Delivery Service	dd394
restaurantId	string	10	Unique Identification of Restaurant	rj394

dateOfTransaction	date	N/A	Time the transaction was made	2023/10/22 22:10:32
-------------------	------	-----	-------------------------------	------------------------

Table 2.20 View Transactions

Manage Orders

Field	Type	Length	Description	Example
orderId	string	30	Unique Identification	Od755
price	double	10	Price of the order	Order payment
OrderTime	datetime	N/A	Time the order was made	2023/02/24 00:48:12
itemNum	int	5	Number of items in the order	7
itemList	array	30	List of items in the order	Apple juice, firfir, burger
trackNum	string	20	Id of the delivery	AA2948
custId	string	10	Unique Identification of customer	CC3948
restaurantId	string	10	Unique Identification of restaurant	rr393
deliveryServiceId	string	10	Unique Identification of delivery service	ds93ud

deliveryStatus	string	20	Delivery status of the order	Delivered
----------------	--------	----	------------------------------	-----------

Table 2.21 Manage Orders

Customer

Field	Type	Length	Description	Example
custId	string	10	Unique Identification of customer	Cr394
cusFName	string	30	Customer's first name	Akram
cusLName	string	30	Customer's last name	Mohamed
username	string	10	Customer's username	akmoh7
password	string	30	Password of the	3894jgkH!

			custom er	
email	string	30	Email of the custom er	hewow @gmail. com

Table 2.22 Customer

Restaurant

Field	Type	Length	Description	Example
restaurantId	string	10	Unique Identification for restaurants	RE443
restaurantName	string	50	Restaurant's name	Angla Burger
username	string	10	Restaurant's username	angl235
password	string	30	Restaurant's password	@nglwj34
email	string	30	Restaurant's email	angla@gmail.com

Table 2.23 Restaurant

Delivery Services

Field	Type	Length	Description	Example
deliveryServiceId	string	10	Unique Identification for delivery services	go234
deliveryServiceName	string	50	Name of delivery service	legoo
username	string	10	Delivery Service's name	legoo272
password	string	30	Delivery Service's password	qsJ@Of
email	string	30	Delivery Service's email	legoo6@gmail.com

Table 2.24 Delivery Services

Food Item

Field	Type	Length	Description	Example
itemId	string	30	Unique Identification of food item	frfr394
itemName	string	50	Name of the food item	Firfir
foodType	string	30	Type of the food	Soup
restaurantId	string	10	Unique Identification for the restaurant	Re958
unitPrice	double	10	Price of a unit item	60.45

Table 2.25 Food Item

2.5.3.2. Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (methods) and the relationships among objects.

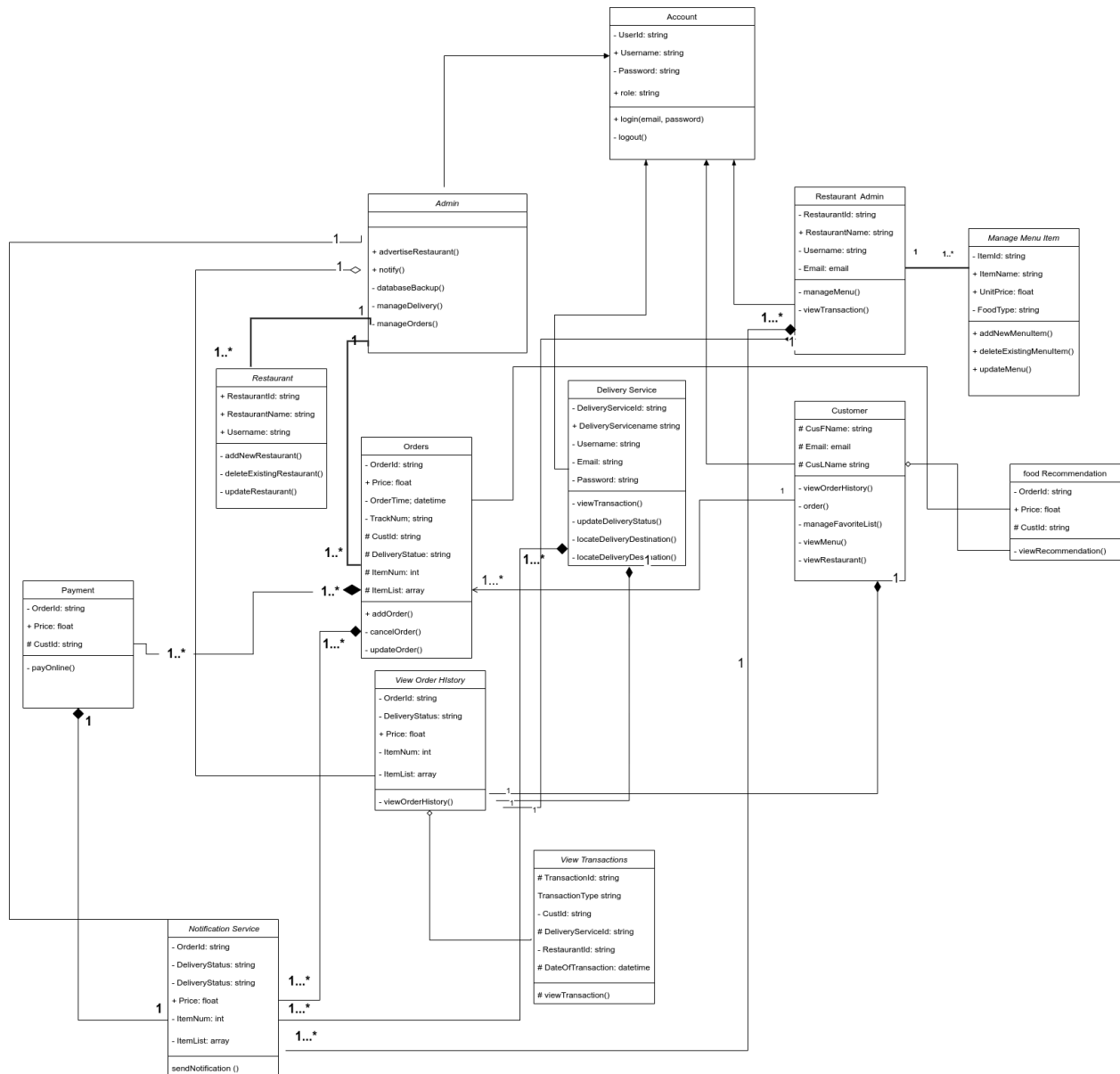


Fig 2.2 Class Diagram

2.5.4. Dynamic Modelling

2.5.4.1. Sequence Diagram

A sequence diagram is an interaction diagram that shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagram are typically associated with use case realizations .

Accordingly, some selected to show sequence diagrams of our proposed system are as follows.

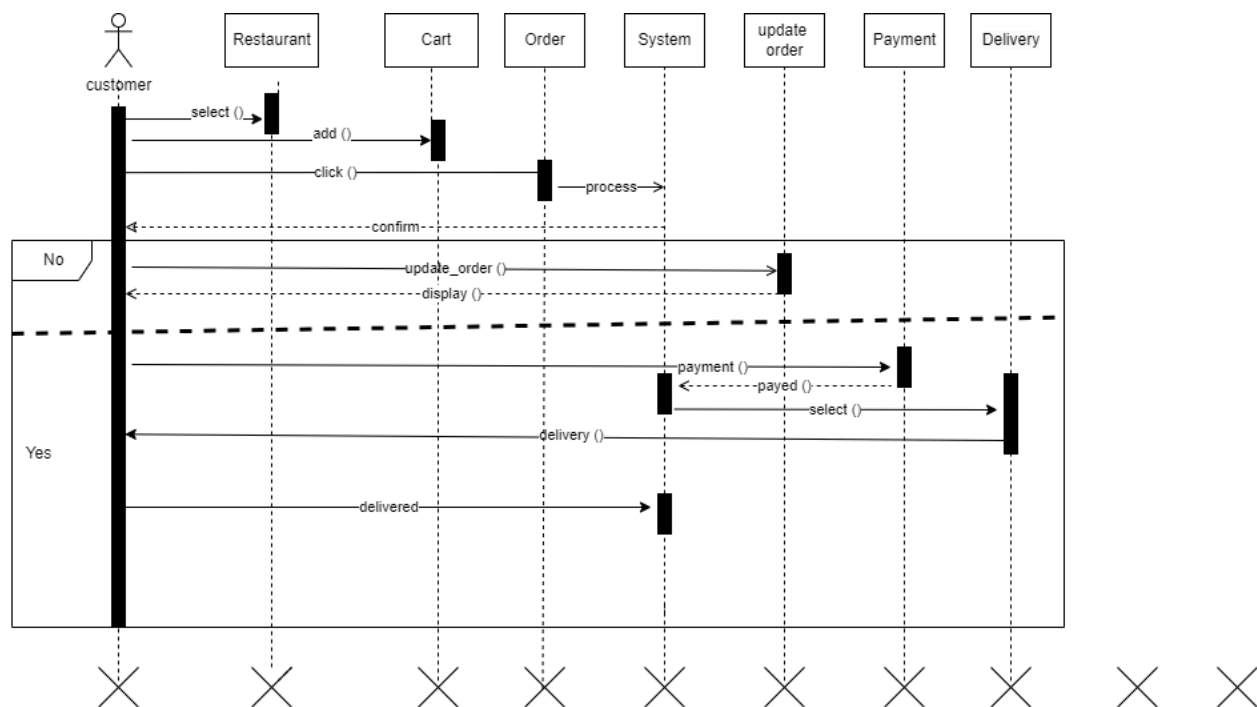
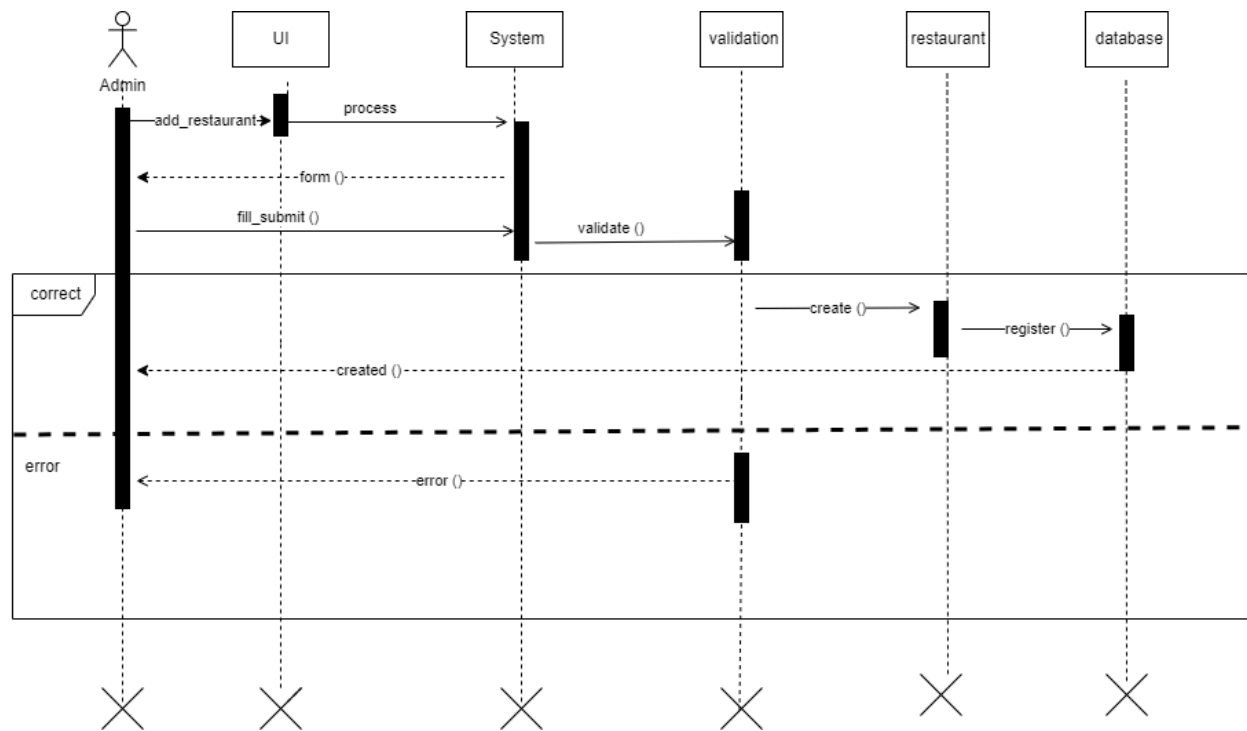


Fig 2.3 Manage Order Sequence Diagram



2.4 Add Restaurant Sequence Diagram

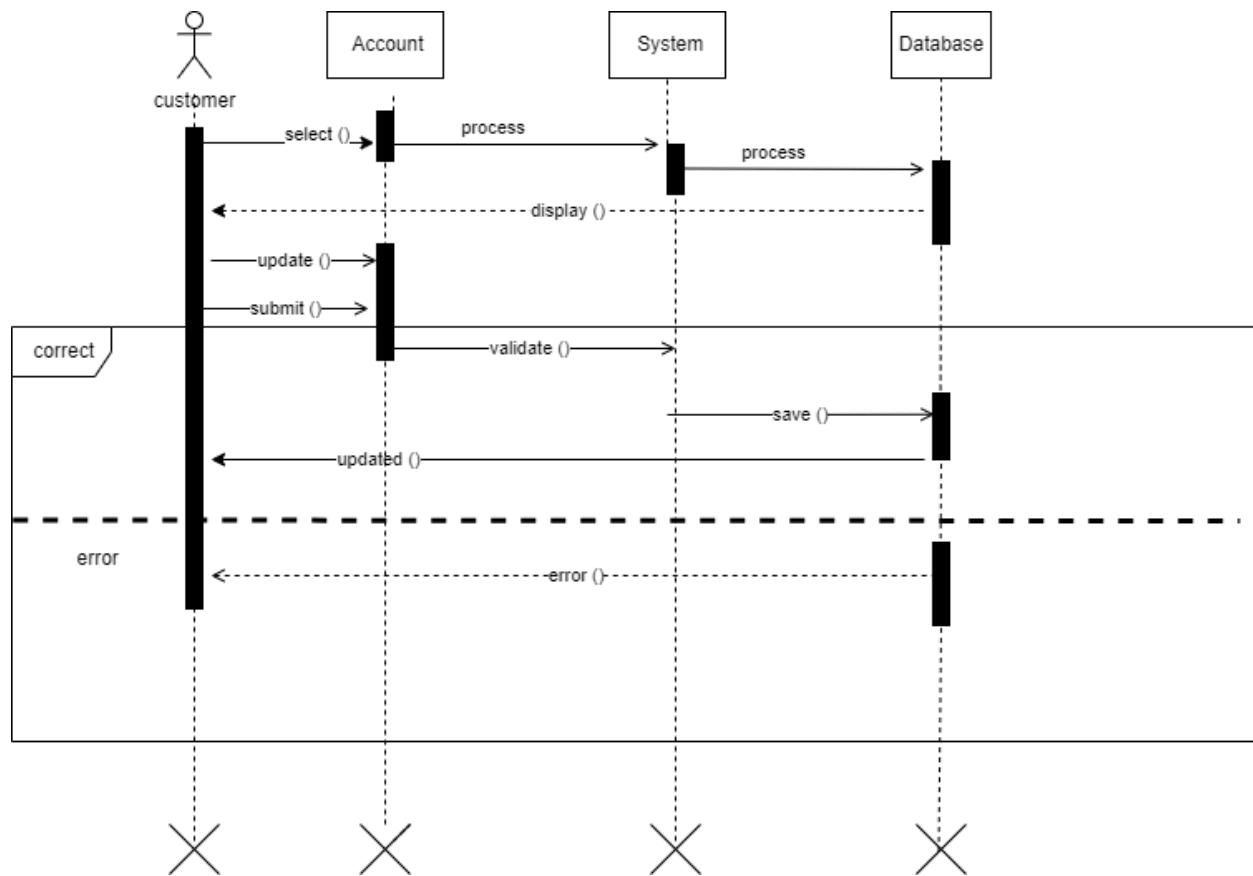


Fig 2.5 Manage Account Sequence Diagram

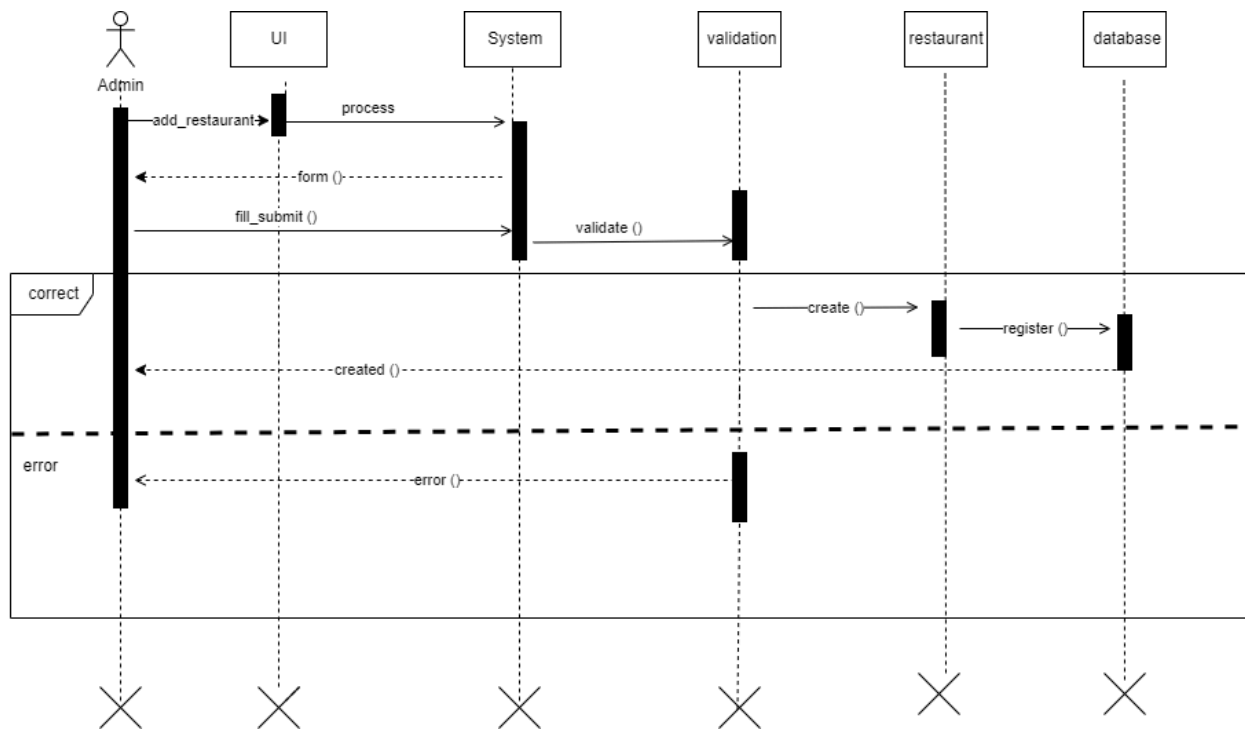


Fig 2.6 Add Restaurant Sequence Diagram

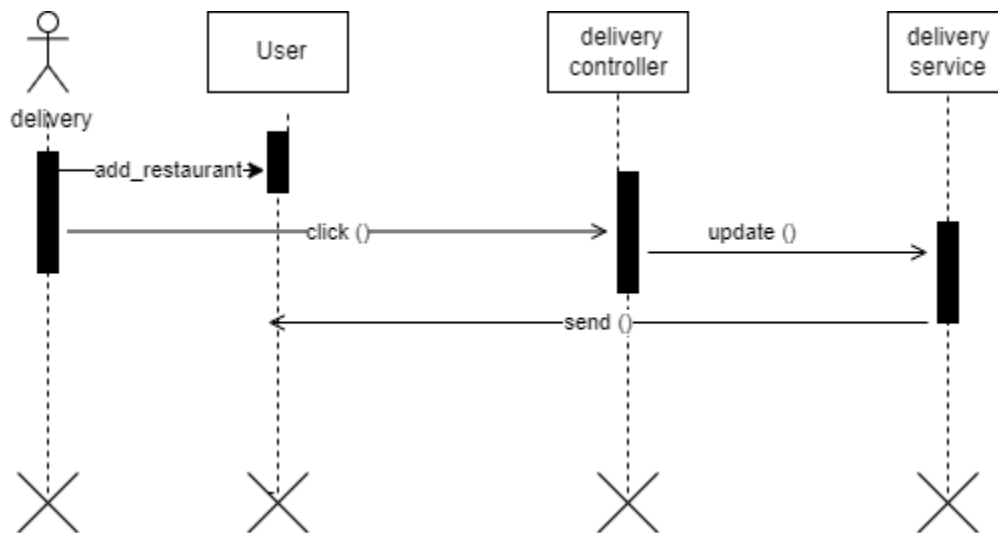


Fig 2.7 Update delivery status sequence diagram

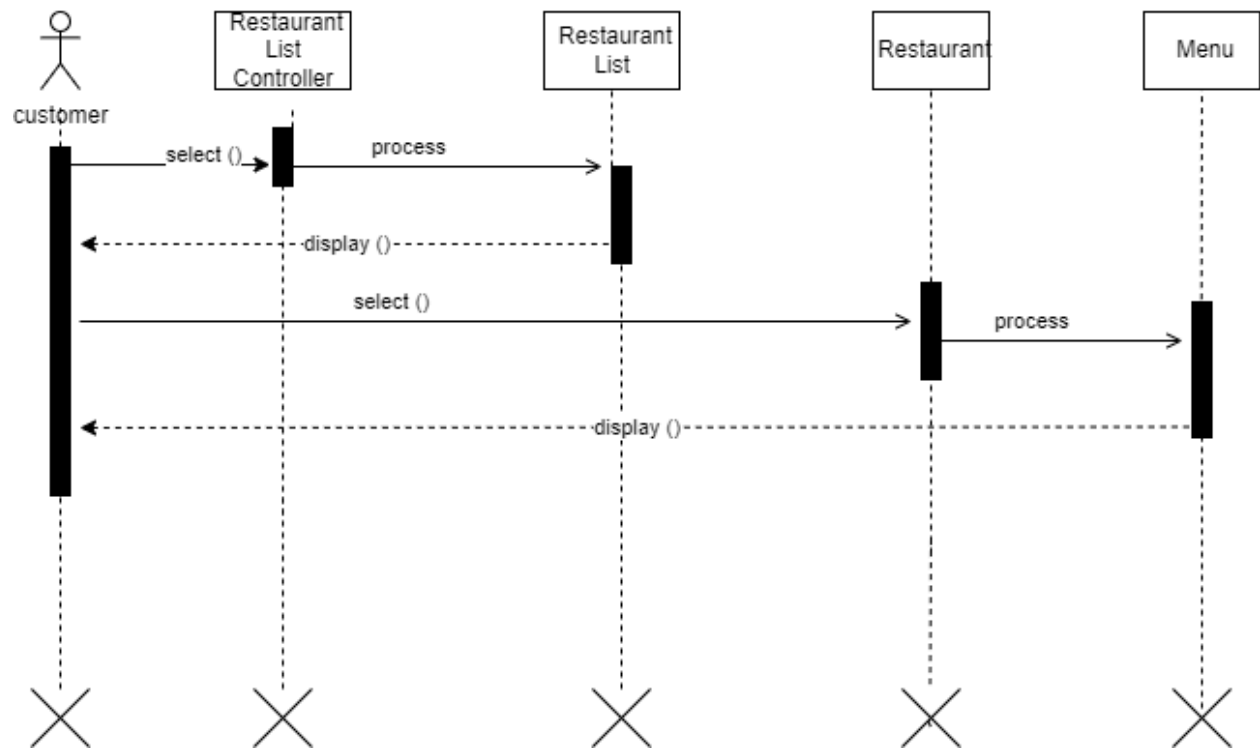


Fig 2.8 Choose Restaurant sequence Diagram

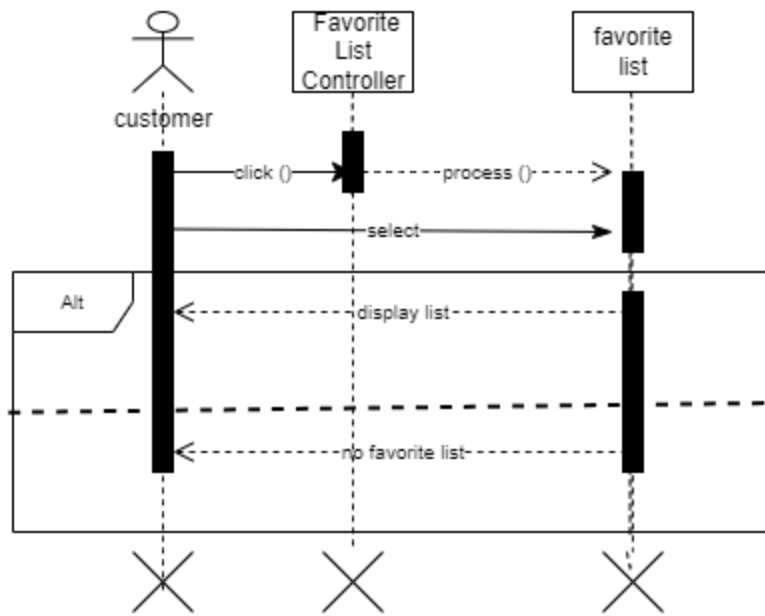


Fig 2.9 View Favorite List

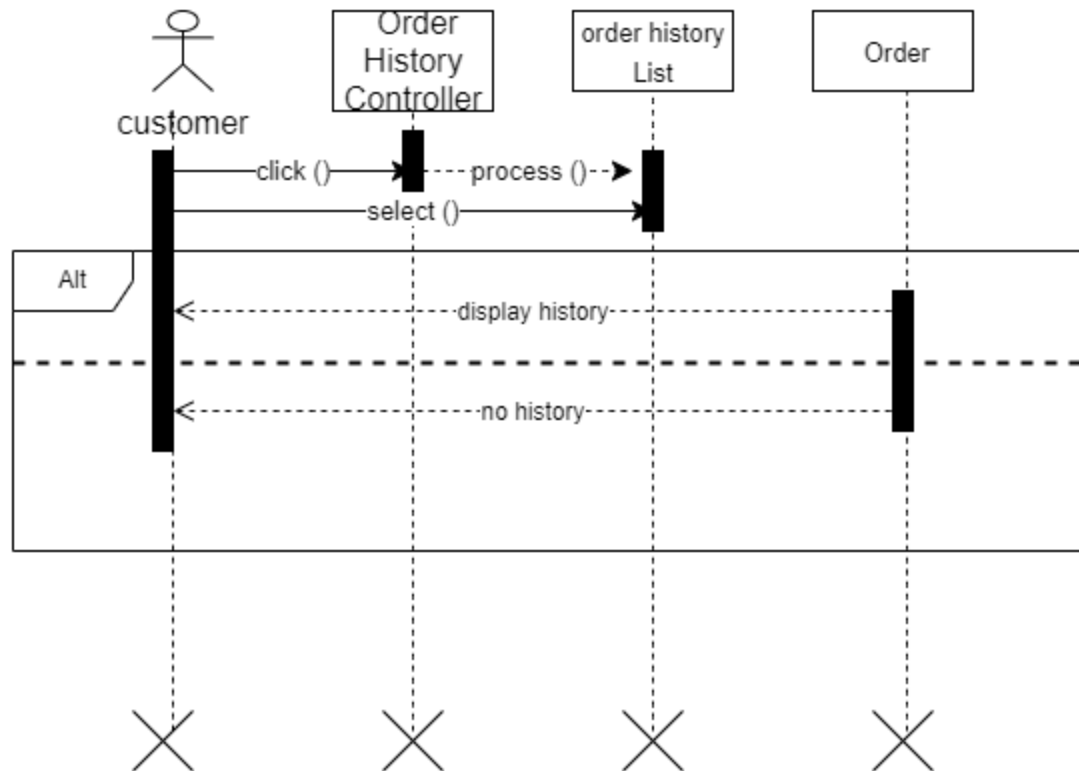


Fig 2.10 View Order History sequence diagram

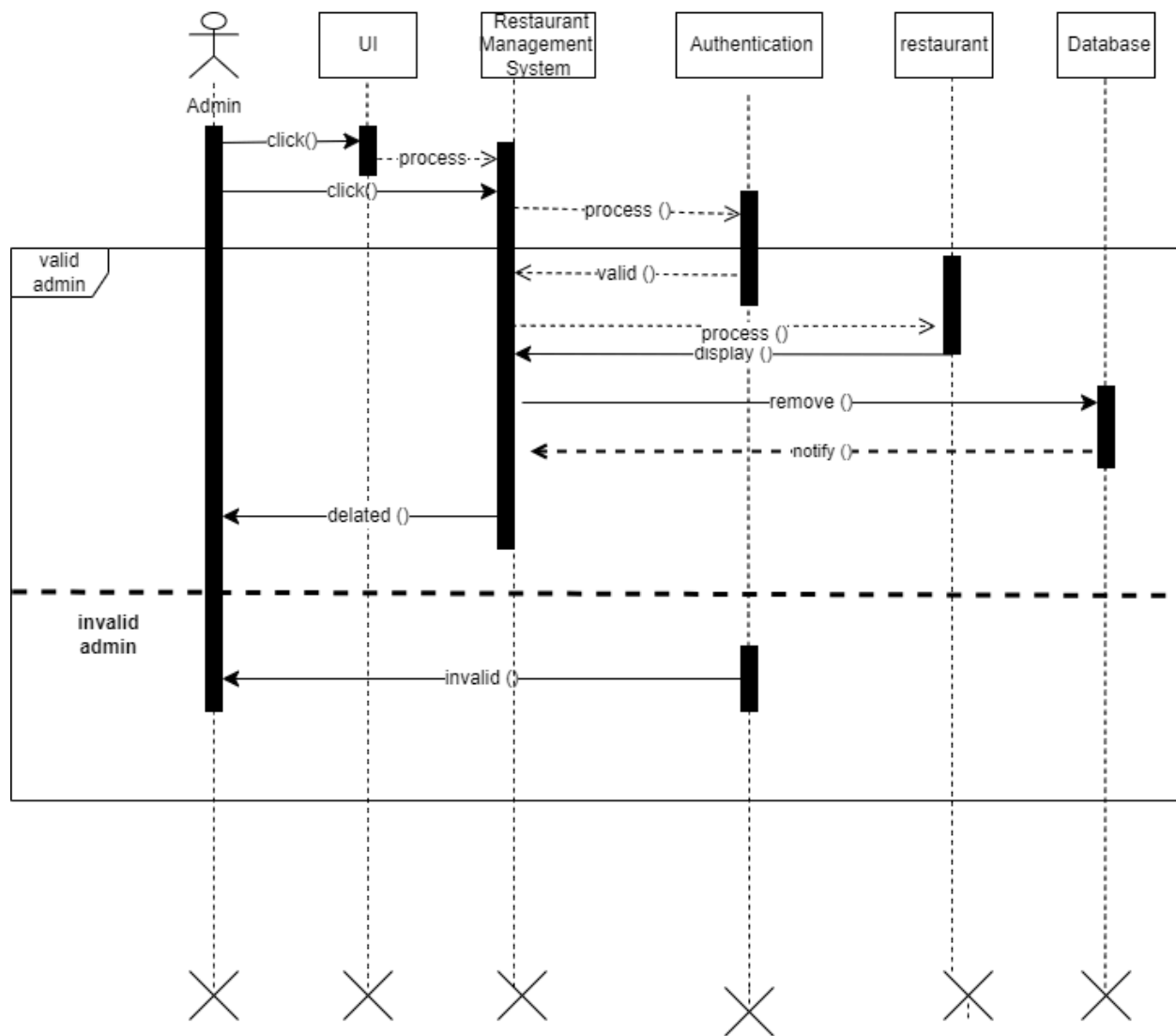


Fig 2.11 Delete restaurant Sequence Diagram

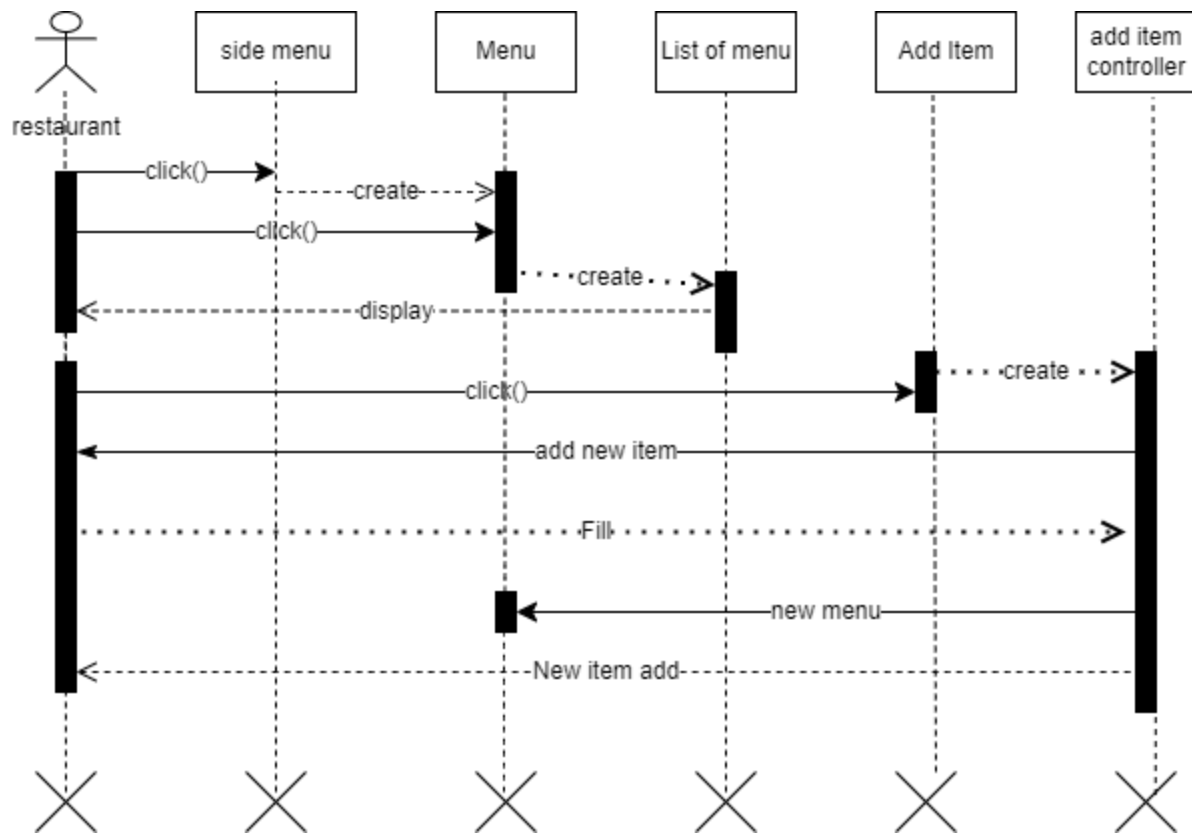


Fig 2.12 Add menu item sequence diagram

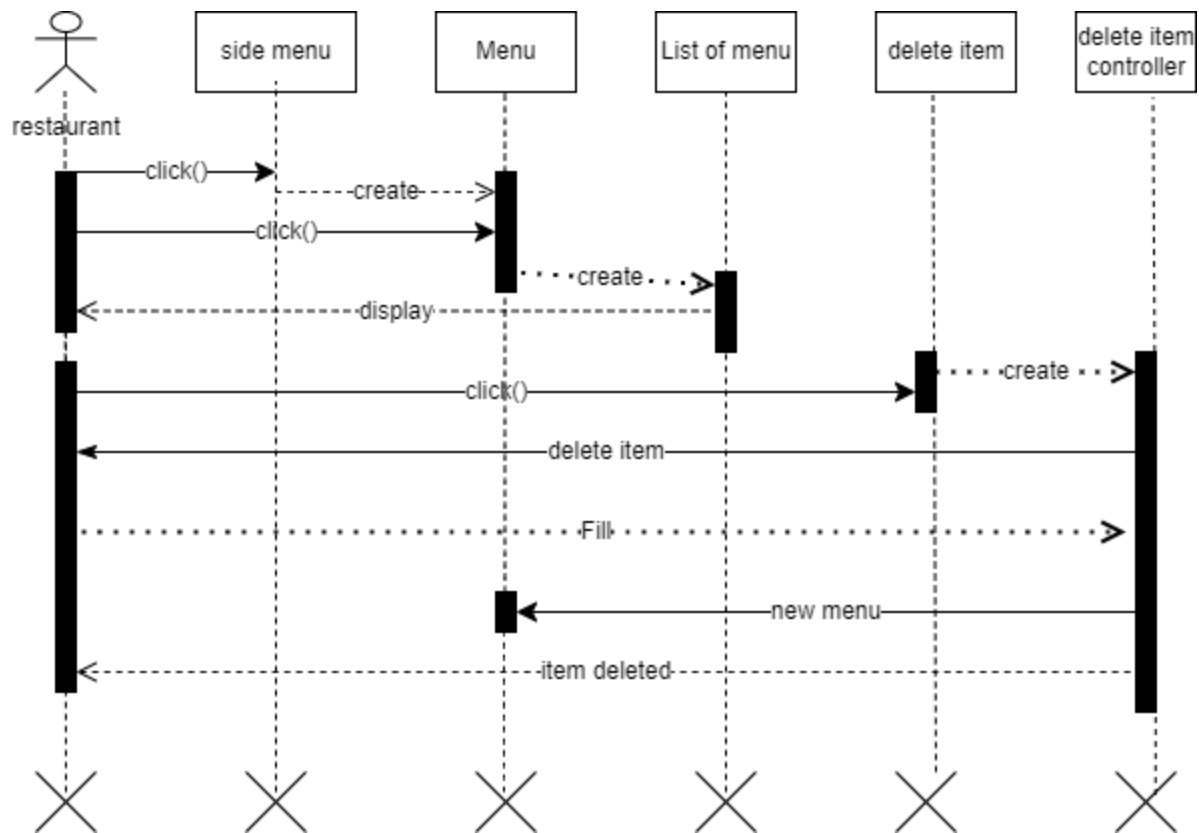


Fig 2.13 Delete Menu Item Sequence Diagram

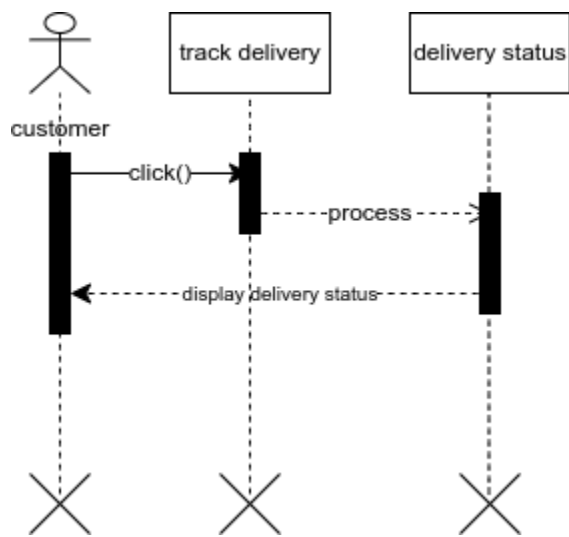


Fig 2.14 Track Delivery Sequence Diagram

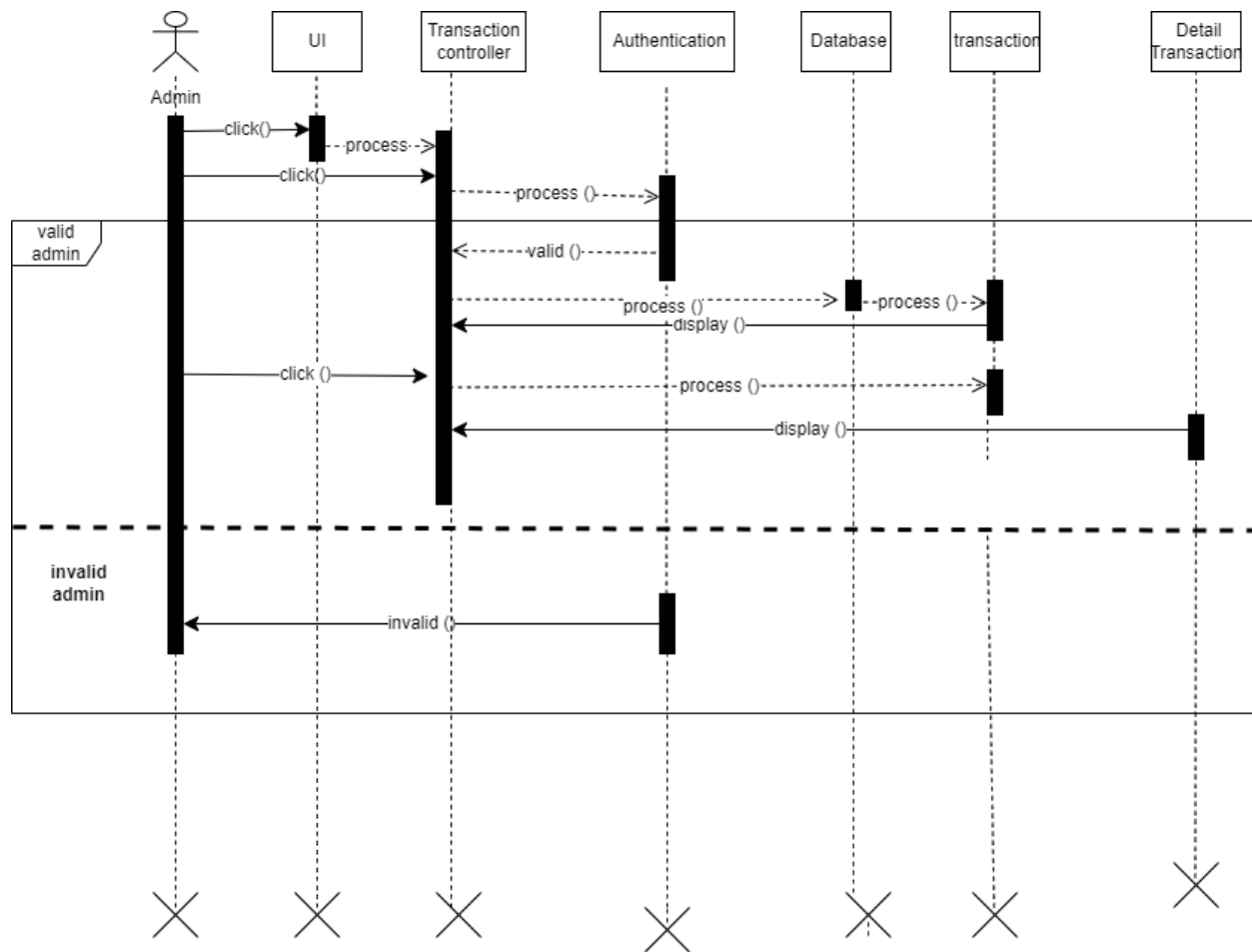


Fig 2.15 View Transaction Sequence Diagram

2.5.4.2 Activity Diagram

Admin

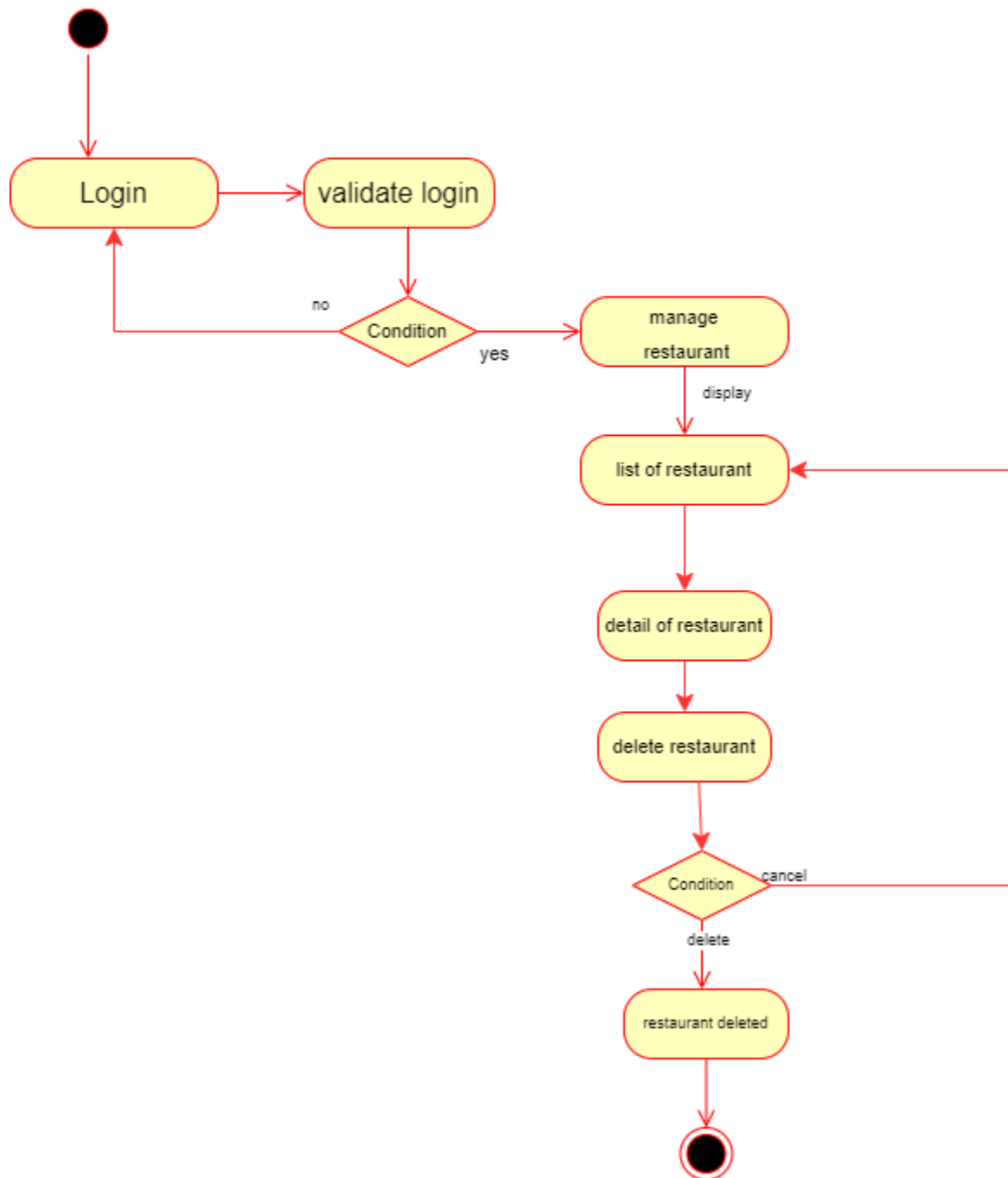


Fig 2.16 Activity Diagram for Delete restaurant

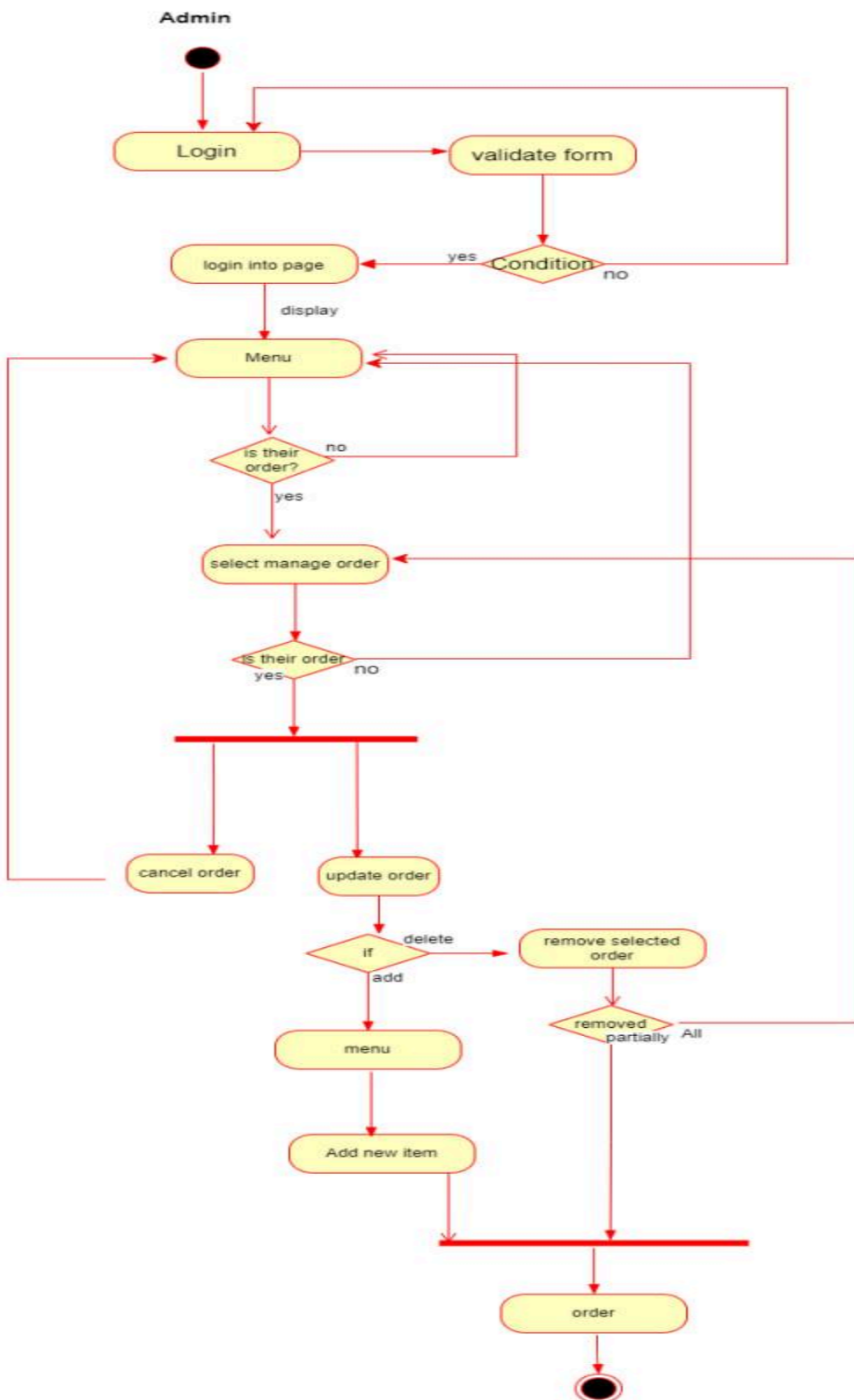


Fig 2.17 Activity Diagram for Deleting Order

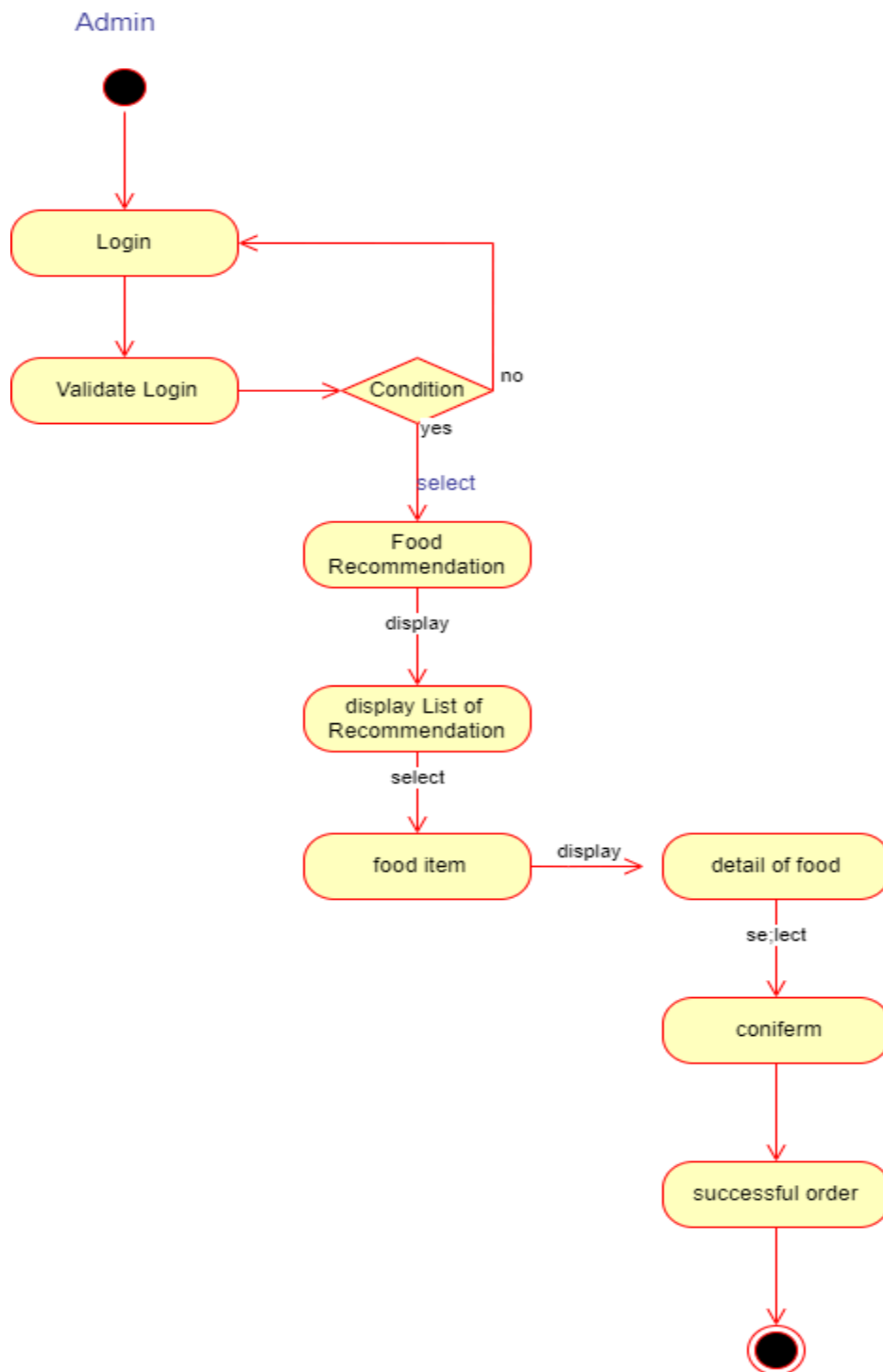


Fig 2.18 Activity Diagram for Food Recommendation By Admin

2.5.4.3 State Machine Diagram

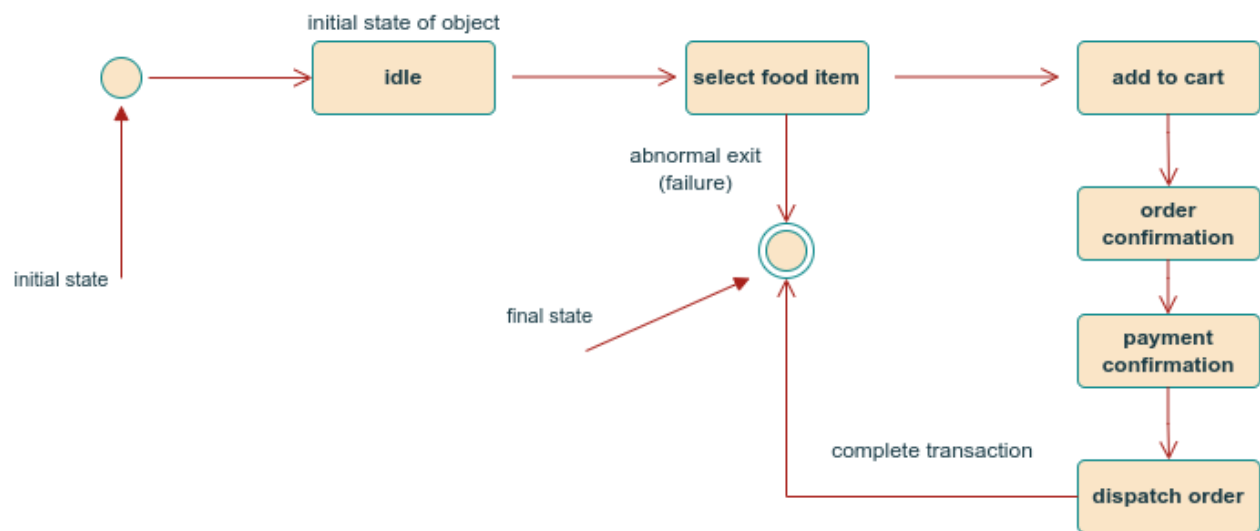


Fig 2.19 State Chart Diagram For Order

2.5.5. User Interface

User interface is a mechanism by which the actor interacts with the system either for query, input or update of information. Figure 2-26 - Figure 2-29 are used to illustrate the user interface of the system.

Fig 2.20 Admin Page for EnblaFoods

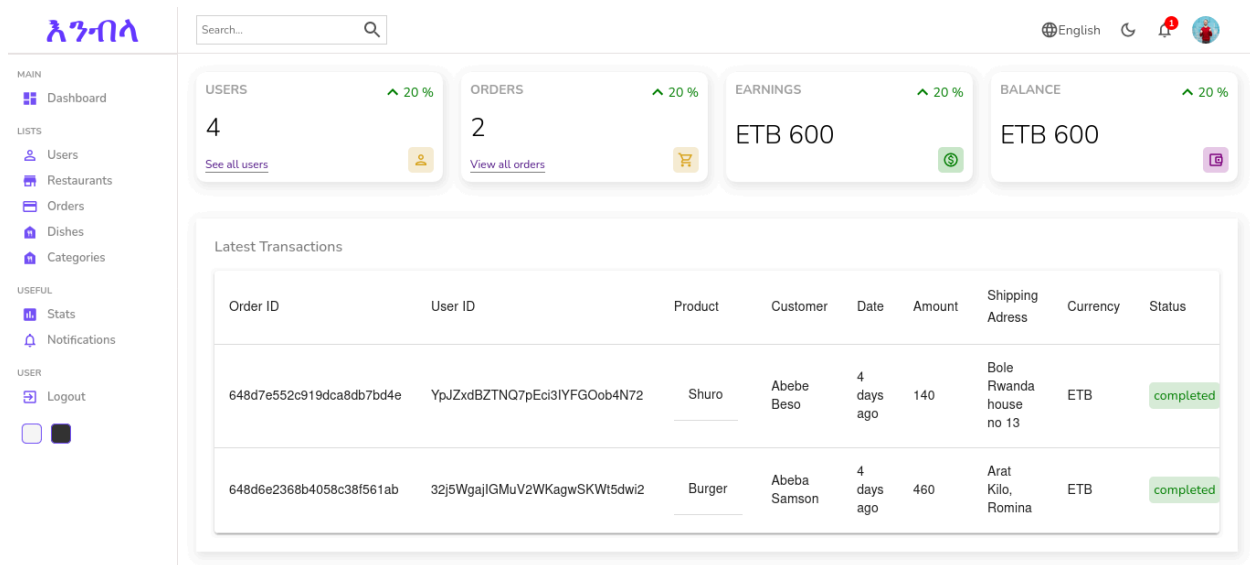


Fig 2.21 Login Interface for EnblaFoods

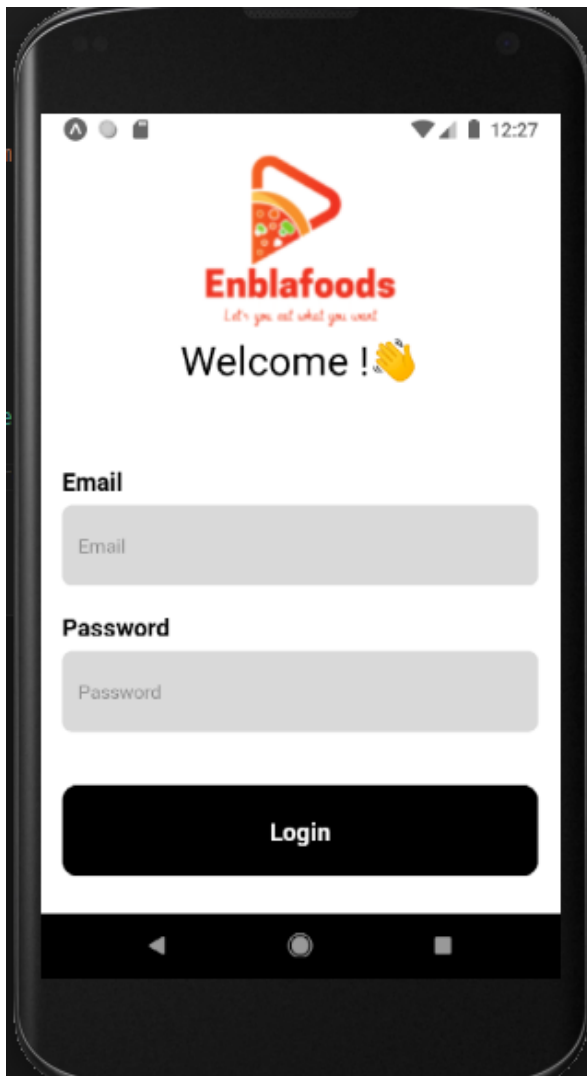


Fig 2.22 Choose menu For EnblaFoods

3. CHAPTER THREE - SYSTEM DESIGN

3.1 Introduction

The food delivery and recommendation mobile app, “EnblaFoods”, will be designed to provide users with personalized food recommendations based on food rating and admin suggestion, while also allowing them to place an order with a third-party delivery service. The app will use an efficient recommendation algorithm to suggest dishes that users may like, making it easier for them to decide what to eat. In addition, the app will provide a seamless ordering experience, allowing users to easily place their orders, track delivery status, and receive notifications on order updates. The third-party delivery service will handle the logistics of delivering the food to the user's doorstep, ensuring fast and reliable delivery.

3.2 Design Goals

The design goals of the system are derived mainly from the non-functional requirement. It helps us in identifying the qualities that the system is optimized to satisfy. The following four major quality perspectives are used to identify the design goals of the system.

Performance

Performance describes the speed and space requirement that is expected from the system.

- **Response time:** As the system is a mobile app-based, responses to remote user requests should be optimized.
- **Throughput:** As the system will be accessed by different users, so it is required to accomplish several users' requests.

Dependability

Dependability refers to the design goals set to minimize system crashes and their consequences. The following are identified as the main dependability criteria.

- **Reliability:** The system must be reliable by using Operating System provided backup systems.
- **Robustness:** The system should have the ability to withstand invalid user input by validating all the inputs of the user.
- **Availability:** The system should be available to everyone wherever internet connection is available.
- **Fault tolerance:** The system should be fault tolerant by displaying warning messages to the user even when erroneous data is inputted.
- **Security:** The system should have a secure authentication mechanism so as to prevent unauthorized users from accessing non-allowed resources.

Maintainability

Maintainability determines behavior of the system to maintain after deployment when new functionalities will be observed.

- **Extensibility:** New functionalities and classes can easily be added.
- **Modifiability:** The system should be easily modifiable after deployment without affecting the current working system.

End User Criteria

- **Usability:** The system should be a user-friendly system so that users can easily access and use it.

3.3 Proposed software architecture

3.3.1 Overview

The most suitable software architecture for our project is the microservice architecture. The key components are:

- User Interface (UI) - Responsible for presenting the food recommendations and handling user interactions.
- Recommendation Service - Uses rating and peoples choice to recommend food to users.
- Food Catalog Service - Stores and retrieves information about available food items.
- Order Management Service - Manages the ordering process and communicates with the delivery service.
- Payment Service - Handles payment processing for orders.
- Delivery Service - Coordinates with third-party delivery providers to deliver food to customers.

Each service can be developed, deployed, and scaled independently, allowing for better maintainability and scalability. Communication between services can be done through APIs

3.3.2 Subsystem Decomposition

System decomposition describes the division of the system into subsystems that are a collection of classes, associations, operations, events, and constraints that are closely interrelated with each other and the responsibilities of each subsystem.

EnblaFoods has the following subsystems.

- User management subsystem
- Order management subsystem
- Inventory management subsystem
- Payment subsystem
- Food Recommendation subsystem
- Notification Service
- Delivery Subsystem

By breaking down the mobile app into these subsystems, the app can be developed, maintained, and scaled more efficiently, while also providing a clear separation of concerns and reducing the complexity of the overall system Coordination with third-party delivery providers. REST APIs to communicate with other services, such as order management.

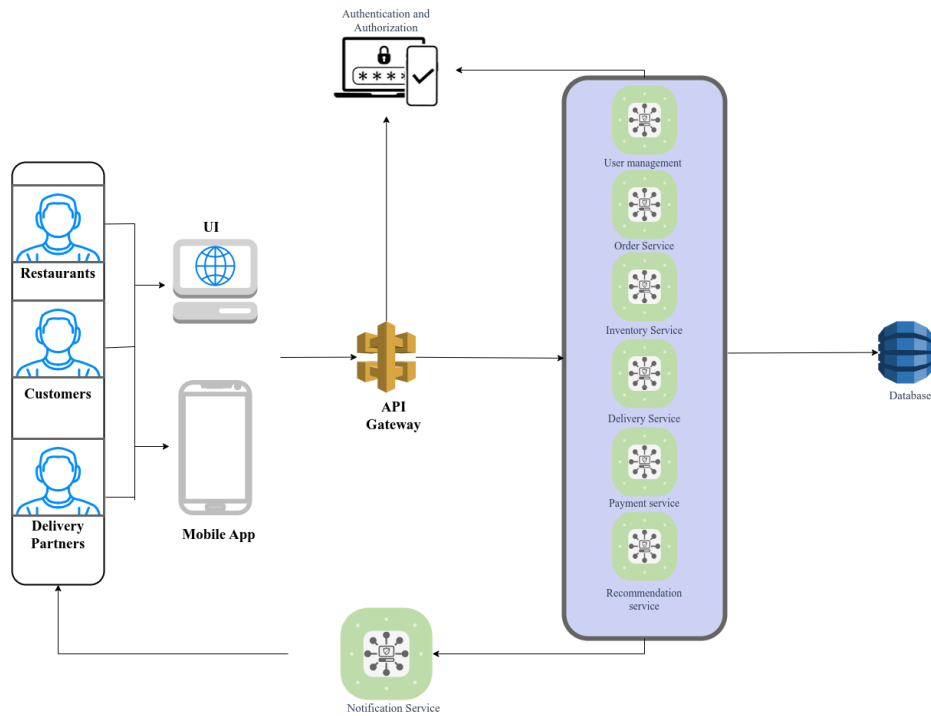


Figure 3.1 System decomposition

The system decomposition and the interaction among the subsystems are shown in figure 3.1.

Subsystem decomposition for the food delivery mobile app that has a third-party delivering service includes the following components:

1. **Recommendation service:** This component includes the food recommendation subsystem as it is responsible for providing food recommendations to the user, based on rating and number of users bought the food.

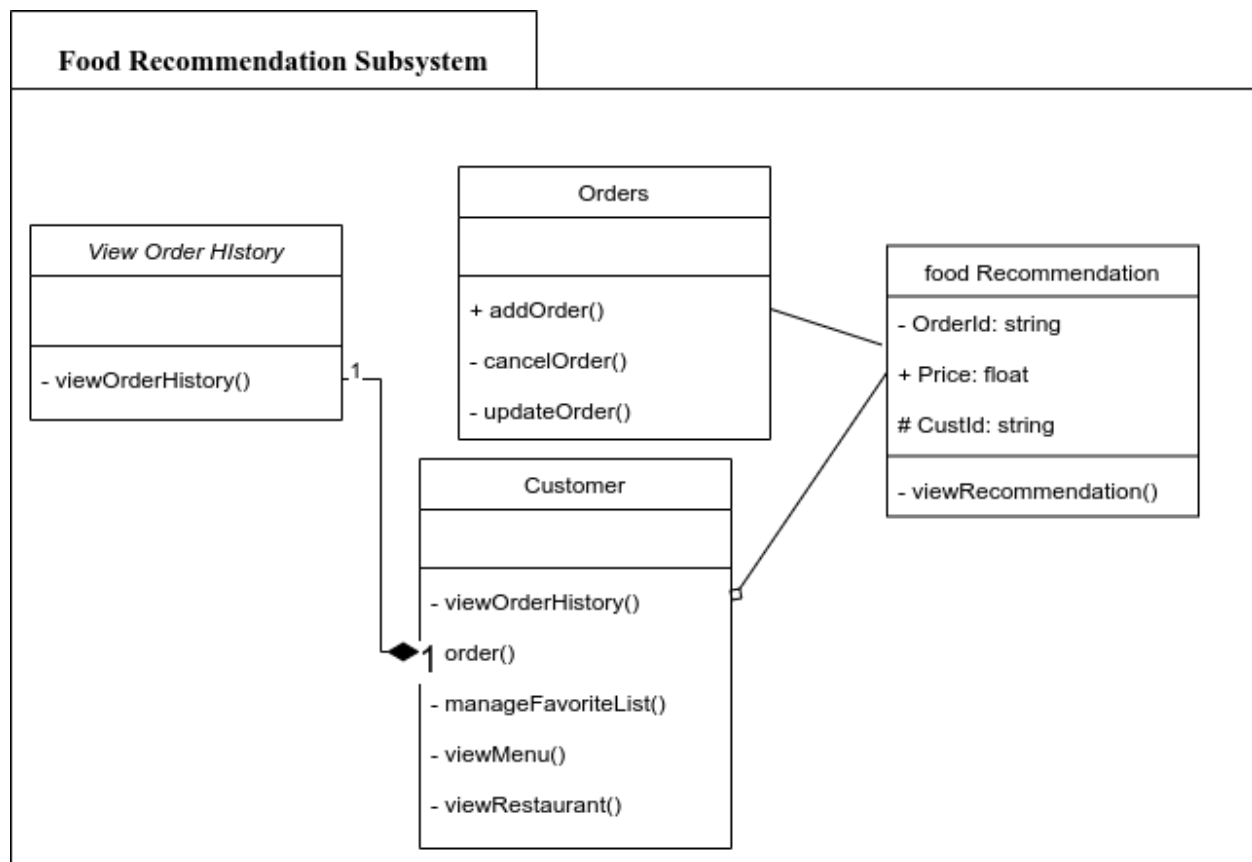


Figure 3.2 Food Recommendation Subsystem

2. **Order service:** This component includes the Order management subsystem as it handles the processing of user orders, including the calculation of prices, the management of payment and delivery information, the transmission of orders to the delivery service and payment status.

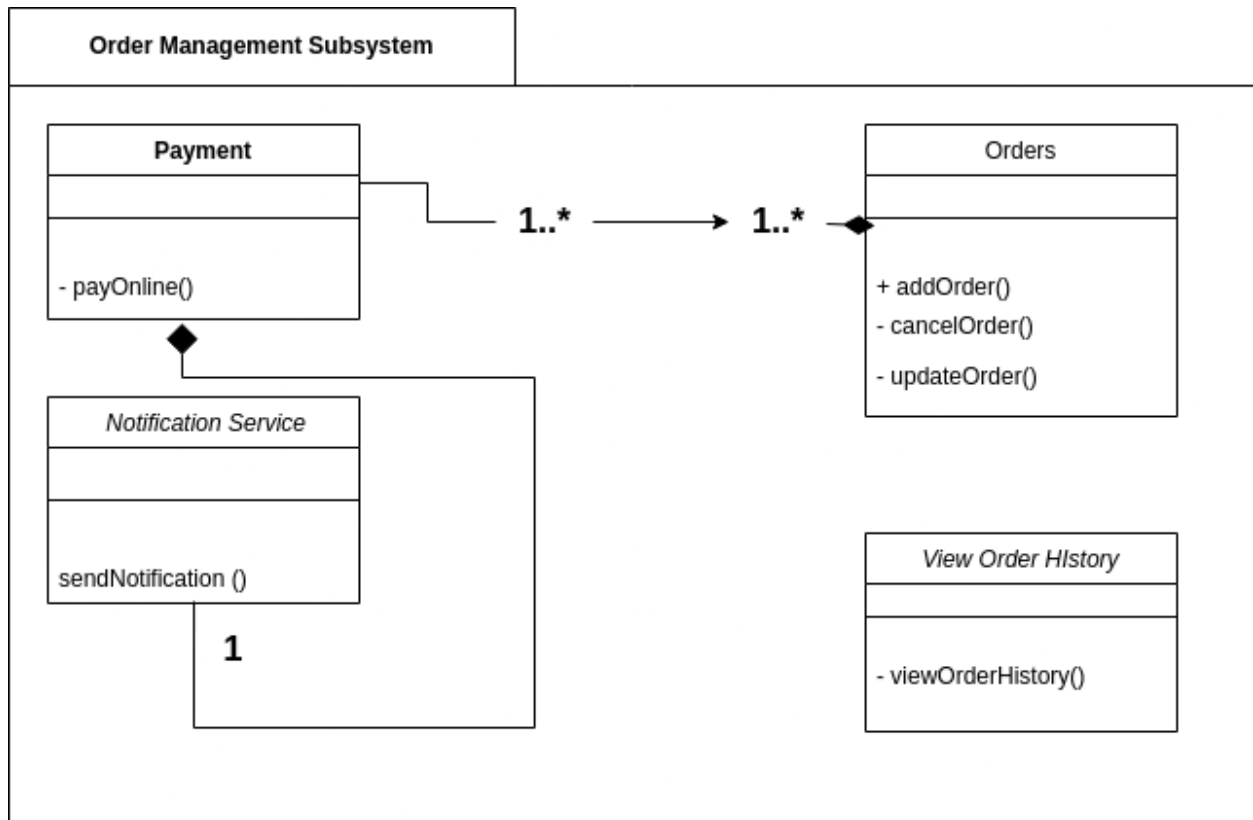


Figure 3.3 Order Management subsystem

3. **User service:** This component includes a user management subsystem that handles user authentication, user registration, and the storage of user information, including payment information, delivery addresses, and food preferences.

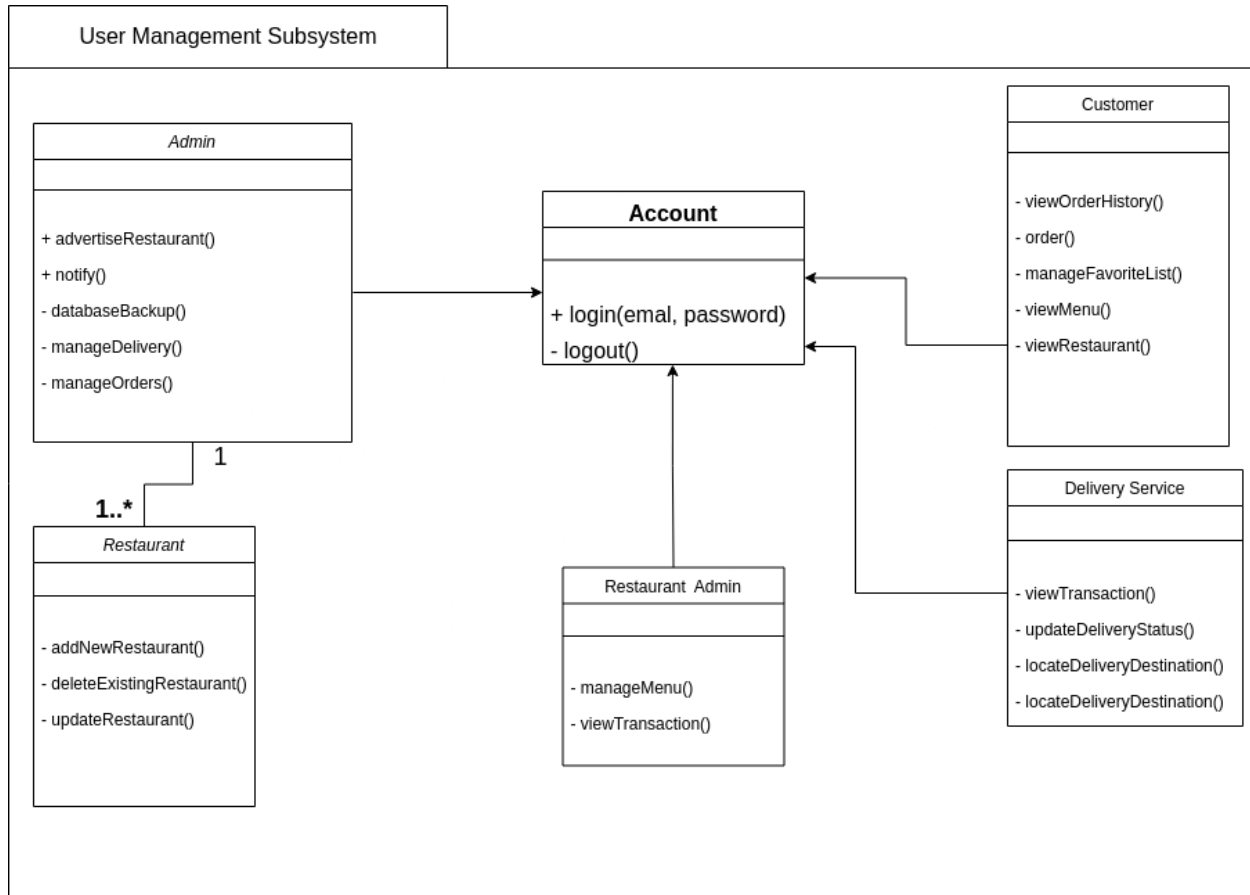


Figure 3.4 User Management subsystem

4. **Inventory service:** This component includes an inventory management subsystem as it maintains a record of the food items available for ordering, their prices, and the restaurants that offer them.

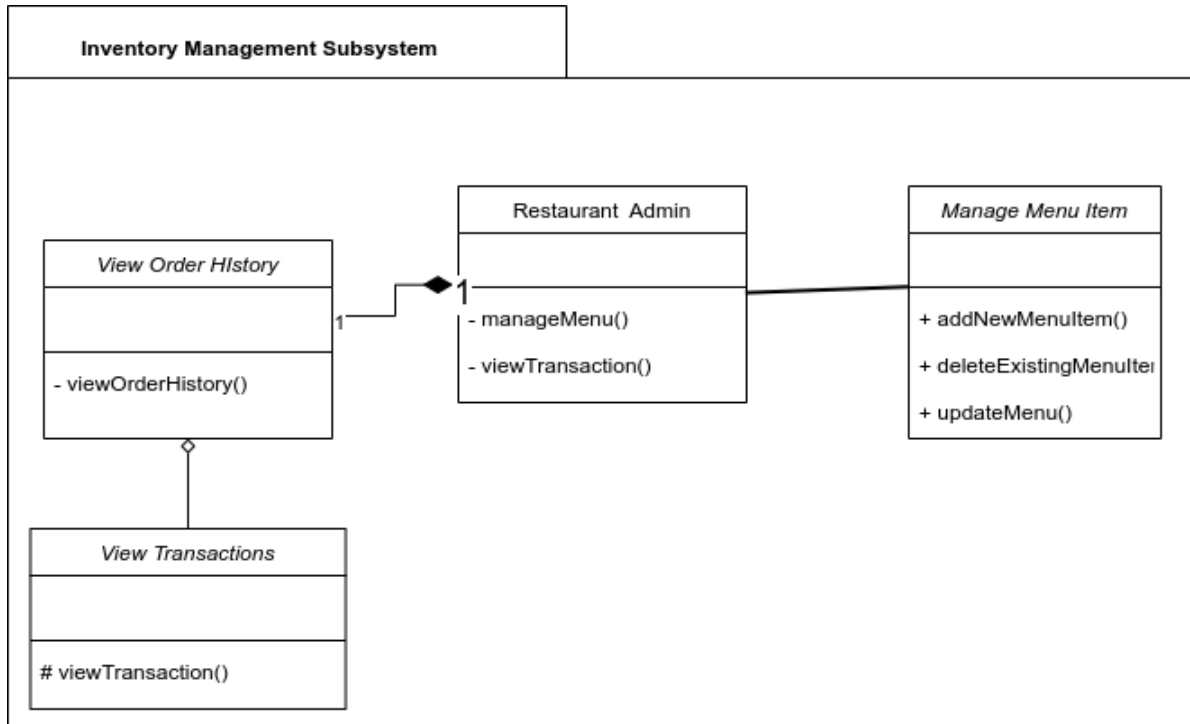


Figure 3.5 Inventory Management subsystem

5. **Notification service:** This subsystem is responsible for sending notifications to the user, including order confirmation, delivery updates, and recommendations.

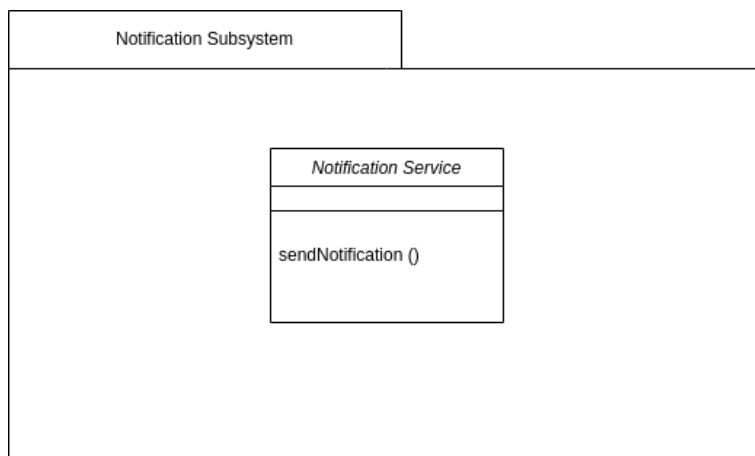


Figure 3.6 Notification Service

6. **Payment service:** This service includes a payment subsystem that handles the processing of payments, including the transmission of payment information to the payment gateway, the management of payment transactions, and the storage of payment information for future use.

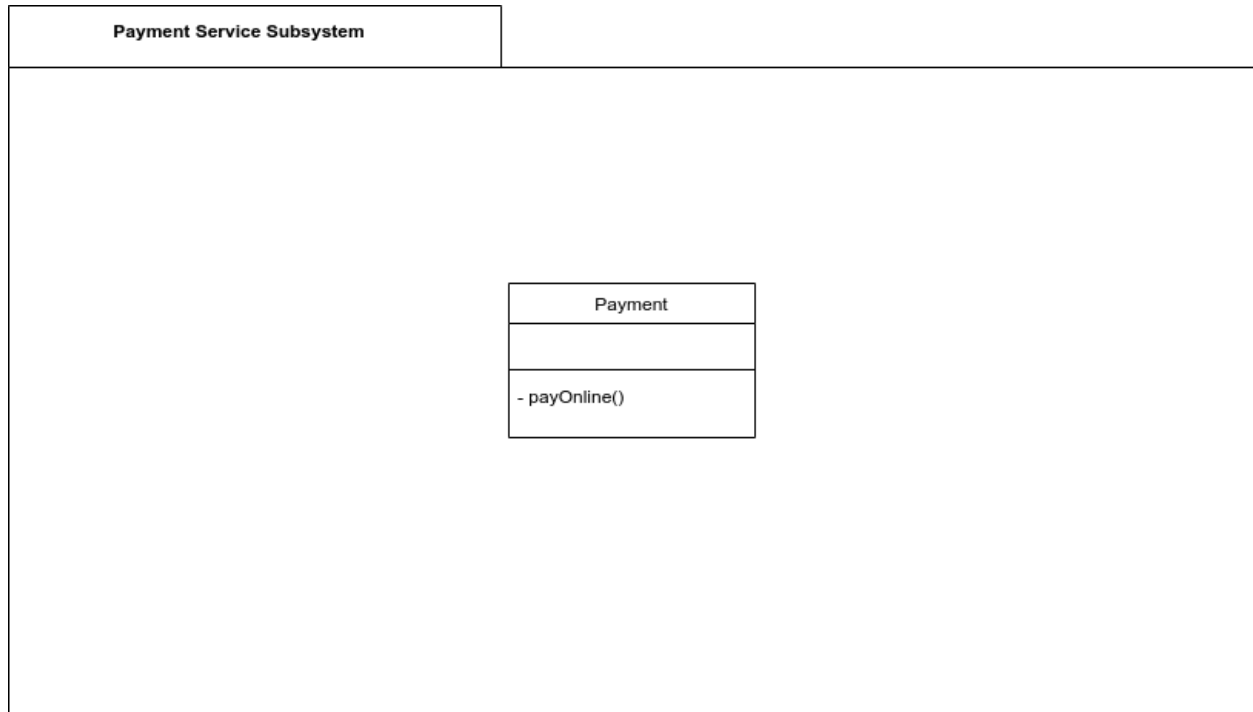


Figure 3.7 Payment Service

7. **Delivery service:** This component includes a delivery subsystem that is responsible for the logistics of delivering the food to the user's doorstep. The delivery service can be a third-party delivery service or an in-house delivery service.



Figure 3.8 Delivery Service

3.3.3 Hardware/software mapping

The hardware/software mapping for our app is be as follows:

1. **Mobile client:** The user interacts with the app through a mobile client running on their smartphone. The client-side software will be developed using React Native.
2. **Server:** The server-side software will be developed using backend development frameworks, such as express,nodeJS and use services like firebase.
3. **Database:** The app stores information in a NoSQL database, such as MongoDB . The database can be hosted on the same cloud-based infrastructure as the server or on a separate database server.
4. **Payment gateway:** The app integrates with a payment gateway, such as Chapa,Telebirr to handle the processing of payments. The payment gateway can be accessed through APIs provided by the payment gateway provider.
5. **Delivery service:** The delivery service is provided by a third-party delivery service.. The app integrates with the delivery service through APIs provided by the delivery service provider.

By mapping the software components to appropriate hardware resources, the app can be developed, deployed, and maintained in a scalable, reliable, and cost-effective manner.

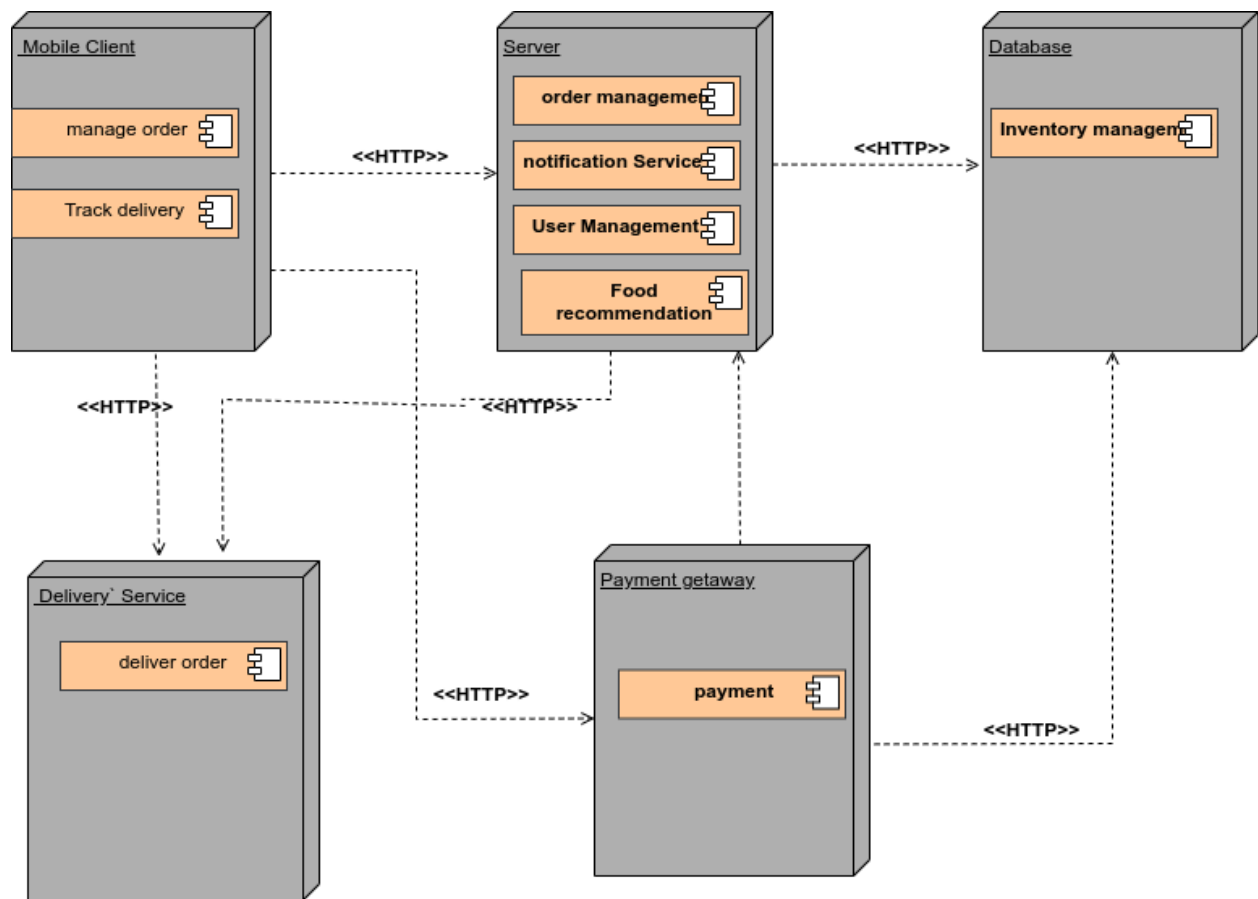


Fig 3.9 Hardware/software mapping

3.9 Persistent Data Management

Persistent data management refers to the storage and retrieval of data that needs to persist beyond the lifetime of individual sessions or requests. This data includes user information, food preferences, order history, delivery information, payment information, and restaurant information.

A suitable persistent data management solution for our app is a NoSQL database called MongoDB and firestore because NoSQL databases are best suited for data that has a flexible schema.

To ensure data consistency, the database will be designed with proper constraints, such as unique keys, foreign keys, and data validation rules. To ensure data security, the database is protected with encryption and access control mechanisms.

By carefully managing the persistent data for the food delivery mobile app, the app can provide reliable and efficient services to the users, while also ensuring the security and privacy of the data.

1. User Account Entity

Attribute Name	Data Type	length
userId	string	10
username	string	30
password	string	30
role	string	30

Table 3.1 user account entity description

2. System Admin Entity

Attribute Name	Data Type	length
userId	string	10
username	string	30
password	string	30
role	string	30

Table 3.2 system admin entity description

3. Restaurant Admin

Attribute Name	Data type	length
restaurantName	String	50
restaurantId	String	10
username	String	10
email	String	30
password	String	30

Table 3.3 restaurant admin entity description

4. Order Entity

Attribute Name	Data Type	Length
orderid	String	30
price	double	10
orderTime	date	N/A
transactionNum	String	20
customerId	String	10
deliveryStatus	String	10
ItemNum	int	5
itemList	Array	30

Table 3.4 order entity description

5. Customer Entity

Attribute Name	Data Type	Length
custId	string	10
cusFName	string	30
cusLName	string	30
email	string	30
custId	string	10
username	string	30
password	string	30

Table 3.5 customer entity description

6.Transaction Entity

Attribute Name	Data Type	Length
transactionId	string	30
transactionType	string	50
custId	string	10
deliveryServiceId	string	10
dateofTransaction	date	N/A
restaurantId	string	10

Table 3.6 transaction entity transaction

7. Food Item Entity

Attribute Name	Data Type	Length
itemId	string	30
itemName	string	50
foodType	string	50
restaurantId	string	10
unitPrice	double	10

Table 3.7 food item entity description

8. Delivery Service Entity

Attribute Name	Data Type	Length
deliveryServiceId	String	10
deliveryServiceName	String	50
username	string	10
password	string	30
email	String	30

Table 3.8 delivery service entity description

9. Payment Entity

Attribute Type	Data Type	Length
orderId	string	30
price	double	10
custId	string	10

Table 3.9 payment entity description

3.3.5 Access Control and Security

The access control and security of EnblaFoods can be achieved through the following measures:

- **User authentication:** Users should be required to provide a unique identifier, such as a username and password or a mobile phone number and OTP, to access the app and their personal information.
- **Data encryption:** Sensitive data, such as passwords, payment information, and personal information, should be encrypted both in transit and at rest to protect against unauthorized access.
- **Authorization:** Access to specific functionality within the app, such as ordering food or managing payment information, should be restricted based on a user's role and permissions.
- **Secure communication:** All communication between the mobile app and the backend services should be encrypted using SSL/TLS or similar protocols to protect against eavesdropping and tampering.

- **Regular security updates:** The app and the backend services should be updated regularly to address known security vulnerabilities and to incorporate the latest security features.
- **Penetration testing:** Regular penetration testing should be conducted to identify and address potential security risks in the app and the backend services.
- **Incident response plan:** A plan should be in place to respond to security incidents, such as data breaches or unauthorized access, in a timely and effective manner.

Subsystem	class	operation	user			
			Customer	Restaurant	Admin	Delivery
User management	Admin	advertiseRestaurant()			✓	
		notify()			✓	
		databasebackup()			✓	
		managedelivery()			✓	
		manageorder()	✓			

	Account	login(email,password)	✓	✓	✓	✓
		logout()	✓	✓	✓	✓
	Customer	order()	✓			
		viewrestaurant()	✓		✓	
		managefavorite()	✓			
		viewmenu()	✓	✓	✓	
	DElivery Service	viewtransaction()	✓	✓	✓	✓
		update delivery status()				✓
		locatedeliverydestination()	✓		✓	✓
	Restaurant Admin	Manage menu()		✓		

		Viewtransaction() ()	✓	✓	✓	✓
	Restaurant	addnewRestaurant() ()			✓	
		deleteExistingRestaurant() ()			✓	
		updateRestaurant() ()			✓	
Order management	Orders	addOrder() ()	✓			
		cancelOrder() ()	✓			
		Update Order() ()	✓			
	Payment	payOnline() ()	✓			
	Notification	sendNotification() ()	✓	✓	✓	✓

	View Order History	viewOrderHistory	✓		✓	✓
Payment Service	payment	payonline()	✓			
Notification	Notification Service	sendNotification()	✓	✓	✓	✓
Inventory management	Vieworder history	Vieworderhistory()	✓		✓	✓
	Restaurant admin	manage menu()		✓		
		viewtransaction()	✓	✓	✓	✓
	Manage menu item	addNewMenuItem()		✓		
		DeleteExistingMenuItem()		✓		
		updateMenuItem()		✓		

	View transaction	Viewtransaction()	✓	✓	✓	✓
Food Recommendation	Orders	addOrder()	✓			
		cancelOrder()	✓			
		Update Order()	✓			
	Food Recommendation	viewRecommendation()	✓			
	View Order History	viewOrderHistory()	✓		✓	✓
	Customer	viewOrderHistory()	✓			
		order()	✓			
		manage Favorite List()	✓			
		viewMenu()	✓			

		viewRestaurant() ()	✓			✓
Delivery Service	Delivery	viewtransaction() ()	✓	✓	✓	✓
		updateDeliveryStatus() ()	✓		✓	✓
		LocateDeliveryDestination() ()	✓		✓	✓

3.3.6 Subsystem services

The subsystem services that EnblaFoods includes:

- **User management:** Allows users to create accounts, update their profiles, and manage their preferences.
- **Food recommendation:** Recommends food items based on the rating and number of people who bought the food.
- **Order management:** Allows customers to place orders, view order history, and track delivery status.
- **Payment management:** Handles payment processing, including secure storage of payment information, payment authorization, and payment settlement.

- **Delivery management:** Coordinates with the third-party delivery service to provide delivery status updates.
- **Restaurant management:** Allows restaurants to manage their menu items, prices, and availability.
- **Inventory management:** Manages the availability of food items and ingredients in real-time, to ensure that only available items can be ordered.
- **Reporting and analytics:** Provides insights into app usage, order history, and delivery performance, to help improve the app and its services.

By providing these subsystem services, the food delivery mobile app can provide a seamless and integrated experience for its users, while also ensuring that the underlying systems and processes are efficient and reliable

4. CHAPTER FOUR: OBJECT DESIGN DOCUMENT

4.1 Detailed Class Diagram

This section describes the classes in a detailed way and their public interfaces. The overview, dependencies, constraints, attributes, and operations of classes of the system are presented below in the following detail.

4.1.1 Transaction Class Diagram

This class is responsible for Depicting the transaction details of the order.

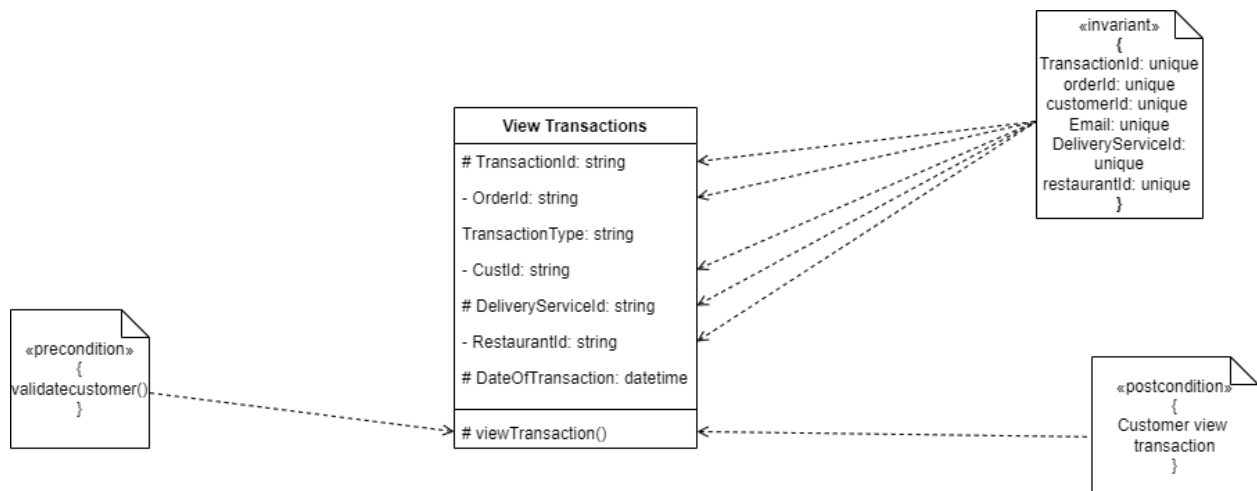


Figure 4.1 Transaction Class Diagram

View Transaction: This method allows the user to view the transaction details of a specific order or all orders. The user can search for transactions by date, order number, or customer name. The method retrieves transaction data from the database and displays it to the user in a readable format.

4.1.2 Food Recommendation Class Diagram

This class is responsible for food recommendations of customers based on their order history and preference.

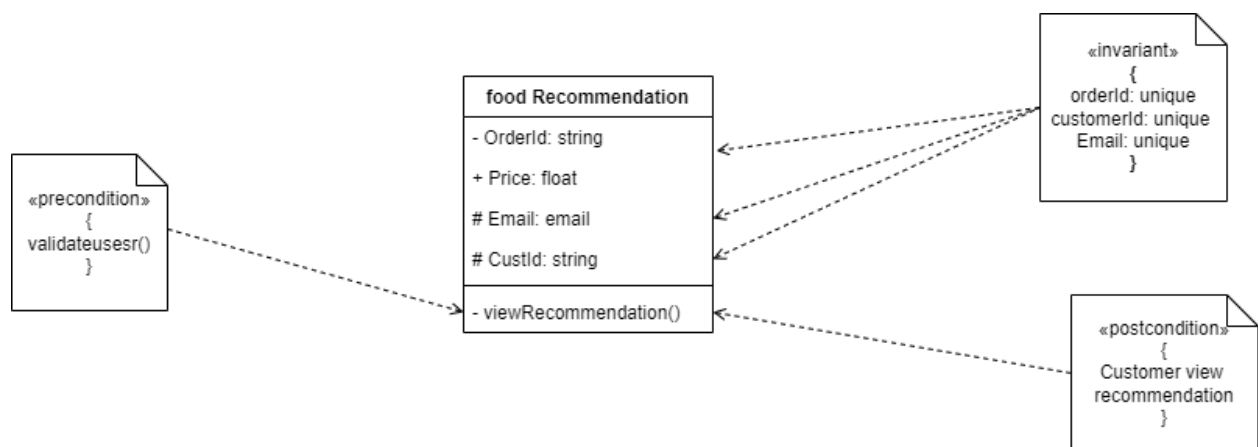


Figure 4.2 Food Recommendation Class Diagram

viewRecommendation: This method allows the user to view the recommended food based on their order.

4.1.3 Manage Menu Item Class Diagram

This class is responsible for managing the menu item of a restaurant in which the restaurant admin can add new menu items, delete existing menu items or update the existing menu item.

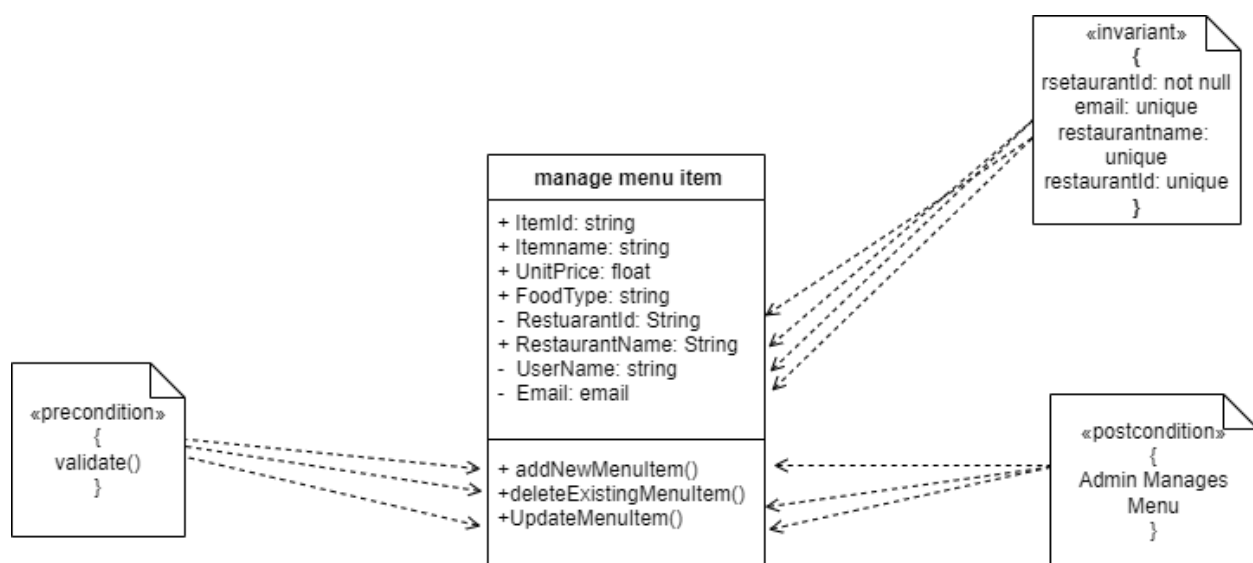


Figure 4.3 Manage Menu Item Class Diagram

Add New Menu Item: This method allows the user to add a new menu item to the restaurant's menu. The user can enter the item name, description, price, and other relevant details. The method validates the input and adds the new menu item to the database.

Delete Existing Menu Item: This method allows the user to delete an existing menu item from the restaurant's menu. The user can select the item to delete, and the method removes the item from the database. The method also updates the orders that contain the deleted item.

Update Menu Item: This method allows the user to update an existing menu item. The user can select the item to update and change its name, description, price, or other relevant details. The method validates the input and updates the item in the database. The method also updates the orders that contain the updated item.

4.1.4 Admin Class Diagram

This class is responsible for the overall management of the system including Advertisement, manage order, manage delivery and backup of the database.

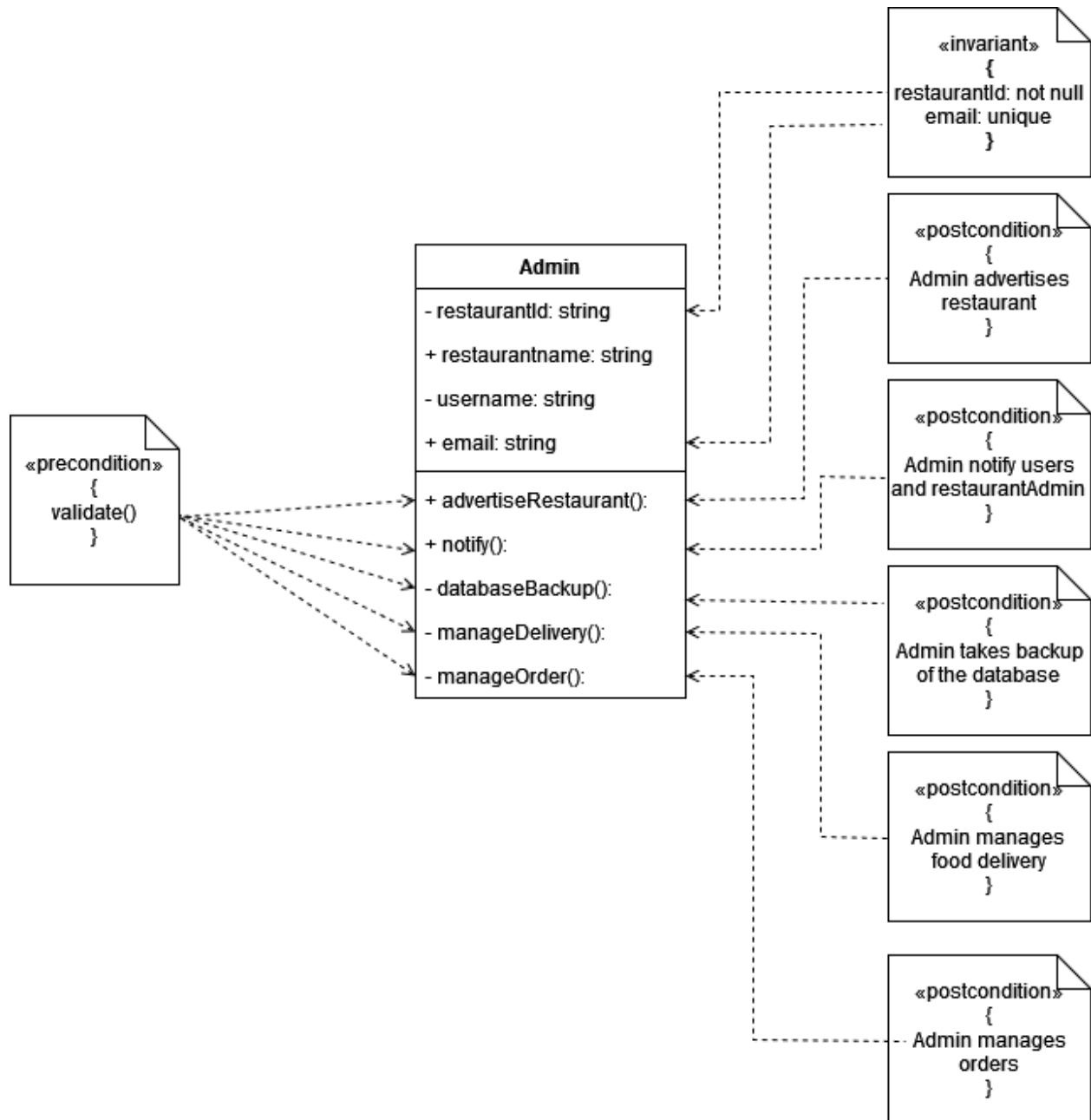


Figure 4.4 Admin Class Diagram

Advertise Restaurant: This method allows the user to advertise the restaurant by sending promotional messages to customers. The user can create a promotional message, select the target audience, and send the message. The method uses a notification system to send the message to the selected customers.

Notify: This method allows the user to send notifications to customers. The user can create a notification message and select the target audience. The method uses a notification system to send the message to the selected customers.

Database Backup: This method allows the user to create a backup of the restaurant's database. The method copies the database to a backup location, ensuring that the data is safe in case of any system failure.

Manage Delivery: This method allows the user to manage the restaurant's delivery system. The user can add or delete delivery personnel, view the delivery schedule, and assign delivery personnel to orders. The method updates the delivery status of orders and the availability of delivery personnel.

Manage Order: This method allows the user to manage orders. The user can view all orders, filter orders by status, assign orders to delivery personnel, and update the order status. The method updates the database with the changes made by the user.

4.1.5 Restaurant Admin class Diagram

This class is responsible for the management of the restaurant by its admin including the restaurant menu and control the transaction of the order.

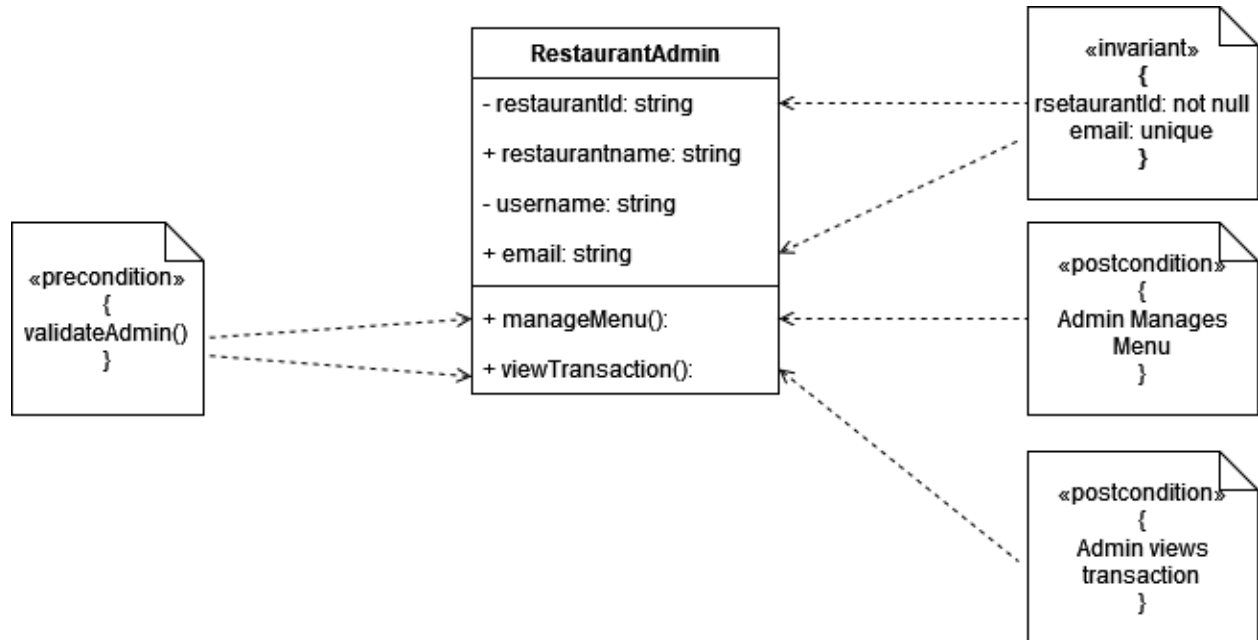


Figure 4.5 Restaurant Admin class Diagram

Manage Menu: Method Description: This method allows the user to manage the restaurant's menu. The user can add, delete, or update menu items. The method updates the database with the changes made by the user.

4.1.6 Customer Class Diagram

This class is responsible for allowing the customers to view a list of restaurants including their menu and view their order history. They can also manage favorite list.

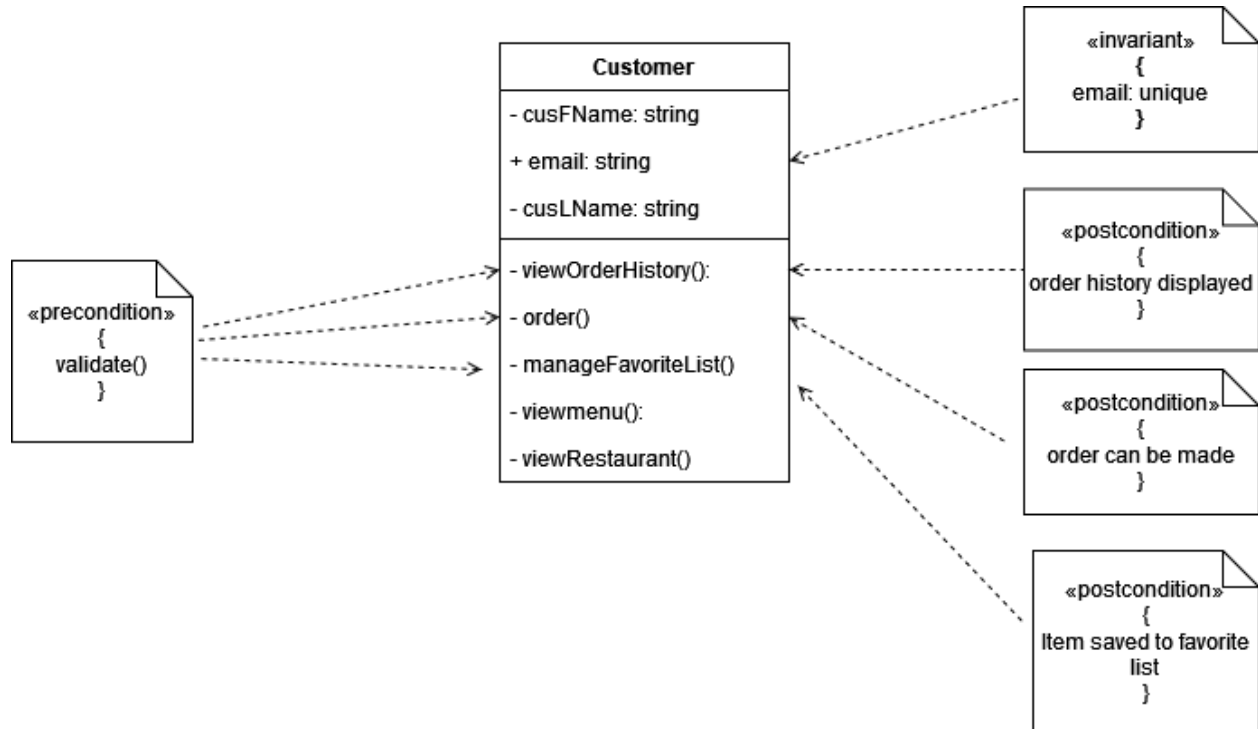


Figure 4.6 Customer Class Diagram

View Order History: This method allows the user to view their order history. The user can filter orders by status, date, or order number. The method retrieves the order data from the database and displays it to the user in a readable format.

Manage Favorite List: This method allows the user to manage their favorite list of menu items. The user can add or delete items from the list. The method updates the user's profile with the changes made by the user.

View Menu: This method allows the user to view the restaurant's menu. The method retrieves the menu data from the database and displays it to the user in a readable format.

View Menu Method: This method allows the user to view the menu of the selected restaurant. The menu items can be listed based on the category, price, or the availability of the item. The user can also view the item description, price, and other details before placing an order.

View Restaurant: This method allows the user to view the restaurant's information, including the address, phone number, opening hours, and ratings. The user can also view the menu and select the items to order from the restaurant.

4.1.7 Account Class Diagram

This class is responsible for creating an environment that allows the system users to interact with the system.

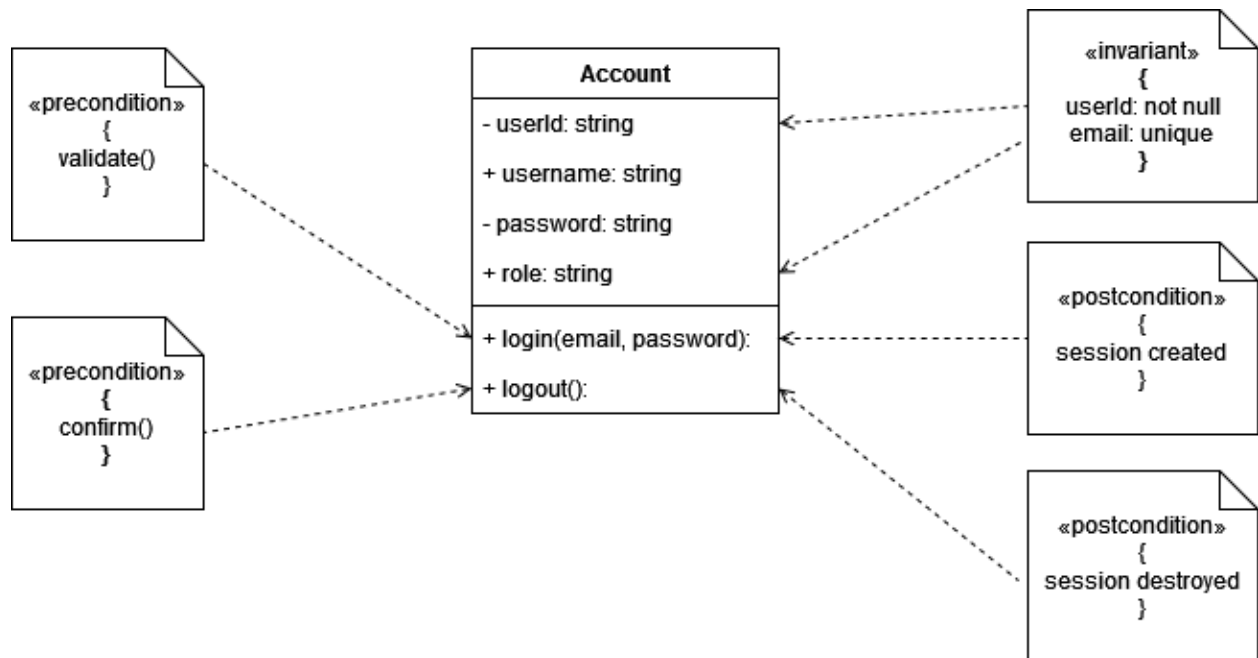


Figure 4.7 Account Class Diagram

4.1.8 Orders Class Diagram

This class is responsible for allowing customers to place orders, cancel orders, and update orders.

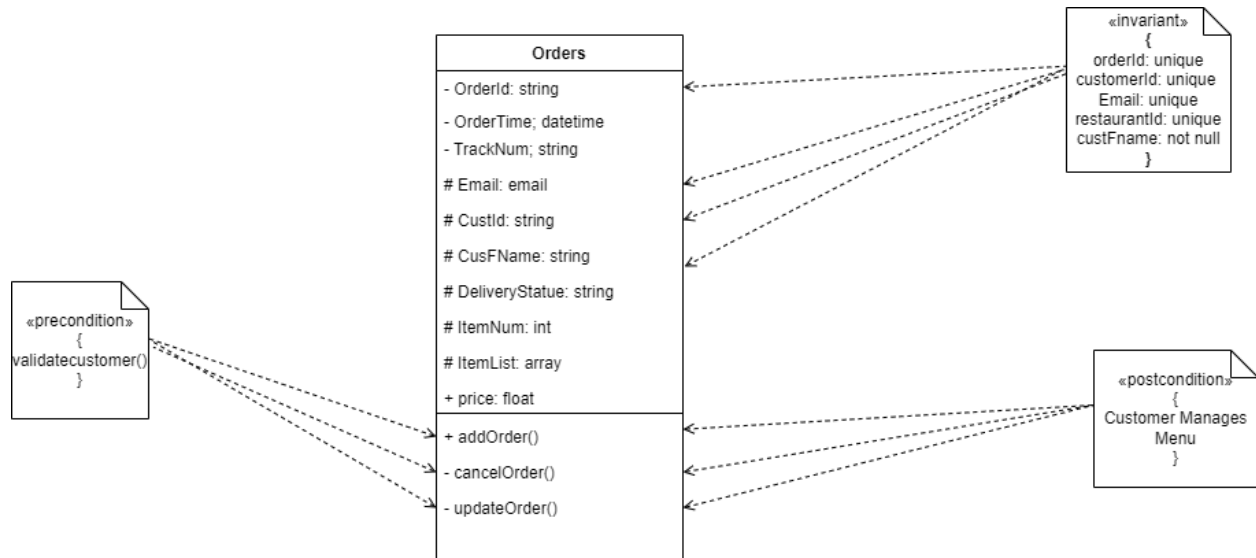


Figure 4.8 Order Class Diagram

Add Order Method: This method allows the user to select the menu items and add them to the order list. The user can also modify the quantity of the items, add special instructions, and view the total price of the order.

Cancel Order Method: This method allows the user to cancel the order before it is confirmed or delivered. The user needs to provide the order details and the reason for canceling the order.

Update Order Method: This method allows the user to modify the order details, such as the quantity of the items, the delivery address, or the payment method. The user needs to provide the order details and the updated information.

4.1.9 Payment Class Diagram

This class is responsible for handling the processing of payments, including the transmission of payment information to the payment gateway, the management of payment transactions, and the storage of payment information for future use.

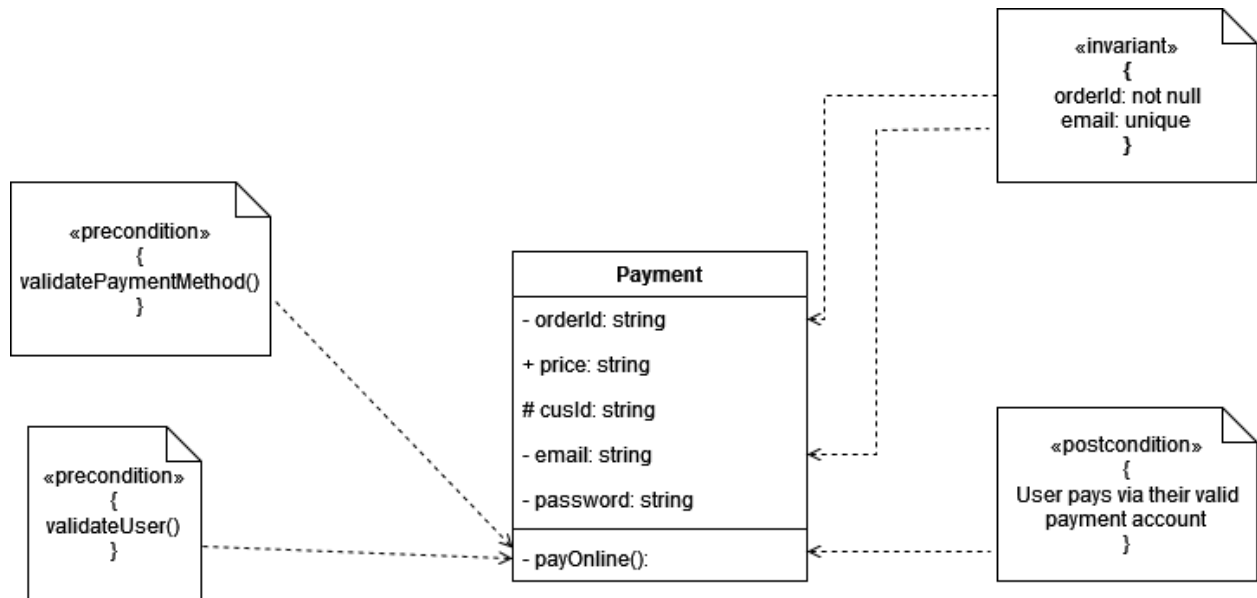


Figure 4.9 Payment Class Diagram

Pay Online: This method allows the user to pay for the order online using a secure payment gateway. The user can select the payment method, enter the payment details, and confirm the payment.

4.1.10 Notification Service Class Diagram

This class is responsible for sending notifications to the user, including order confirmation, delivery updates, and recommendations.

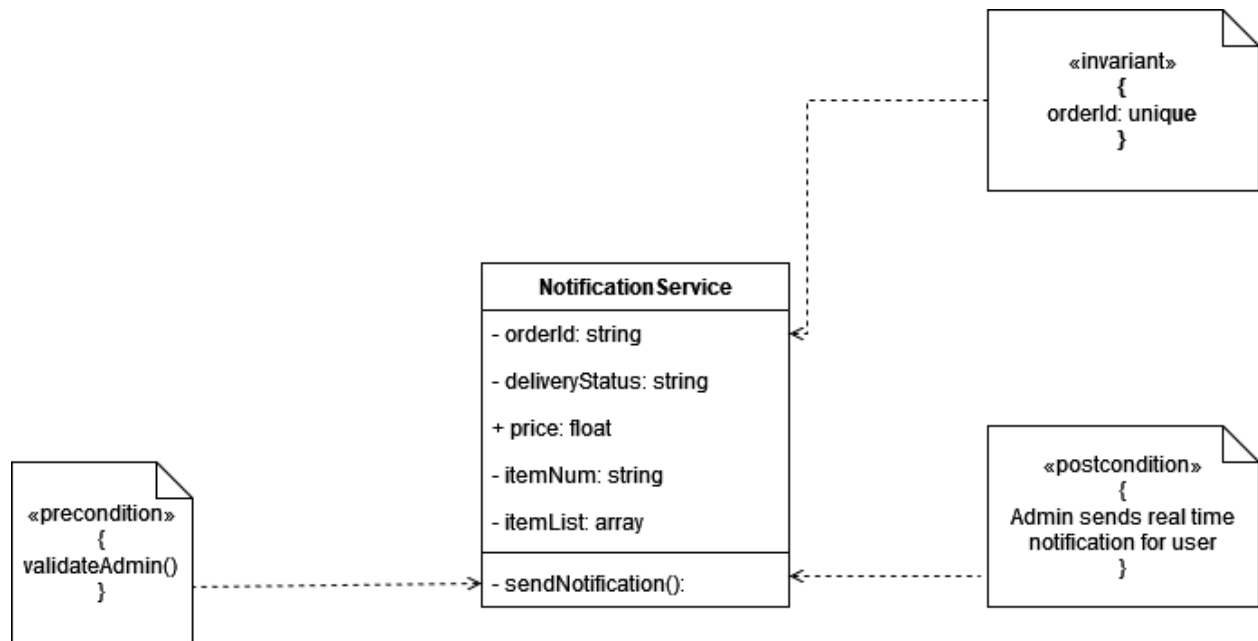


Figure 4.10 Notification Service Class Diagram

Send Notification: This method allows the system to send notifications to the user regarding the order status, payment confirmation, or any other updates related to the order.

4.1.11 Restaurant Class Diagram

This class is responsible for allowing restaurants to manage their menu items, prices, and availability.

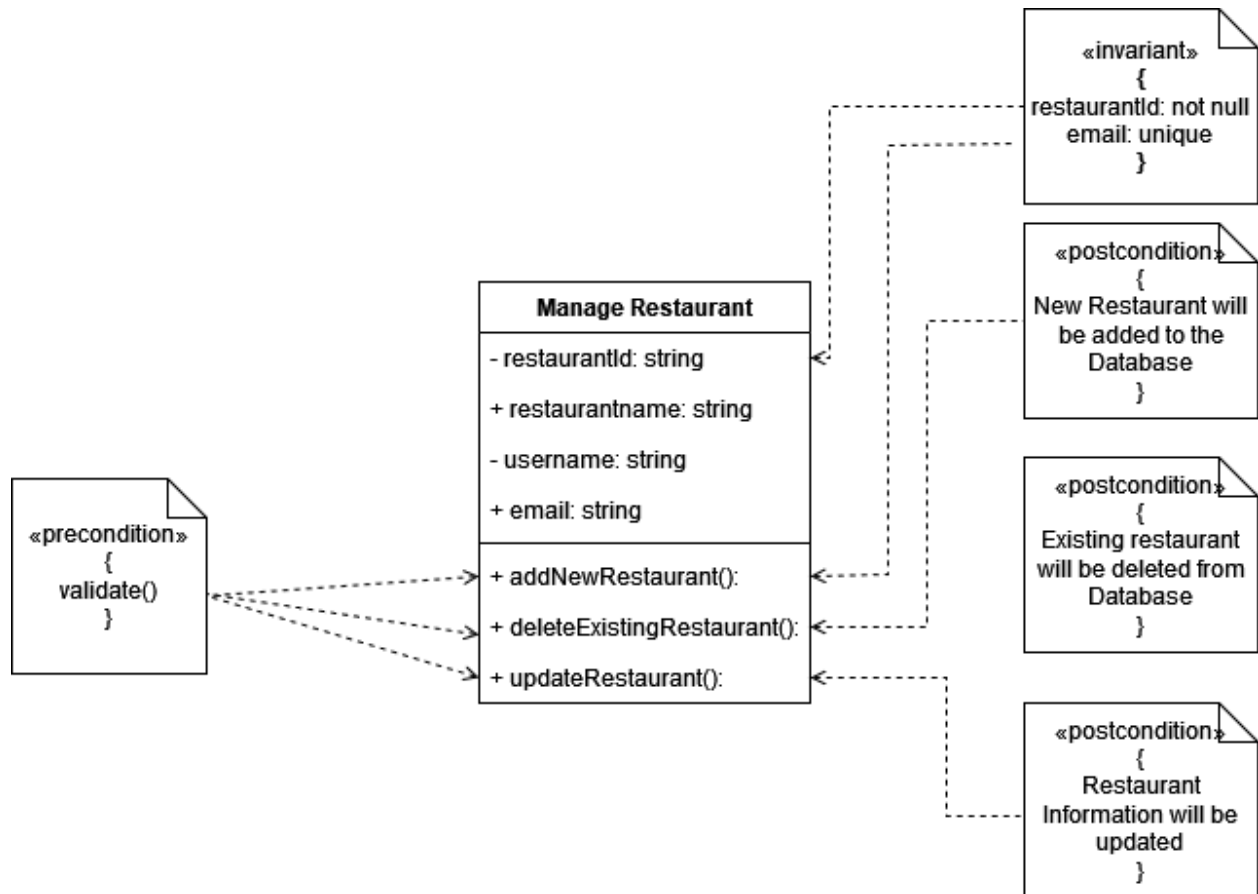


Figure 4.11 Restaurant Class Diagram

Add New Restaurant: This method allows the system administrator to add a new restaurant to the system. The administrator needs to provide the restaurant's information, including the name, address, phone number, and menu items.

Delete Existing Restaurant: This method allows the system administrator to delete an existing restaurant from the system. The administrator needs to provide the restaurant's details and confirm the deletion.

Update Restaurant: This method allows the system administrator to update the restaurant's information, such as the address, phone number, opening hours, or menu items. The administrator needs to provide the restaurant's details and the updated information.

4.1.12 Delivery Service Class Diagram

This component is responsible for the logistics of delivering the food to the user's doorstep. The delivery service can be a third-party delivery service or an in-house delivery service.

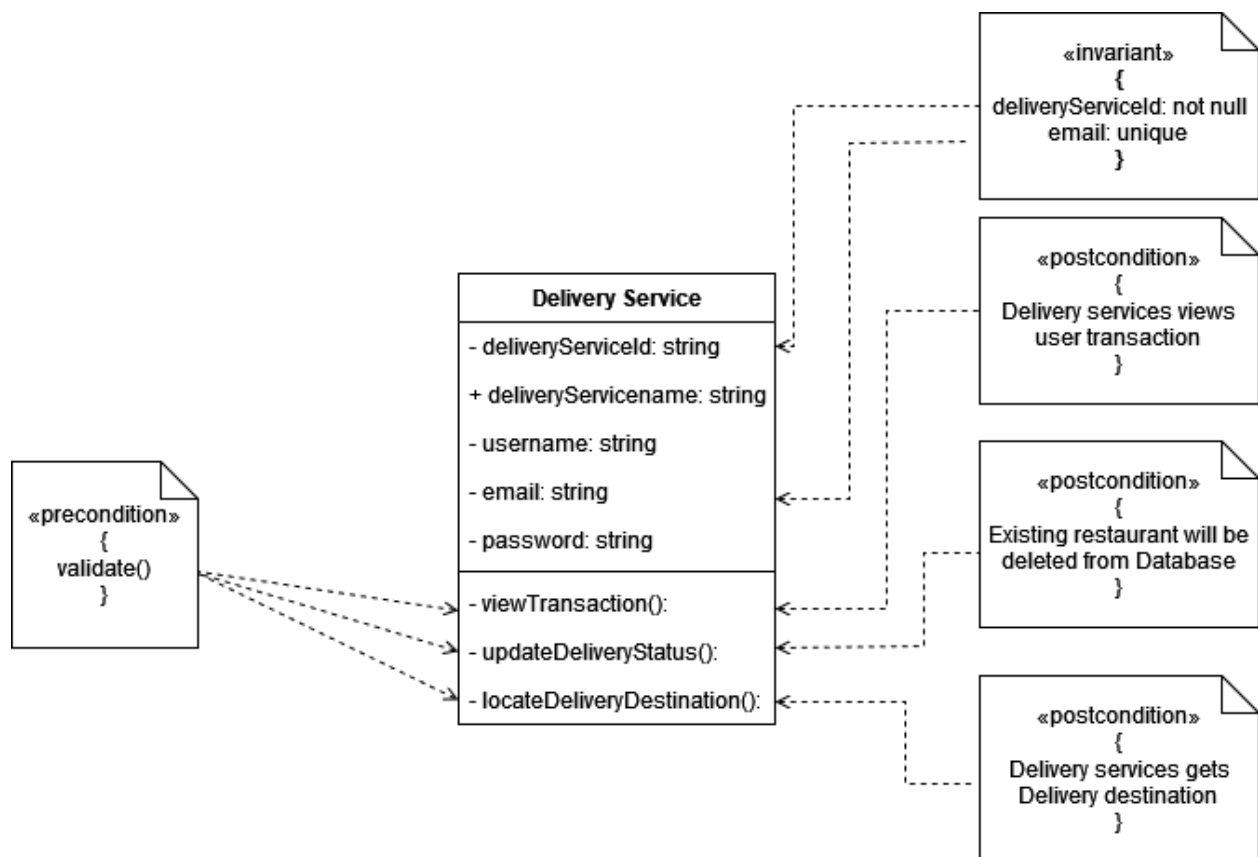


Figure 4.12 Delivery Service Class Diagram

Update Delivery Status: This method allows the delivery person to update the order delivery status, such as out for delivery, delivered, or canceled. The delivery person needs to provide the order details and the status update.

Locate Delivery Destination: This method allows the delivery person to locate the delivery destination. The delivery person can view the delivery address and help the delivery navigate to the destination.

4.2 Packages

In order to reduce the complexity of the application domain, smaller parts that are identified during the analysis called classes are organized into packages.

EnblaFoods has the following packages:

- User Interface
- Data Management
- Order Management
- User Management

- **User Interface Package**

The interface package contains different classes, which control the loading and unloading of the interfaces with which the customers interface different parts of the system

- **Data Management Package**

The data management package contains classes responsible for data storage and information retrieval triggered by the subsystems.

- **Order Management Package**

The order management package contains classes responsible for handling the processing of user orders, payment and delivery information.

Based on the classes, the following diagram is package diagram for EnblaFoods

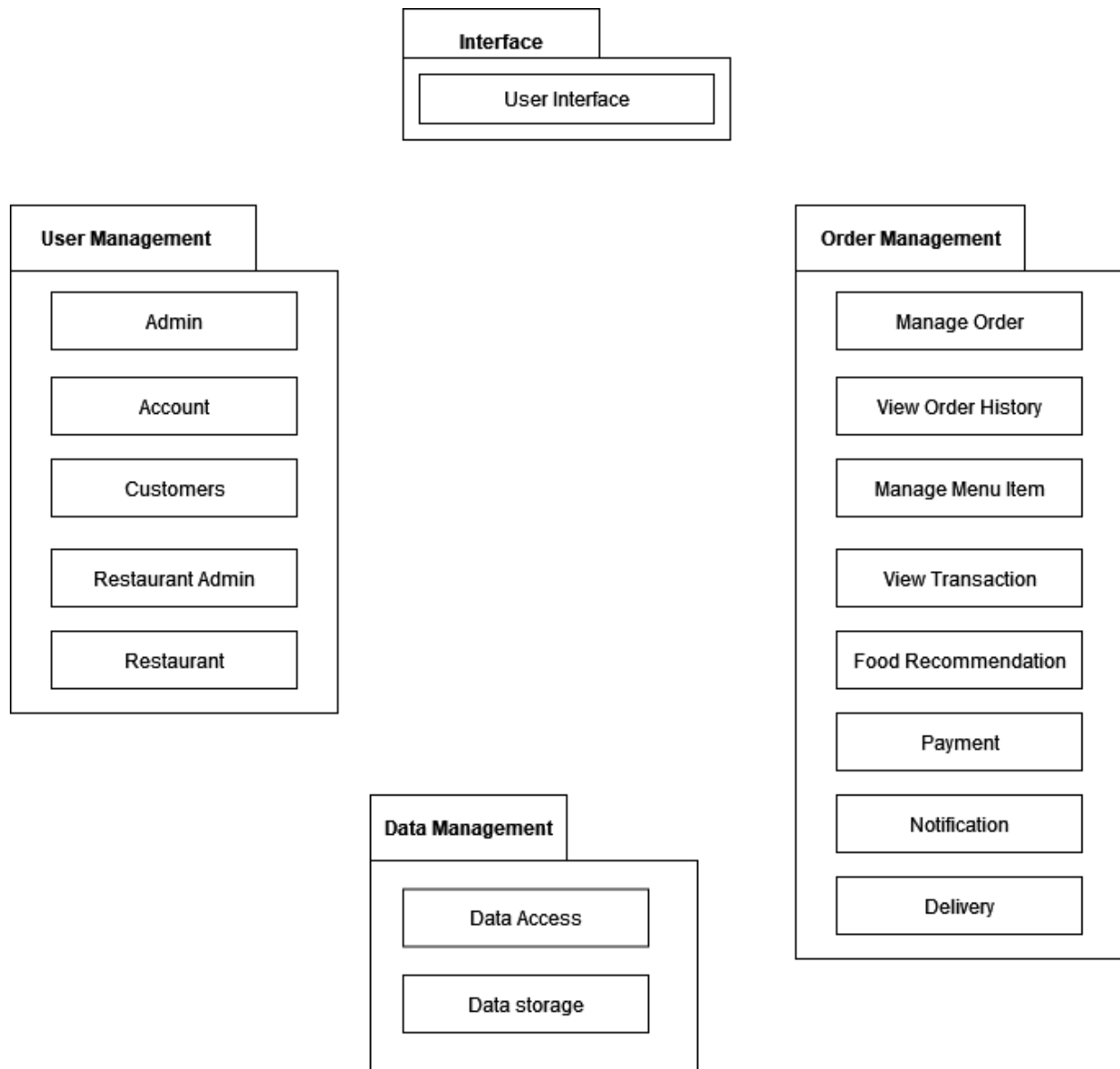


Figure 4.13 Package diagram

Dependency of packages

Users interact with different packages of the systems only through the interfaces designed. We want our system to have high cohesion and less coupling. Therefore we need to reduce dependencies among packages.

We have a user management and order management package which will interact with the Data Management for their storage and retrieval. And those packages can be accessed directly through their interfaces via the user interface

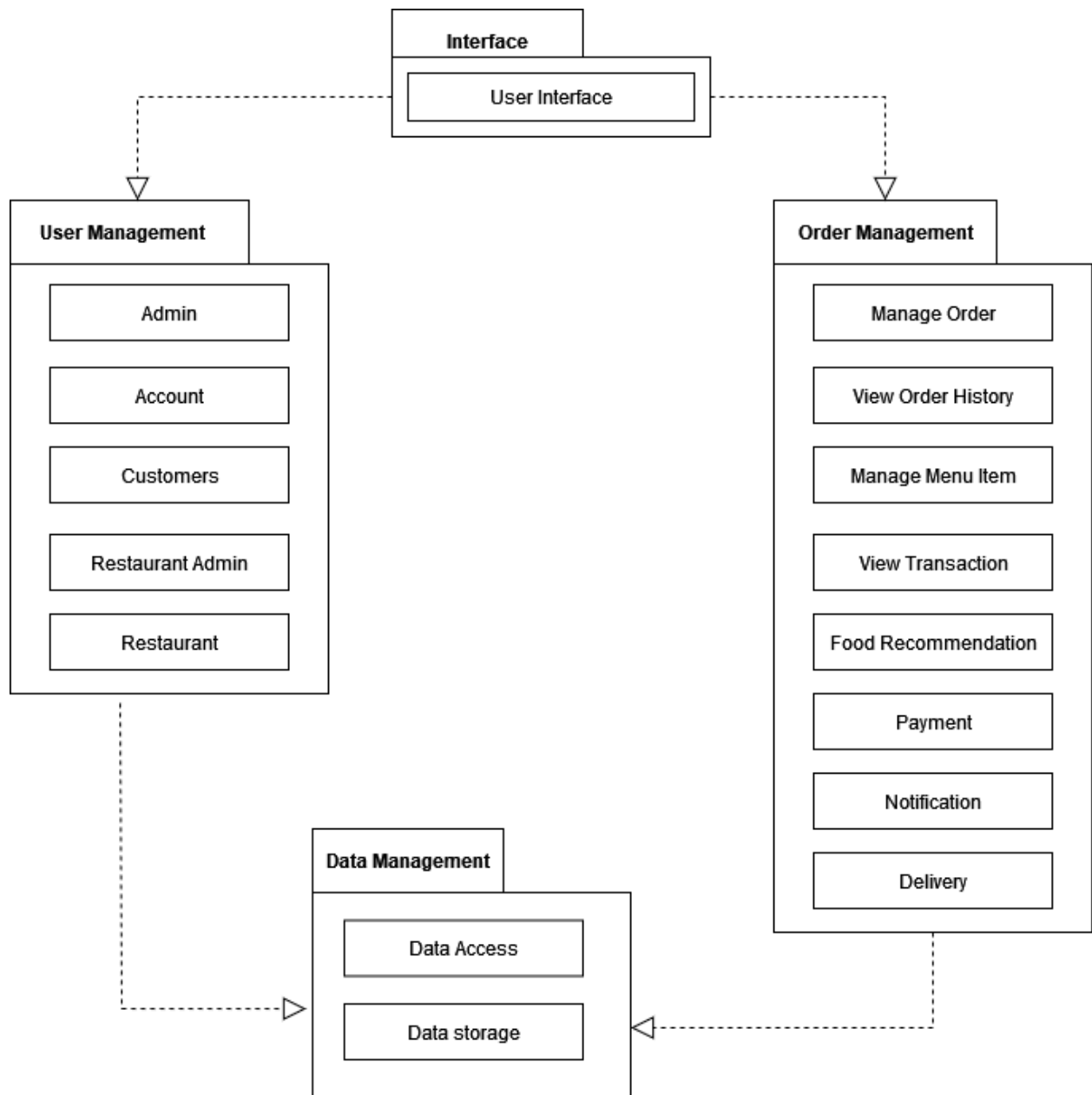


Figure 4.14 Packages dependency diagram

5. CHAPTER FIVE - IMPLEMENTATION AND TESTING

5.1 INTRODUCTION

The Implementation phase is the phase which puts the design document into code. This document consists of screenshots and tested input/output results of the implemented system.

5.2 Mapping Models to code

5.3.1 Mapping Association

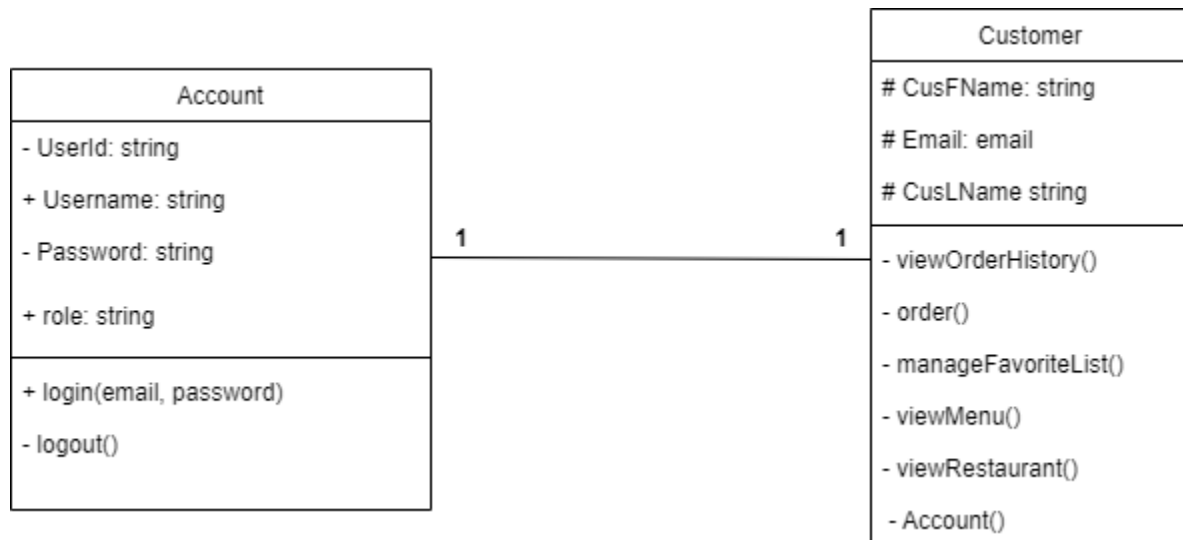


Fig. Mapping Account to customer

Account to Customer

```
import mongoose from "mongoose";
```

```
const orderSchema = new mongoose.Schema(  
  {  
    customerId: {
```

```
    type: String,
    required: true,
  },
  customer: {
    first_name: {
      type: String,
      required: true,
    },
    last_name: {
      type: String,
      required: true,
    },
    email: {
      type: String,
      required: true,
    },
  },
  products: [
    {
      name: {
        type: String,
        required: true,
      },
```

```
    quantity: {  
      type: Number,  
      required: true,  
    },  
    price: {  
      type: Number,  
      required: true,  
    },  
  },  
],  
totalPrice: {  
  type: Number,  
  required: true,  
},  
orderDate: {  
  type: Date,  
  default: Date.now,  
},  
shippingAddress: {  
  address: {  
    type: String,  
    required: true,  
  },  
}
```

```
city: {  
    type: String,  
    required: true,  
},  
  
paymentStatus: {  
    type: String,  
    enum: ["pending", "completed", "failed"],  
    default: "pending",  
},  
  
tx_ref: {  
    type: String,  
    required: true,  
},  
  
currency: {  
    type: String,  
    enum: ["ETB", "USD"],  
    default: "ETB",  
},  
  
{ timestamps: true }  
);
```

```
const Order = mongoose.model("Order", orderSchema);  
  
export default Order;
```

Abaysew, [6/21/2023 4:51 PM]

```
const handleLogin = (e) => {  
  
  e.preventDefault();  
  
  setLoading(true);  
  
  signInWithEmailAndPassword(auth, email, password)  
  
    .then((userCredential) => {  
  
      // Signed in  
  
      const user = userCredential.user;  
  
      set({ type: "LOGIN", payload: user });  
  
      navigate("/");  
  
    })  
  
    .catch((error) => {  
  
      setError(true);  
  
      setLoading(false);  
  
    });  
  
};
```

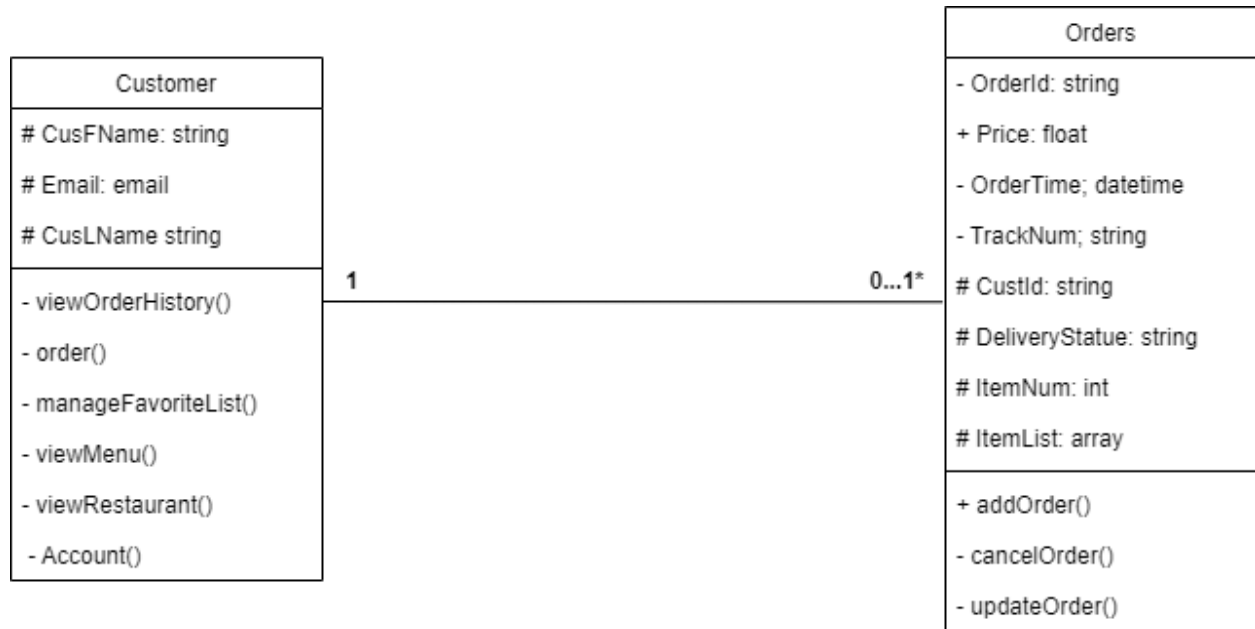


Fig. Mapping customer to order

Customer to Order Class

```

export const getOrderById = (req, res) => {

  const userId = req.params.userId;

  Order.find({ customerId: userId })

    .then((order) => res.status(200).json(order))

    .catch((err) => res.status(404).json("No order with this id"));

};

```




Fig. Mapping Customer to View order history

Customer to View order history

customer -> orders -> by userId

```

export const getOrderById = (req, res) => {
  const userId = req.params.userId;

  Order.find({ customerId: userId })

    .then((order) => res.status(200).json(order))

    .catch((err) => res.status(404).json("No order with this id"));
};
  
```

5.2 Source code of Major Classes

Restaurant

```
export default {  
  name: 'resturant',  
  title: 'Restaurants',  
  type: 'document',  
  fields: [  
    {  
      name: 'name',  
      type: 'string',  
      title: 'Name',  
      validation: rule=> rule.required(),  
    },  
    {  
      name: 'description',  
      type: 'string',  
      title: 'Description',  
      validation: rule=> rule.max(200),  
    },  
    {  
      name: 'image',  
      type: 'image',  
      title: 'image of the restaurant',
```

```
},  
  
{  
  name: 'lat',  
  type: 'number',  
  title: 'latitude of the restaurant',  
},  
  
{  
  name: 'lng',  
  type: 'number',  
  title: 'longitude of the restaurant',  
},  
  
{  
  name: 'address',  
  type: 'string',  
  title: 'Restaurant address',  
  validation: rule=> rule.required(),  
},  
  
{  
  name: 'rating',  
  type: 'number',  
  title: 'Enter a number between 1 to 5',  
  validation: rule=>rule.required().min(1).max(5).error('Please enter a value between 1 to 5')  
},
```

```
{
  name: 'reviews',
  type: 'string',
  title: 'Reviews'
},
{
  name: 'phone',
  type: 'string',
  title: 'Phone Number'
},
{
  name: 'email',
  type: 'string',
  title: 'Email'
},
{
  name: 'type',
  title: 'Category',
  validation: rule=> rule.required(),
  type: 'reference',
  to: [{type: 'category'}]
},
{
```

```

    name: 'dishes',
    type: 'array',
    title: 'Dishes',
    of: [{type: 'reference', to: [{type: 'dish'}]}]
  }
]
}

```

User Login

Account -> user

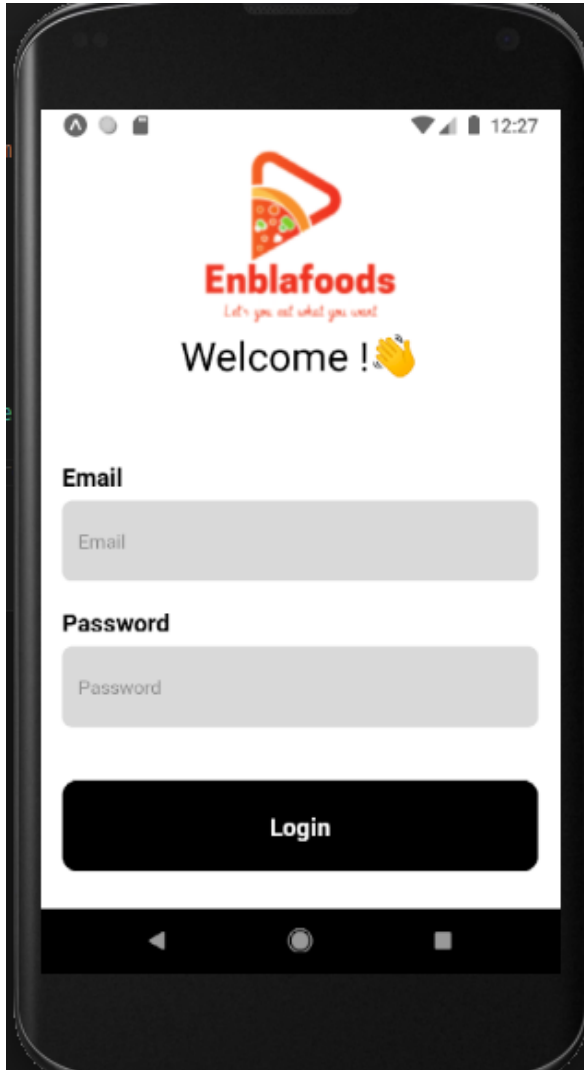
```

const handleLogin = (e) => {
  e.preventDefault();
  setLoading(true);
  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      // Signed in
      const user = userCredential.user;
      set({ type: "LOGIN", payload: user });
      navigate("/");
    })
    .catch((error) => {
      setError(true);
      setLoading(false);
    });
});

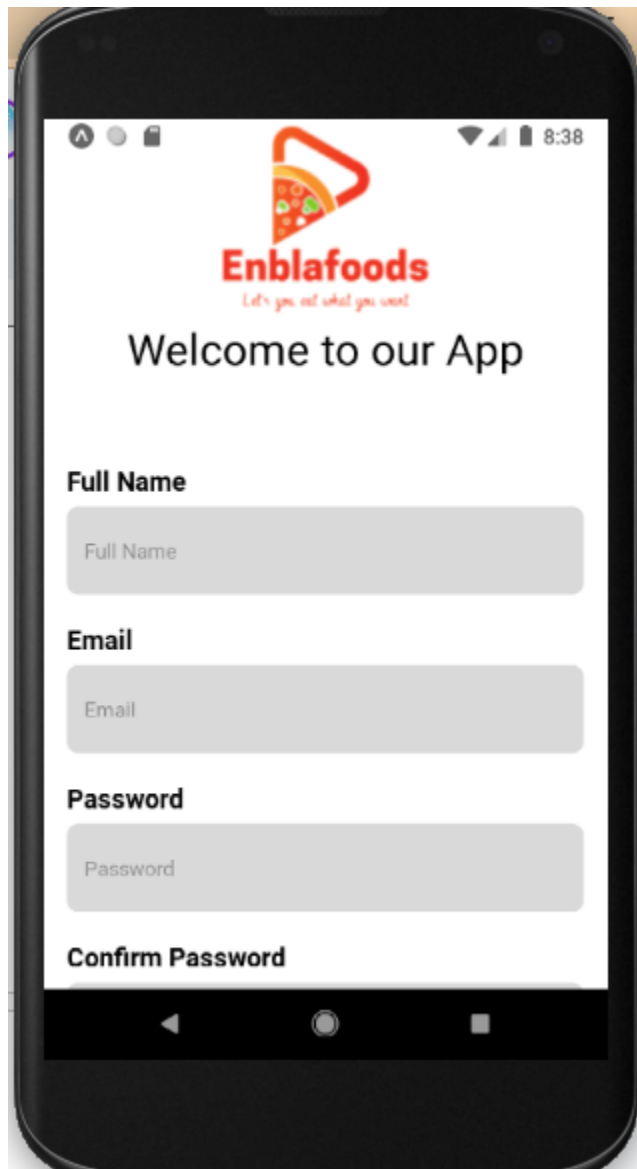
```

5.3 SCREENSHOTS


5.3.1 Login -Client Side



5.3.2 Sign up Page



5.3.3 Admin Panel



MAIN

Dashboard

LISTS

Users

Restaurants

Orders

Dishes

Categories

USEFUL

Stats

Notifications

USER

Logout

Search...

English

USERS

4

See all users

20 %

ORDERS

2

View all orders

20 %

EARNINGS

ETB 600

20 %

BALANCE

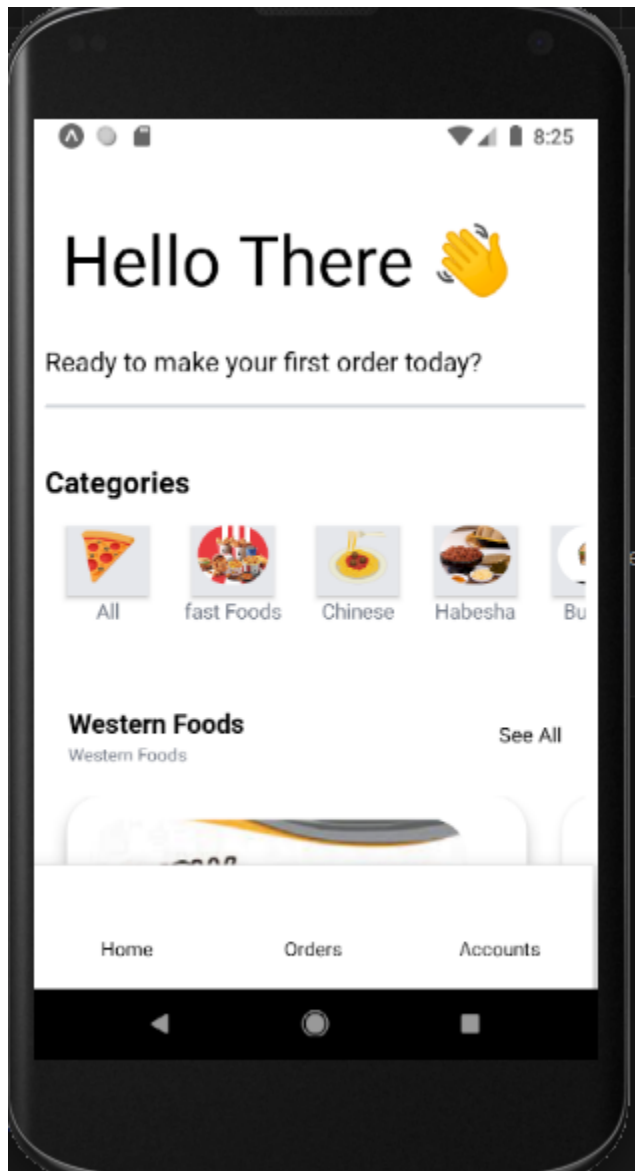
ETB 600

20 %

Latest Transactions

Order ID	User ID	Product	Customer	Date	Amount	Shipping Address	Currency	Status
648d7e552c919dca8db7bd4e	YpJZxdBZTNQ7pEci3YFGOob4N72	Shuro	Abebe Beso	4 days ago	140	Bole Rwanda house no 13	ETB	completed
648d6e2368b4058c38f561ab	32j5WgajlGMuV2WKagwSKWt5dwi2	Burger	Abeba Samson	4 days ago	460	Arat Kilo, Romina	ETB	completed

5.3.4 Home Screen



5.3.5 Order screen -Admin side



Search...

English

MAIN

Dashboard

LISTS

Users

Restaurants

Orders

Dishes

Categories

USEFUL

Stats

Notifications

USER

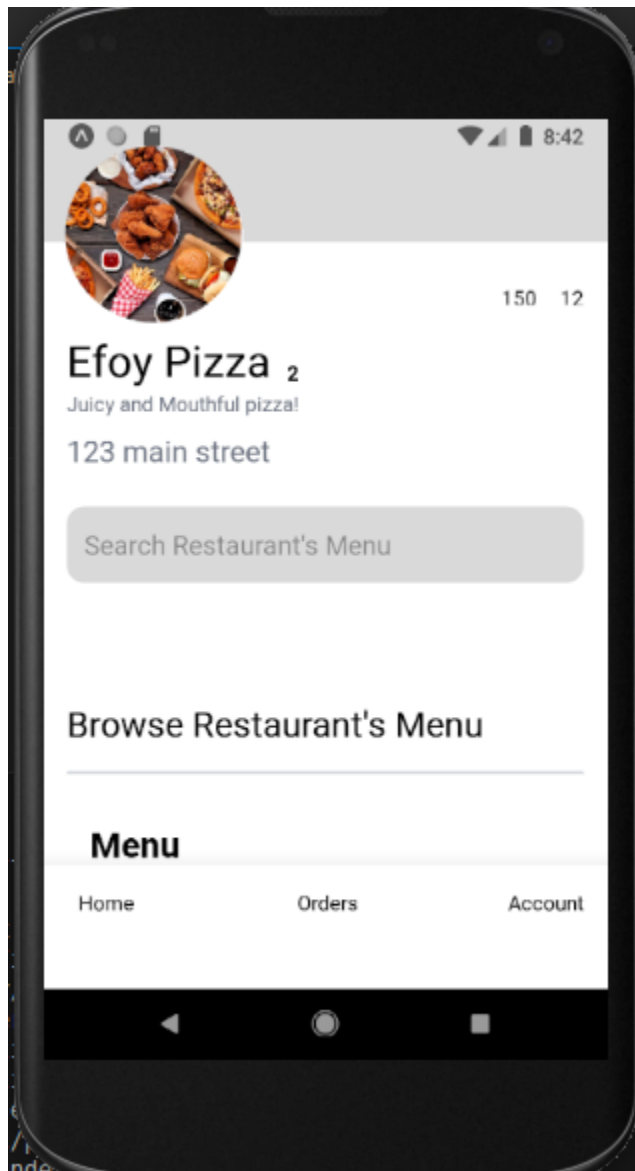
Logout



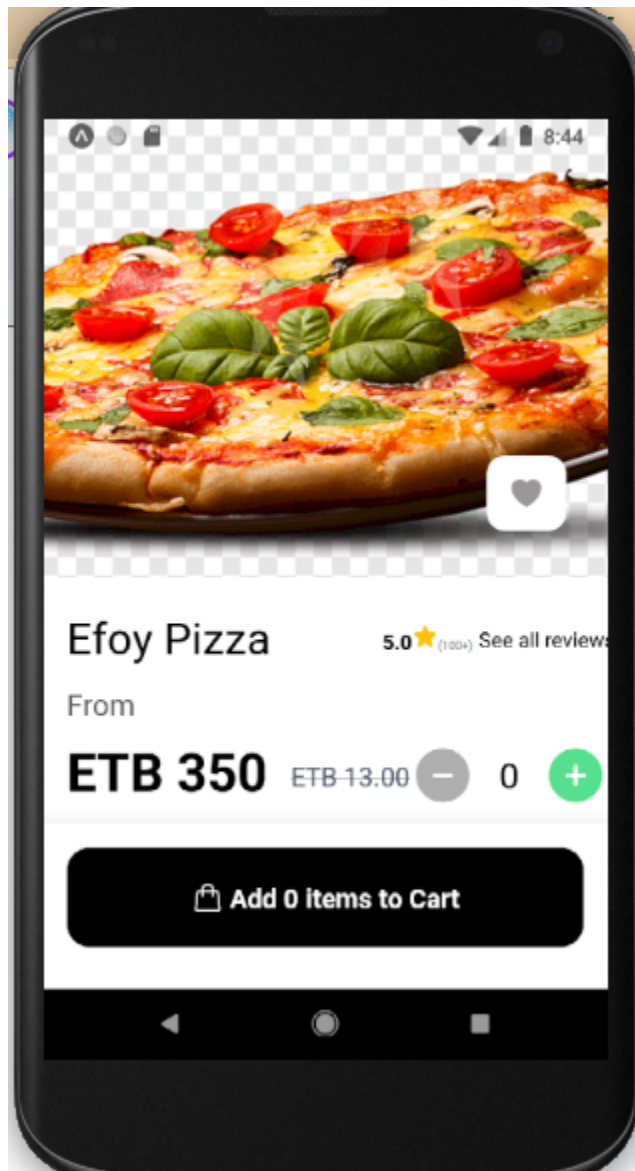
View Orders

Order ID	User ID	Product	Customer	Date	Amount	Shipping Address	Currency	Status
648d6e2368b4058c38f561ab	32j5WgajlGMuV2WKagwSKWt5dwi2	Burger	Abeba Samson	5 days ago	460	Arat Kilo, Romina	ETB	complete
648d7e552c919dca8db7bd4e	YpJZxdBZTNQ7pEci3IYFGOob4N72	Shuro	Abebe Beso	5 days ago	140	Bole Rwanda house no 13	ETB	complete
649305548fe63fad19d65b34	xHm77AbwZNOqsOWbllRzk0FNqrC3	Shuro	Abraham Yohannes	8 hours ago	140	Bole Rwanda house no 13	ETB	complete


5.3.6 Restaurant screen - Client side



5.3.7 Dish screen - Client side



5.3.8 Dish screen - Admin side



English

MAIN

- Dashboard

LISTS

- Users
- Restaurants
- Orders
- Dishes
- Categories

USEFUL

- Stats
- Notifications

USER

- Logout


Add New Dish

Add New

<input type="checkbox"/>	ID	Dish Name	Price	Description	Action
<input type="checkbox"/>	03ee455:	Efoy Pizza	350	Pizza	View Delete
<input type="checkbox"/>	082d1af:	Red Tibs	200	Delicious Tibs	View Delete
<input type="checkbox"/>	0ec4364:	Sandwich	200	Sandwich	View Delete
<input type="checkbox"/>	3d8a15b:	Pasta	230	Pasta from Wow Burger	View Delete
<input type="checkbox"/>	XUAd6Z:	Kitfo	234	Nice Kitfo	View Delete
<input type="checkbox"/>	ac40bdb:	Margherita Pizza	300	Margherita Pizza	View Delete

1-6 of 6

5.3.10 Account screen - Admin side



English

MAIN

- Dashboard

LISTS

- Users
- Restaurants
- Orders
- Dishes
- Categories


USEFUL

- Stats
- Notifications

USER

- Logout

Add New User



First Name

Abeba

Last Name

Samson

Email

abeba@gmail.com

Phone

+251 934898978

Password

Address

Arat Killo, Romina Restaurant

Send

5.3.11 Food Category - Admin side

አንብላ

MAIN

Dashboard

LISTS

Users

Restaurants

Orders

Dishes

Categories

USEFUL

Stats

Notifications

USER

Logout

Search...

English

Add New Category

Add New

<input type="checkbox"/>	ID	Category Name	Description	Action
<input type="checkbox"/>	0d1ec	All	All Foods	View Delete
<input type="checkbox"/>	6365C	fast Foods	Fast Foods	View Delete
<input type="checkbox"/>	63a41	Chinese	chinese Food	View Delete
<input type="checkbox"/>	XUAd	Habesha	Habesha foods	View Delete
<input type="checkbox"/>	bb29E	Burger	Nice Burger	View Delete

1-5 of 5

5.3.12 Restaurant List - Admin side

አንብላ

MAIN

Dashboard

LISTS

Users

Restaurants

Orders

Dishes

Categories

USEFUL

Stats

Notifications

USER

Logout

Search...

English

Add New Restaurant

Add New

<input type="checkbox"/>	ID	Restaura...	Adress	Email	Rating	Phone	Action
<input type="checkbox"/>	40a07	Wow Burger	Addis Ababa, Ethiop	hegde@gmail.com	5	251949327898	View Delete
<input type="checkbox"/>	47ba1	Efoy Pizza	123 main Street	harry@den.com	2	912345678	View Delete

1-2 of 2

5.4 TESTING

Testing is the process of analyzing a system or its components to detect the differences between specified and existing behavior. It is a systematic attempt to find errors in a planned way in the implemented system [1]. This document finds the differences by evaluating the system components.

5.4.1 General Testing Strategy

In the testing phase of this document we have used black box testing method which conducts testing on the specified functions, if they are fully operational and/or if they have errors. We have used the following testing techniques to test our system.

5.4.2 System Testing

Requirements testing: this test was conducted to inspect the use case model and identify use case instances that are likely to cause failures and it is used to find the difference between the implemented system and the functional requirements.

Valid Email and Password

5.4.3 Usability Testing

Usability testing tries to find faults in the user interface design of the system [1]. We performed the test by random users to check if there are faults in users are confused by the user interface design of the system and if the system has accomplished its intended purpose and to see if the users will accept the system as a new way to report a crime incident

5.5 Test Case

Some selected test cases for the functional requirements are as follows:

Test Scenario 1: User Login (Admin, customer, Restaurant)

Test case 1.1 Enter Valid Email and password

Implementation

Tabel 5-1: Test Case Enter Valid Email and Password

Feature to be tested	Test if login is successful
Preconditions	User must be registered first
Input test data	Correct Email and password must be inserted
Steps to be executed	<ol style="list-style-type: none">1. Insert valid Email and password in the given field2. Press Submit
Expected Result	Successful login of user to the system
Actual Result	User logged into the system successfully
Pass/Fail	Pass

→ Test case 1.2 Enter Invalid name and password

Implementation

Table 5-2: Test Case Enter Invalid name and password

Feature to be tested	Test if login is successful
Preconditions	User must be registered first
Input test data	Invalid Email and password is inserted
Steps to be executed	<div>1. Insert invalid Email and password in the given field</div> <div>2. Press Submit</div>

Expected Result	User failed to log into the system
Actual Result	Error message is displayed and prompt user to insert valid Email and password
Pass/Fail	Pass

Test Scenario 2: Manage Order(customer)

→ Test case 2.1 if the customer have order

Implementation

Tabel 5-3: Test Case If The Customer Have Order

Feature to be tested	Manage order
Preconditions	User must be login first
Test data	Customer have order
Steps to be executed	1.Go to your order

	2. Press manage order
Expected Result	User can update his/her order
Actual Result	Your order will be updated
Pass/Fail	Pass

4.1.8 Orders Class Diagram This class is responsible for allowing customers to place orders, cancel orders, and update orders.

→ Test case 2.2 if the customer have no order

Implementation

Tabel 5-4: Test Case If The Customer Have No Order

Feature to be tested	Manage order
Preconditions	User must be login first
Test data	Customer does not have any order

Steps to be executed	<ol style="list-style-type: none"> 1. Go to your order option 2. Press order
Expected Result	User can update his/her order
Actual Result	A prompt saying the user hasn't placed an order yet is displayed.
Pass/Fail	Pass

Test Scenario 3: Payment (Customer)

→ Test case 3.1 if the customer has orders in cart

Implementation

Tabel 5-5: Test Case If The Customer Has Order In Cart

Feature to be tested	Payment
Preconditions	User must be logged in and has orders selected
Test data	Customer has orders in his cart

Steps to be executed	1. Press add to cart 2. Press payment button
Expected Result	User views orders and correct total price
Actual Result	Selected orders list shown and the correct total price shown
Pass/Fail	Pass

→ Test case 3.2 if the customer has no orders in cart

Implementation

Tabel 5-6: Test Case If The Customer Has No Orders In Cart

Feature to be tested	Payment
Preconditions	User must be logged in and has orders selected
test data	Customer has orders in his cart

Steps to be executed	<ol style="list-style-type: none"> 1. Go to your cart 2. Press cart
Expected Result	User is informed that the cart is empty.
Actual Result	An empty page with “Cart is currently empty” message is shown.
Pass/Fail	Pass

Test Scenario 4: View Order History (Customer)

Test case 4.1 When there is previous orders

Implementation

Tabel 5-7: Test Case When There Is Previous Orders

Feature to be tested	View Order History
Preconditions	User must be logged in

Test data	Customer must have ordered before
Steps to be executed	<ol style="list-style-type: none"> 1. Go to your order history 2. Press order history
Expected Result	Customer is shown their previous orders
Actual Result	A list of customer's previous orders is shown.
Pass/Fail	Pass

Test case 4.2 When there is no previous orders

Implementation

Tabel 5-8: Test Case When There Is No Previous Orders

Feature to be tested	View Order History
Preconditions	User must be logged in

Test data	Customer must have not ordered before
Steps to be executed	<ol style="list-style-type: none"> 1. Go to your order history 2. Press order history
Expected Result	Customer informed that there was no order before.
Actual Result	An empty page with “Order has not been made yet.” message is shown.
Pass/Fail	Pass

Testing Materials

- Facilities required: The system works on any electronic devices that can be connected to an Internet connection.
- Hardware Required: The testing is conducted on a computer that has an Internet connection.
- Software required: The computer should have a web browser that works properly in order to conduct the testing

5.6 Conclusion & Recommendations

The documentation for a Enblafoods delivery application has been carefully prepared to ensure that the application meets the needs of its users and is developed in a way that is efficient and effective. The documentation has been written in a clear and concise manner. The documentation

is also flexible enough to allow for changes to the requirements of the application as it is developed.

Recommendations

Based on the documentation, the following recommendations are made for future development of the food delivery application:

- The application should be developed using a scalable architecture that can handle a large number of users.
- The application should be secured against unauthorized access.
- The application should be tested thoroughly to ensure that it meets all of the requirements.
- The application should be monitored to ensure that it is performing as expected.
- The application should be recommend food based on users preference and order history to ensure that it is performing as expected
- The application should provide real-time tracking of orders so that users know when their food will arrive
- The application should provide Additional payment gateways for flexible payment options

By following these recommendations, the food delivery application can be developed into a successful product that meets the needs of its users.

Reference

- [1].Bruegge, B. and Dutoit, A.H., 2009. Object--Oriented Software Engineering. Using UML, Patterns, and Java. *Learning*, 5(6), p.7.
- [2].<https://chat.openai.com>
- [3].<https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture-4179579423a9>
- [4].[Beck & Andres, 2005] K. Beck & C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed., Addison-Wesley, Reading, MA, 2005.
- [5].[Booch et al., 2005] G. Booch, J. Rumbaugh, & I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA, 2005.
- [6].[Elssamadisy, 2009] A. Elssamadisy, *Agile Adoption Patterns*, Addison-Wesley, Reading, MA, 2009.
- [7]. Paulk and colleagues (1995), the CMM-based assessment approach
- [8]. <https://monocept.medium.com/functional-and-non-functional-requirement-gathering-f122f810e522>
- [9].[Erl, 2005] T. Erl, *Service-Oriented Architectures: Concepts, Technology, and Design*, Prentice Hall, Upper Saddle River, NJ, 2005.
- [11]. <https://www.lucidchart.com/pages/uml-use-case-diagram>
- [12].<https://www.slideshare.net/kalsoomhAsgher/online-food-ordering-app-documentation>
- [13].https://www.researchgate.net/publication/352353954_An_empirical_study_of_online_food_delivery_services_from_applications_perspective