SOFTWARE DESIGN SPECIFICATION

KINGO COFFEE

KINGOFFEE

Introduction to Software Engineering_SWE3002_41

TEAM 10
TEAM LEADER | JISU RYOU
TEAM MEMBER | CHANHO KIM
TEAM MEMBER | KANGIN PARK
TEAM MEMBER | YELIM SO
TEAM MEMBER | JUNSUNG LEE
TEAM MEMBER | SUYOUNG BAE

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

1. PREFACE

1.1 READERSHIP

This Software Design SPECIFICATION (SDS) is divided into 10 sections with various subsections. The structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 10 is the main reader. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers.

1.2 SCOPE

This describes the software architecture and software design decisions for the implementation of KINGO COFFEE application.

1.3 OBJECTIVE

The purpose of the software design specification document is to document and track the necessary information required to effectively define architecture and system design to give the development team guidance on architecture of the system to be developed. The Software design specification document is created during the planning phase of the project. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholders whose input/approval into the UI is needed.

1.4 DOCUMENT STRUCTURE

1) Preface: this chapter describes readership, Scope, Objective, and structure of this document.
2) Introduction: this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.
3) Overall System Architecture: this chapter describes the overall architecture of the system using context diagram, sequence diagram, and use case diagram.
4) System Architecture – Frontend: this chapter describes architecture of the frontend system using class diagram and sequence diagram.

5) System Architecture – Backend: this chapter describes architecture of the backend system using class diagram and sequence diagram.
6) Protocol Design: this chapter describes design of several protocols which are used for communication of client and server.
7) Database Design: this chapter describes database design using several ER diagrams and SQL DDL.
8) Testing Plan: this chapter describes the testing plan for our system.
9) Development Plan: this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
10) Supporting Information: this chapter describes baseline of this document.

2. INTRODUCTION

2.1 OBJECTIVES

In this chapter, we describe the various tools and diagrams which we have applied to this project in the design phase.

2.2 APPLIED DIAGRAMS

2.2.1 UML

Unified Modeling Language (UML) is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is an especially important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

2.2.2 USE CASE DIAGRAM

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow.

2.2.3 Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent and when.

2.2.4 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

2.2.5 Context Diagram

Context diagram is the highest level in a Data Flow Diagram. It is a tool popular among Business Analysts who use it to understand the details and boundaries of the system to be designed in a project. It points out the flow of information between the system and external components.

2.2.6 Entity Relationship Diagram

An entity–relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

2.3 Applied Tools

2.3.1 Draw.io

draw.io is a tool to create diagrams. It can visualize information faster than other tools and can work with teams simultaneously.

2.3.2 Microsoft Powerpoint

MS Powerpoint users can create a diagram or flowchart in a presentation using the SmartArt feature.

2.4 Project Scope

This describes the software architecture and software design decisions for the implementation of KINGO COFFEE application.

2.5 References

- Team1, 2020 Spring. Software Design Document, SKKU
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
- https://whatis.techtarget.com/definition/use-case-diagram
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- https://www.edrawmax.com/context-diagram/
- https://www.techopedia.com/definition/1200/entity-relationship-diagram-erd

3. System Architecture – Overall

3.1 Objectives

In this chapter, we describe and show the organization of the system, ranging from the frontend design to the backend design of the application for the project.

3.2 System Organization

3.2.1 Context Diagram

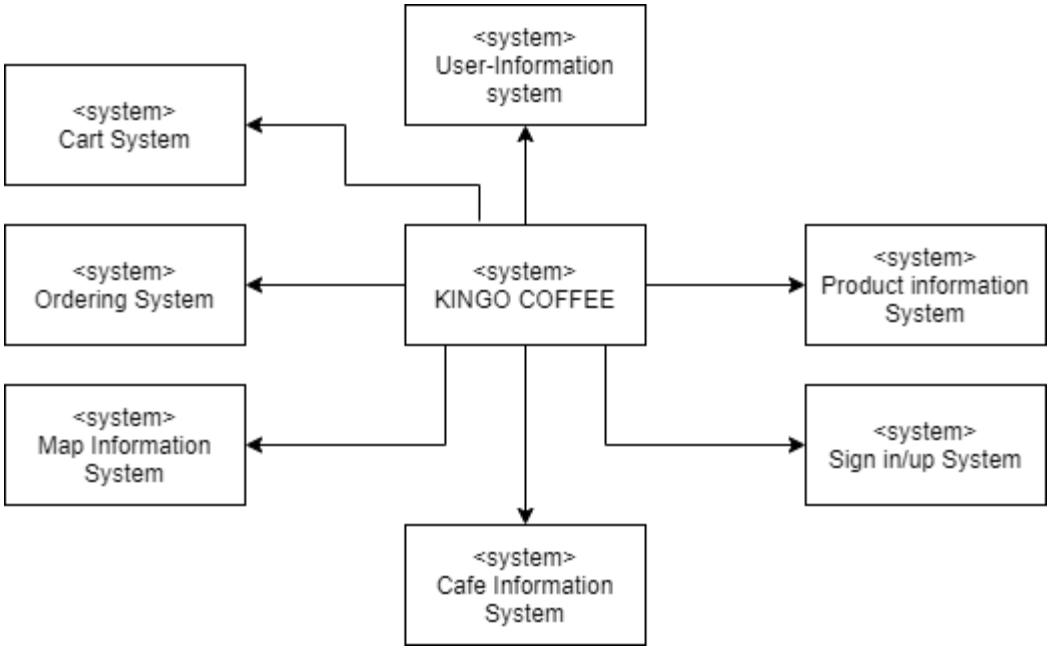

FIGURE 1 KINGO COFFEE CONTEXT DIAGRAM

3.2.2 SEQUENCE DIAGRAM

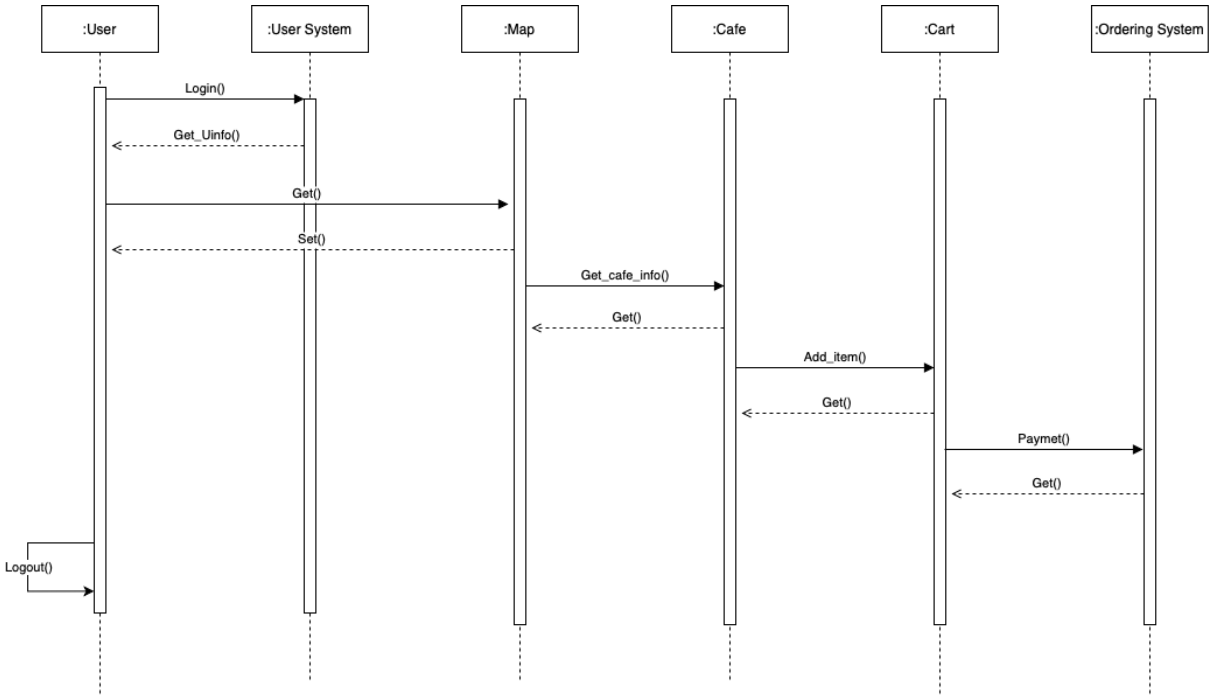

FIGURE 2 KINGO COFFEE SEQUENCE DIAGRAM

3.2.3 USE CASE DIAGRAM

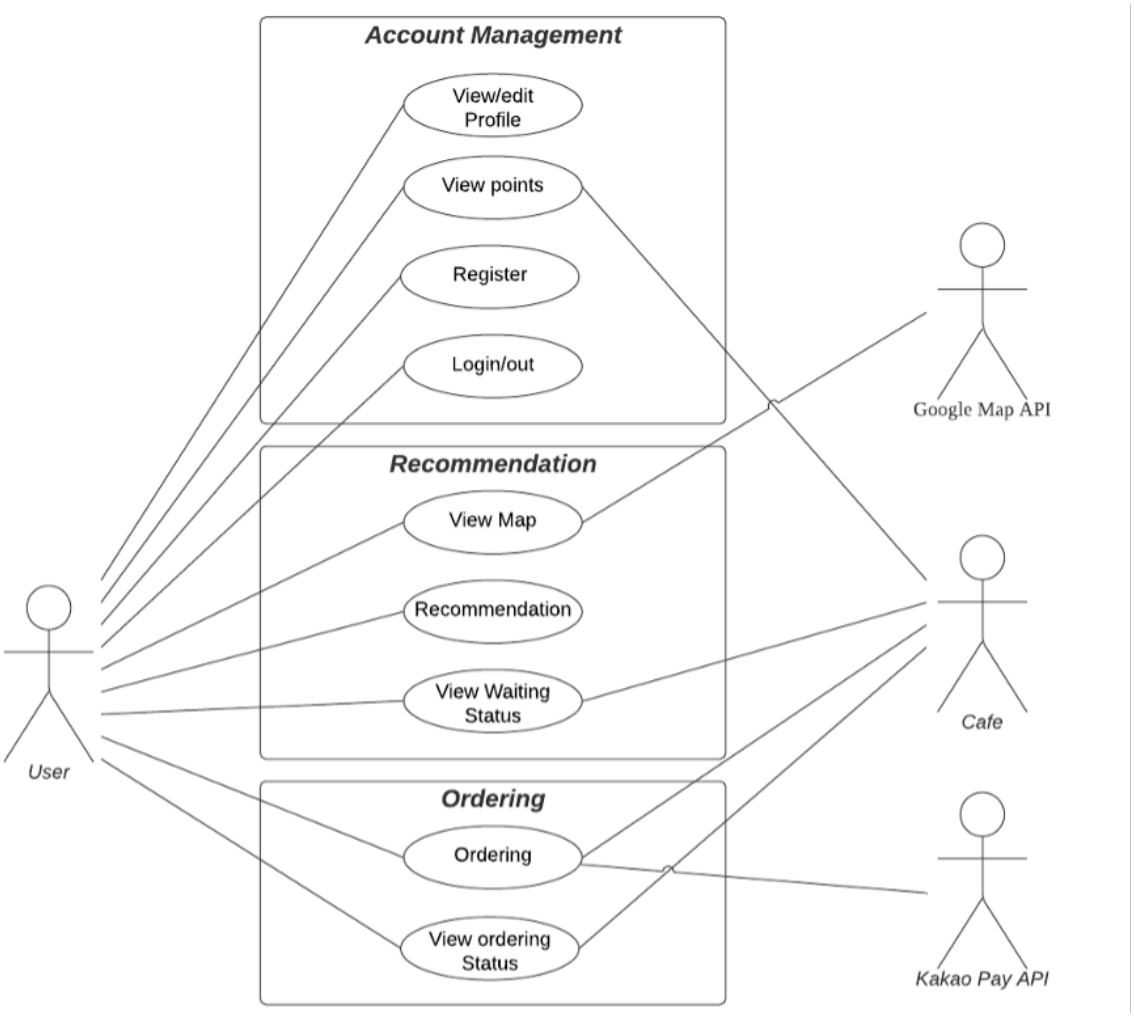

FIGURE 3 KINGO COFFEE USE CASE DIAGRAM

4. SYSTEM ARCHITECTURE – FRONTEND

4.1 OBJECTIVES

This chapter describes the structure, attributes and function of the frontend system and describes the relation of each component.

4.2 SUBCOMPONENTS

4.2.1 AUTHENTICATION

4.2.1.1 Attribute

- UserEmail : Email required to identify User Authentication.
- UserRole : Data on User's authority.
- UserPassword : Password required to identify User Authentication

- Username : User name to be used by user
- CampusInfo: Information about Campus to which User belongs

4.2.1.2 Methods

- Register(User) : Register New User
- LogIn(UserEmail, UserPassword) : User log in
- LogOut(UserId) : User log out

4.2.1.3 Class Diagram

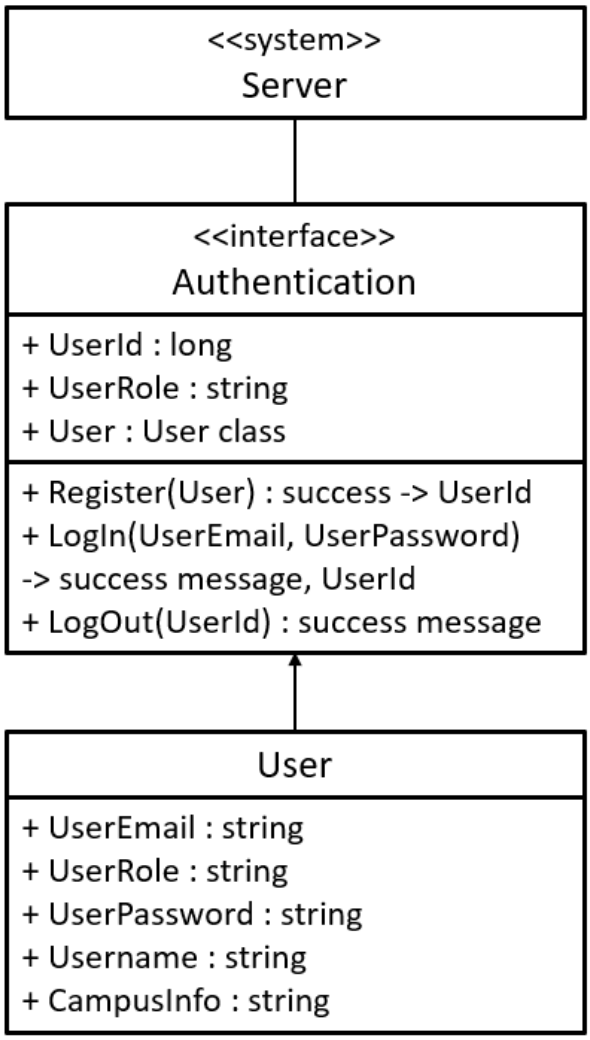

FIGURE 4 AUTHENTICATION CLASS DIAGRAM

4.2.1.4 Sequence Diagram

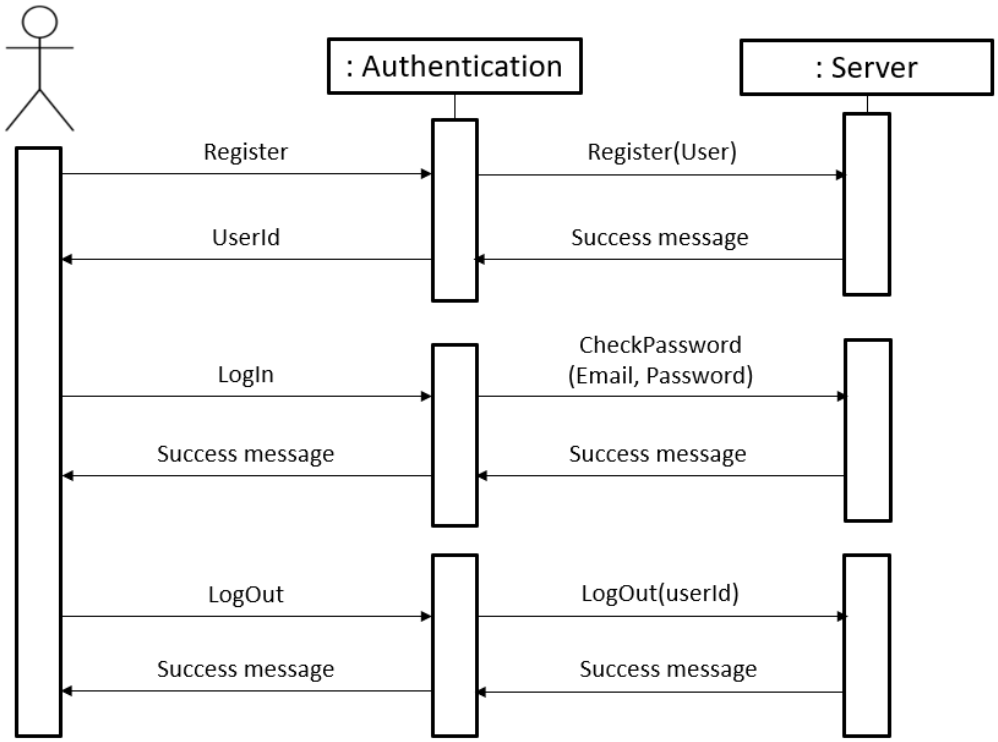

FIGURE 5 AUTHENTICATION SEQUENCE DIAGRAM

4.2.2 PROFILE

It is an interface that allows users to read and modify information about registered users.

4.2.2.1 Attribute

- UserId : Id required for identifying User's information

4.2.2.2 Methods

- GetProfile(UserId) : A method for obtaining information from a nearby Cafe
- SetProfile(UserId, User) : A method that provides Map View based on information in the Cafe

4.2.2.3 Class Diagram

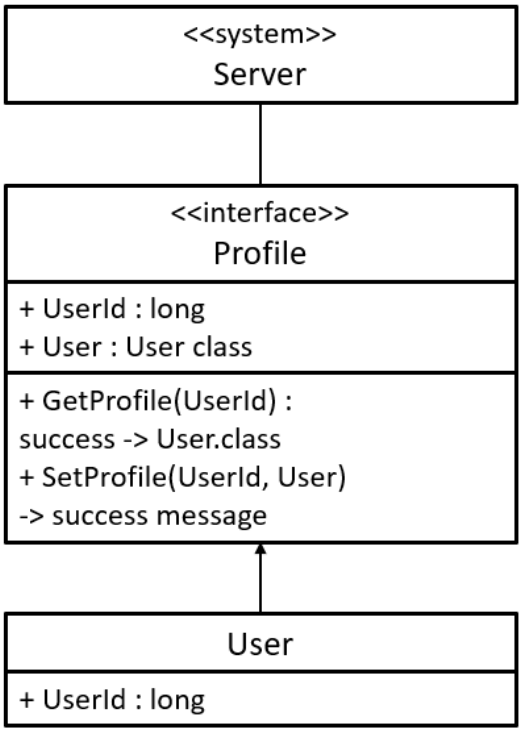

FIGURE 6 PROFILE CLASS DIAGRAM

4.2.2.4 Sequence Diagram

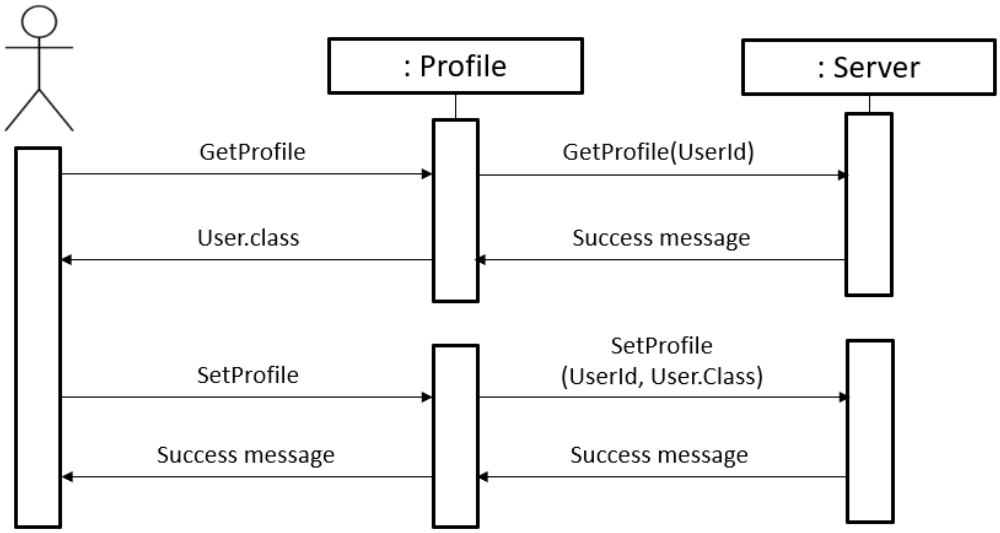

FIGURE 7 PROFILE SEQUENCE DIAGRAM

4.2.3 MAP VIEW

MapView is an interface that receives status information from nearby cafes through User's Location. We also show the Map based on that information.

4.2.3.1 Attribute

- UserId : Identifying User's information
- UserLocation : Data indicating the location information of the User
- CafeStatus : Data on the status of nearby Cafe.

4.2.3.2 Methods

- GetCafeInfo(UserId, UserLocation) : Method for obtaining information from nearby cafes
- ShowMap(List<Cafe> status, UserLocation) : A method for providing Map View based on information from a cafe
- SelectCafe(Cafe) : A method for checking the detailed order status and menu for Cafe.

4.2.3.3 Class Diagram

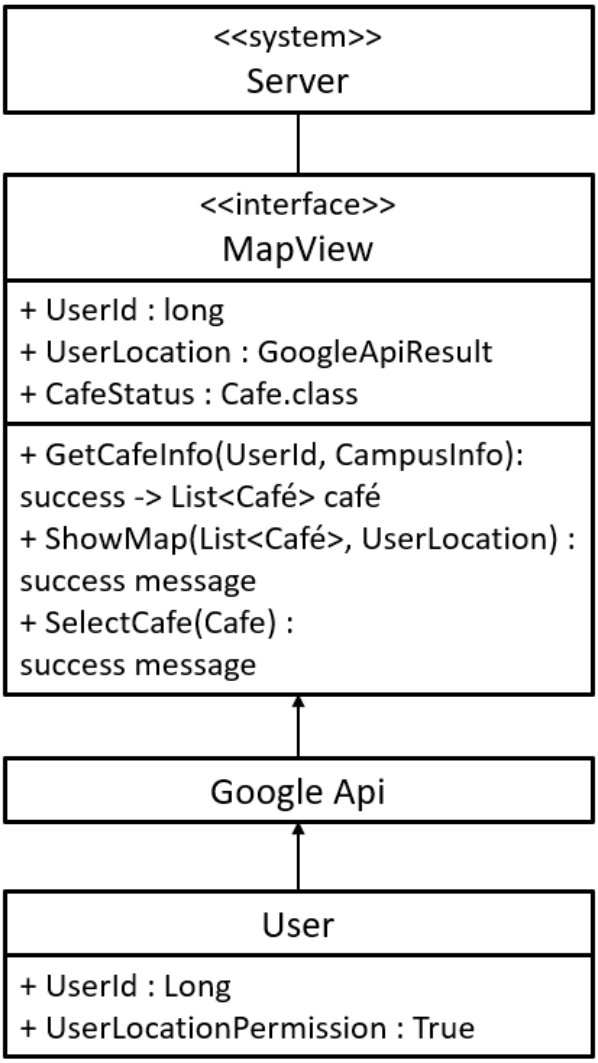

FIGURE 8 MAP VIEW CLASS DIAGRAM

4.2.3.4 Sequence Diagram

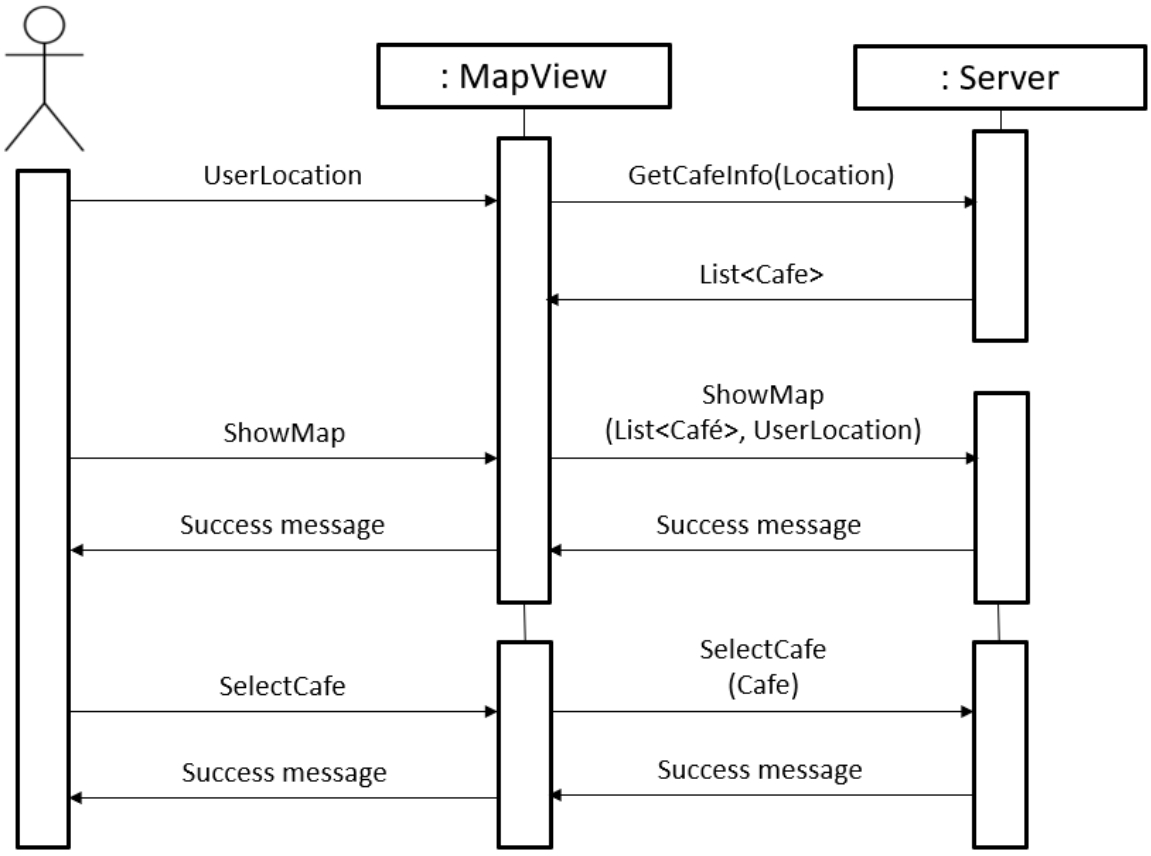

FIGURE 9 MAP VIEW SEQUENCE DIAGRAM

4.2.4 CAFE

Interface for managing Cafe, which is commonly used by apps

4.2.4.1 Attribute

- CampusInfo : Information about the Campus to which it belongs
- Menu : Information about the Menu
- CafePhoneNumber: Phone number of Cafe
- DetailLocation : Provide accurate location of Cafe
- User : Owner's User info

4.2.4.2 Methods

- RegisterCafe(Cafe, UserRole) : Registering Cafe's information
- GetOrderList(CafeId, UserRole) : Read Order list of Cafe
- UpdateCafeStatus(CafeId, UserRole, OrderId) : Update Cafe's Order Status

- DeleteOrder(CafeId, UserRole, OrderId): If an order from Cafe has been processed, delete that order

4.2.4.3 Class Diagram

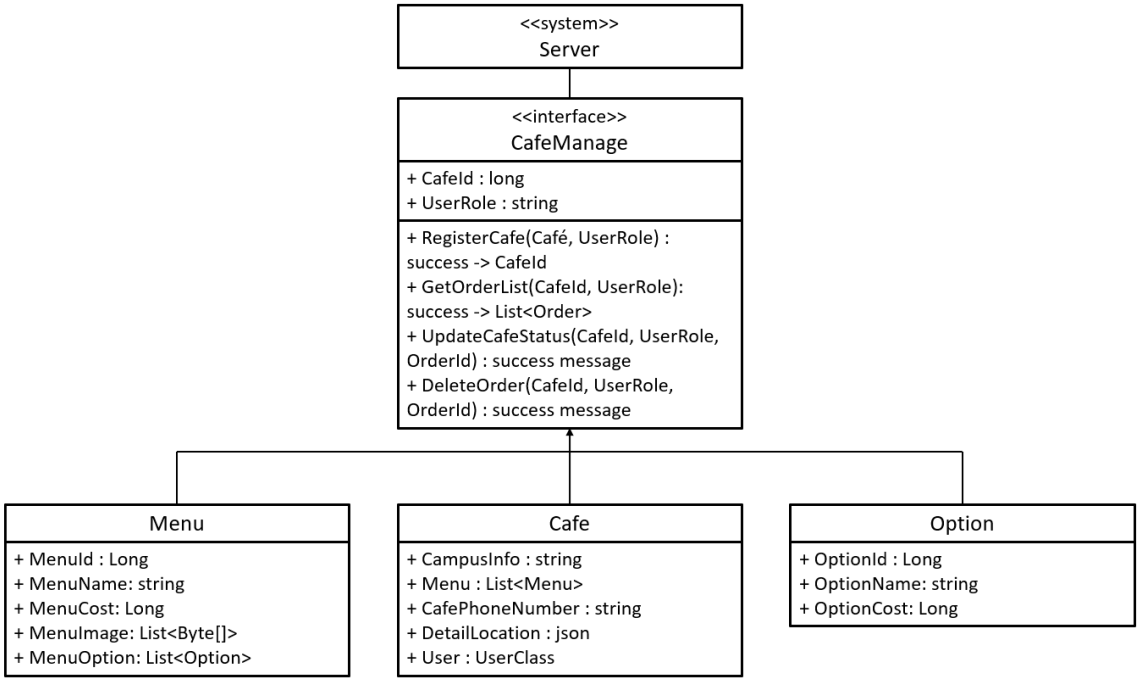

FIGURE 10 CAFE CLASS DIAGRAM

4.2.4.4 Sequence Diagram

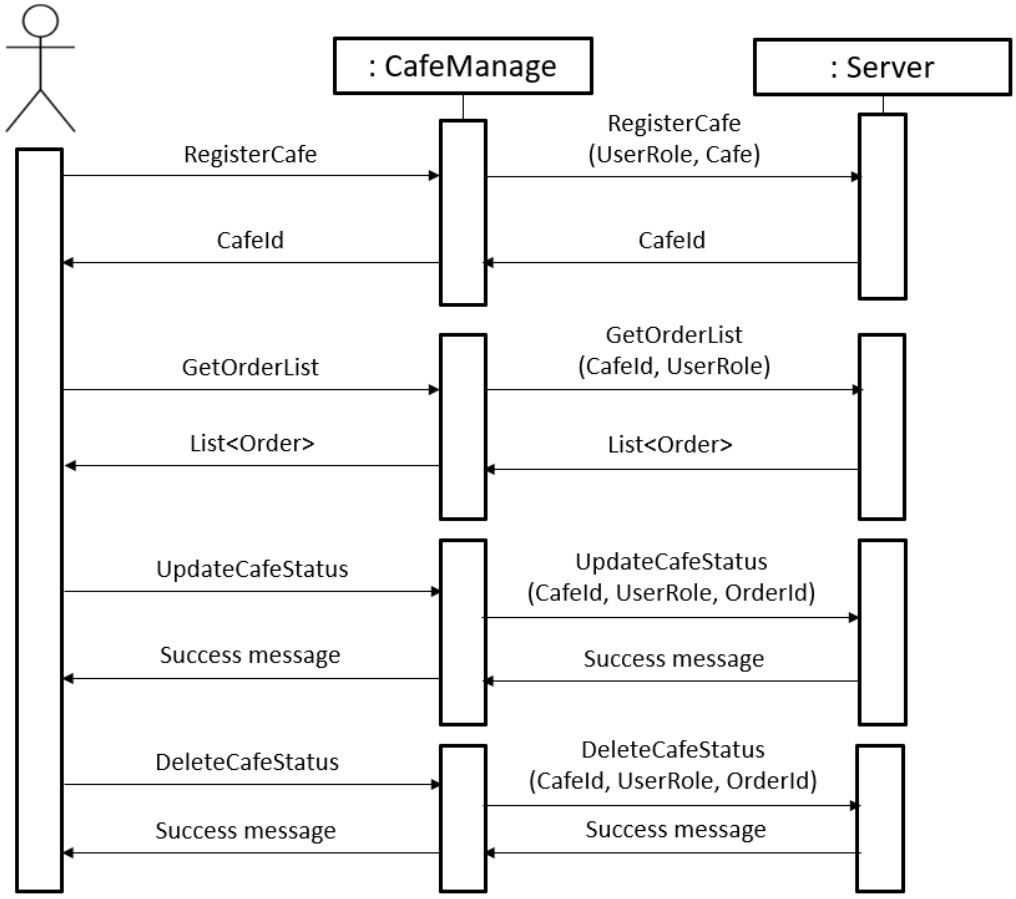

FIGURE 11 CAFE SEQUENCE DIAGRAM

4.2.5 ORDER

The interface that manages the order used in the store

4.2.5.1 Attribute

- UserId : required to identify the User of the order
- CafeId : Attributes necessary for identifying Cafe in the relevant order
- MenuId : required to identify Menu for the order
- OptionId : required for additional Option identification
- OrderTime : required for identify order time

4.2.5.2 Methods

- MakeOrder(UserId, Order) : Create Order
- GetOrderInfo(UserId) : Methods to read the cucrrent User's order status
- DeleteOrder(UserId, OrderId) : Cancel this order

4.2.5.3 Class Diagram

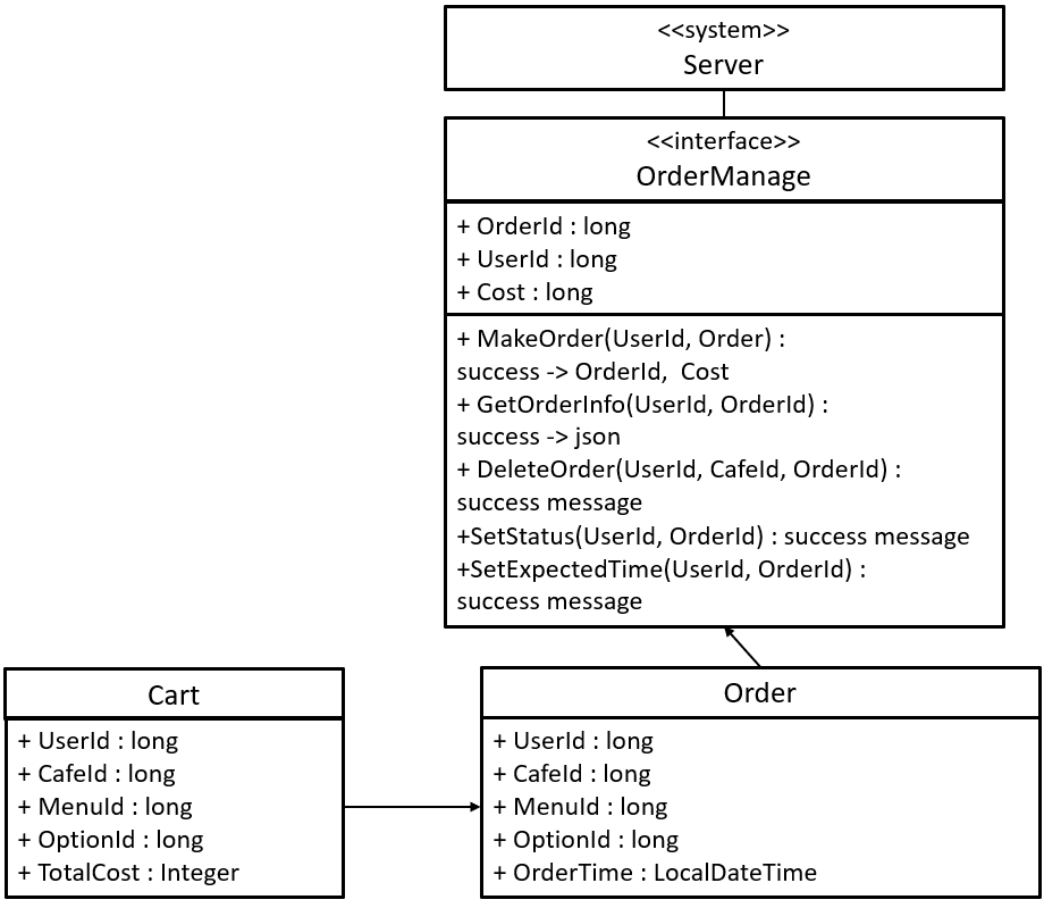

FIGURE 12 ORDER CLASS DIAGRAM

4.2.5.4 Sequence Diagram

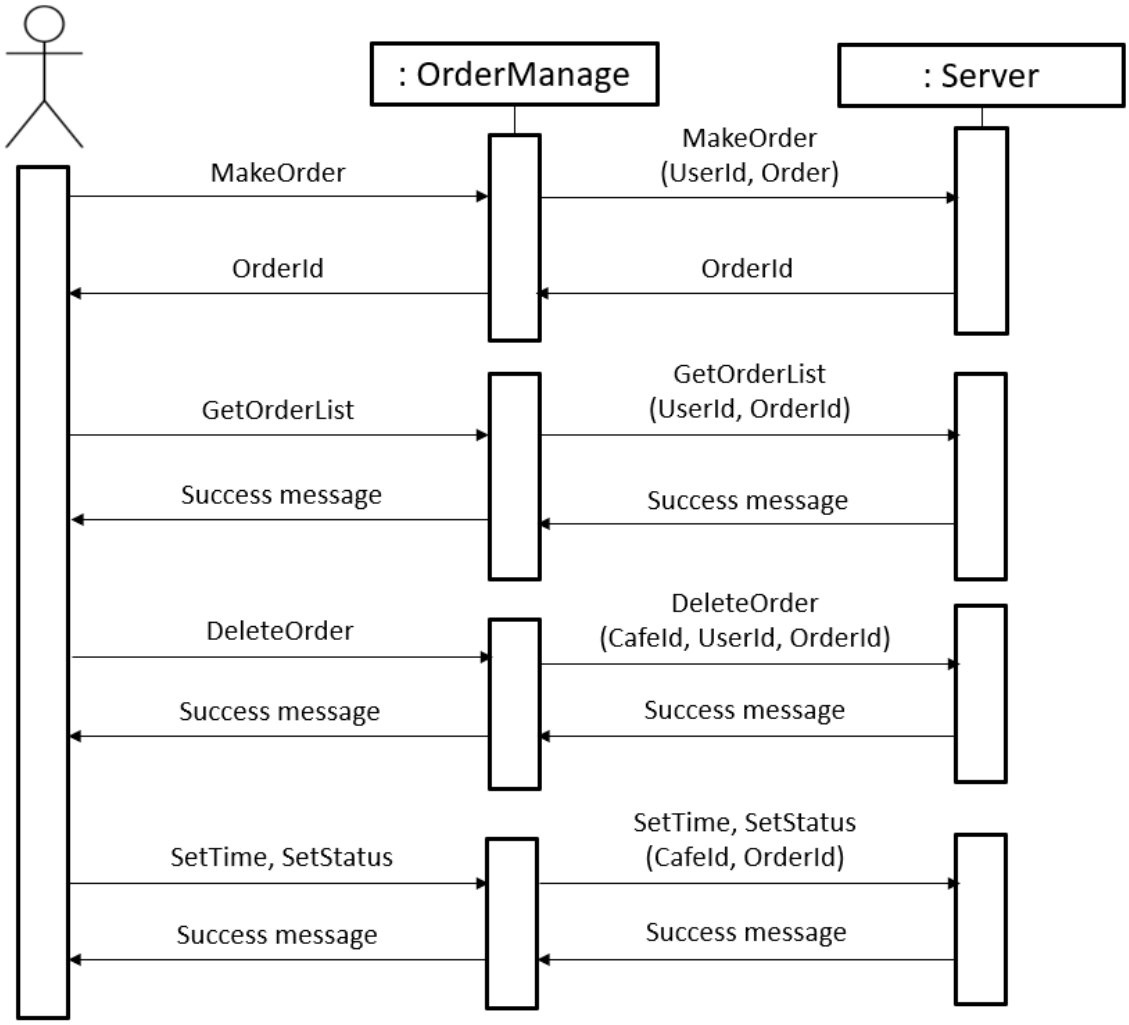

FIGURE 13 ORDER SEQUENCE DIAGRAM

4.2.6 PAYMENT

interface that uses api to process payments

4.2.6.1 Attribute

- UserId : id for identifying the User of the order
- CafeId : id for identifying Cafe of the order
- OrderId : id for identifying the order

4.2.6.2 Methods

- GetOrderCost(UserId, OrderId) : Methods to determine the cost of the order
- Payment(UserId, OrderId) : pay using api.

4.2.6.3 Class Diagram

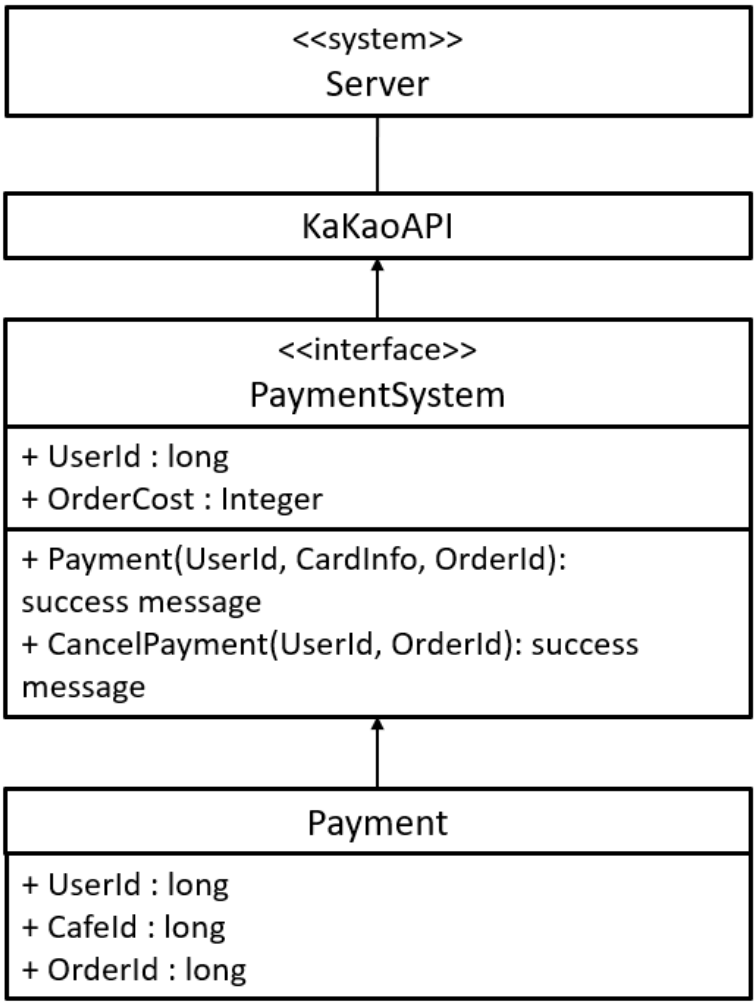

FIGURE 14 PAYMENT CLASS DIAGRAM

4.2.6.4 Sequence Diagram

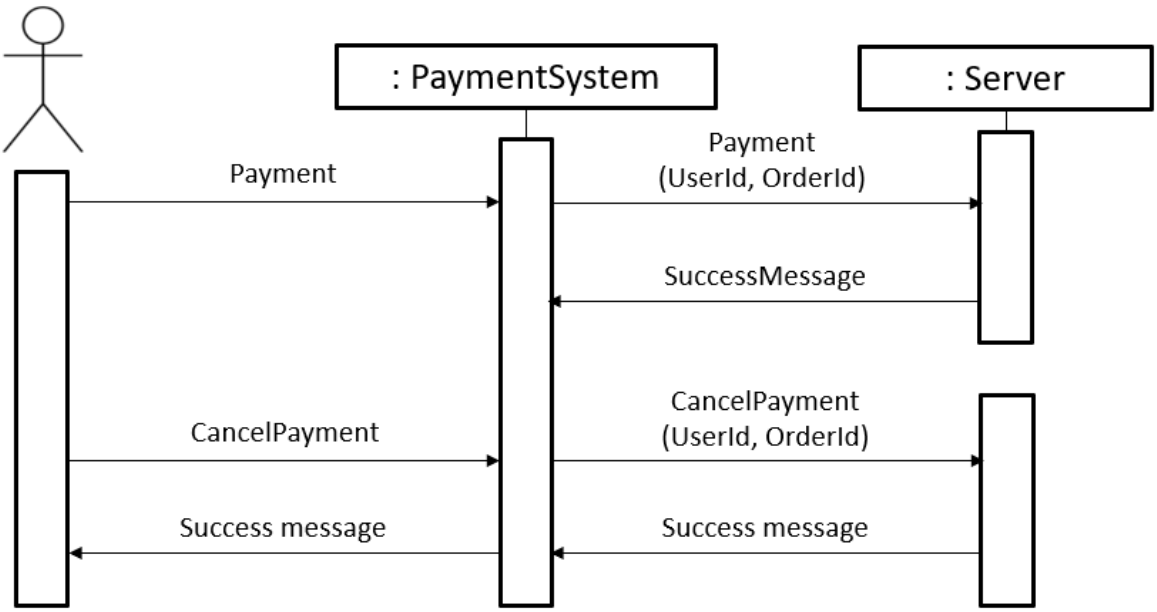

FIGURE 15 PAYMENT SEQUENCE DIAGRAM

5. SYSTEM ARCHITECTURE – BACKEND

5.1 OBJECTIVES

This chapter describes the structure of the backend systems and subsystems.

5.2 OVERALL ARCHITECTURE

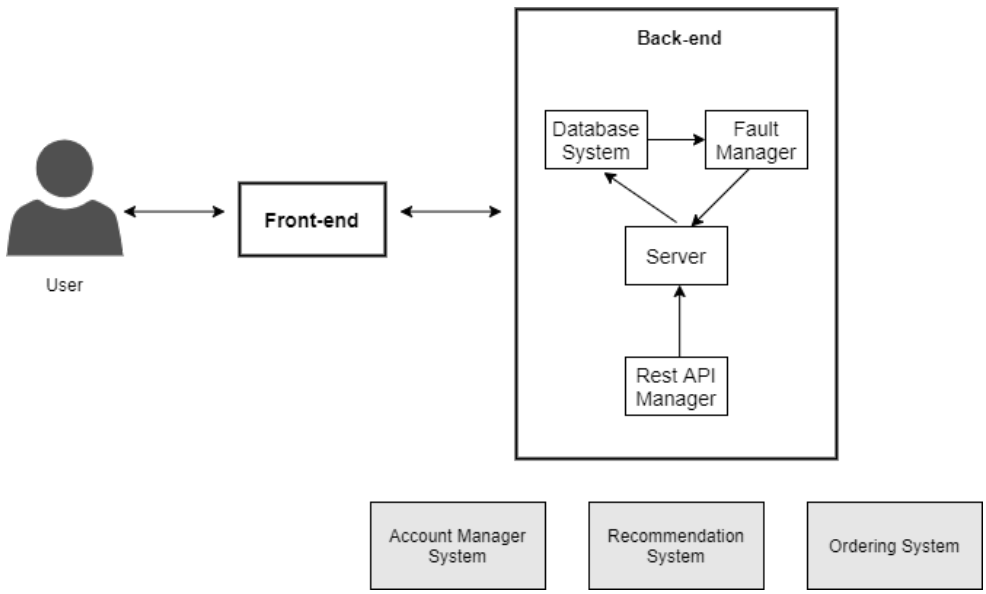

FIGURE 16 ACCOUNT MANAGEMENT CLASS DIAGRAM

5.3 SUBCOMPONENTS

5.3.1 ACCOUNT MANAGEMENT SYSTEM

5.3.1.1 Class Diagram

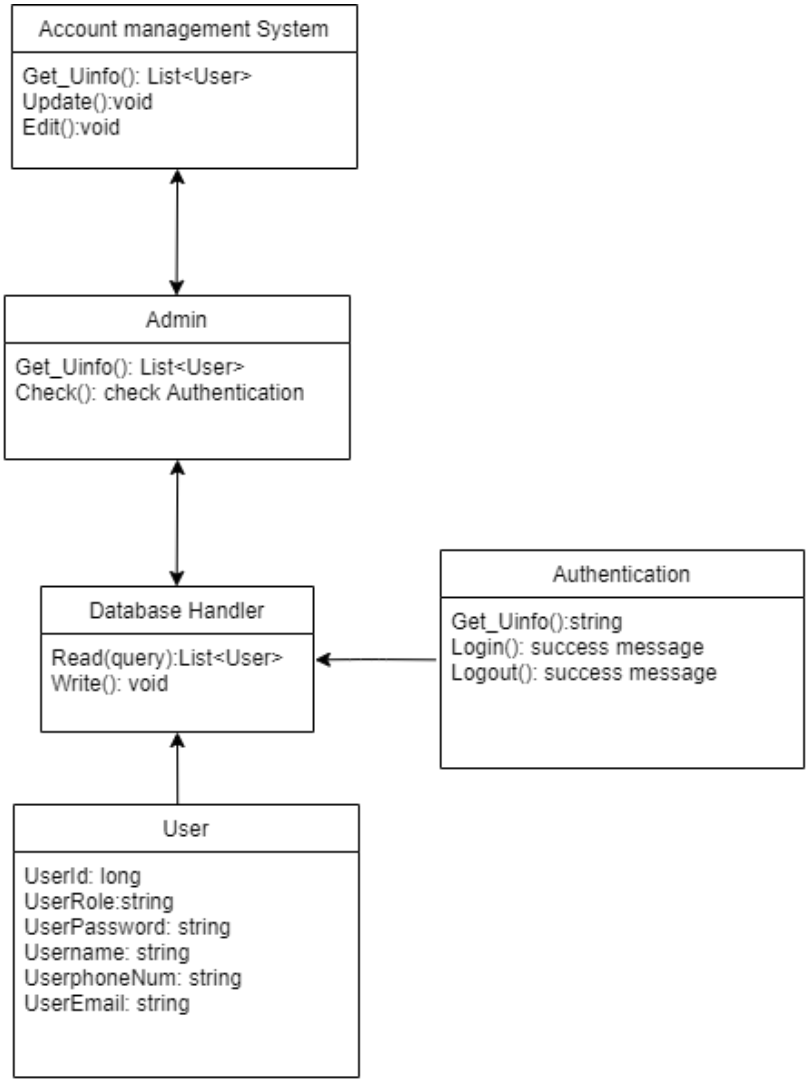

FIGURE 17 ACCOUNT MANAGEMENT CLASS DIAGRAM

5.3.1.2 Sequence Diagram

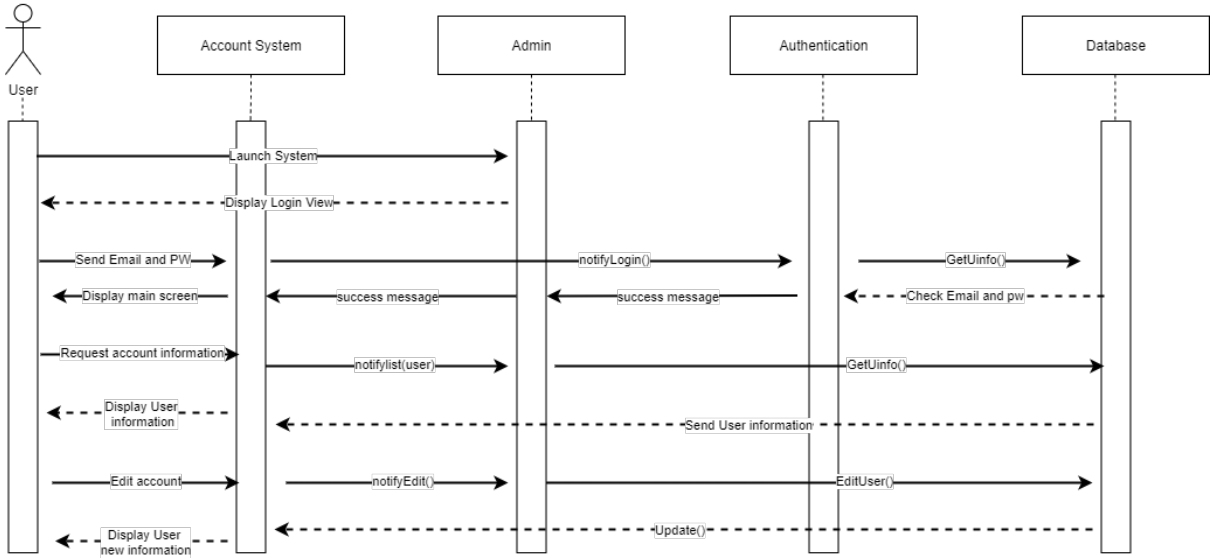

FIGURE 18 ACCOUNT MANAGEMENT SEQUENCE DIAGRAM

5.3.2 RECOMMENDATION SYSTEM

Interface for café recommendations. When a user presses the cafe recommendation button, the appropriate cafe is recommended for the user through the user's current location, cafe location, and cafe current waiting status information.

5.3.2.1 Class Diagram

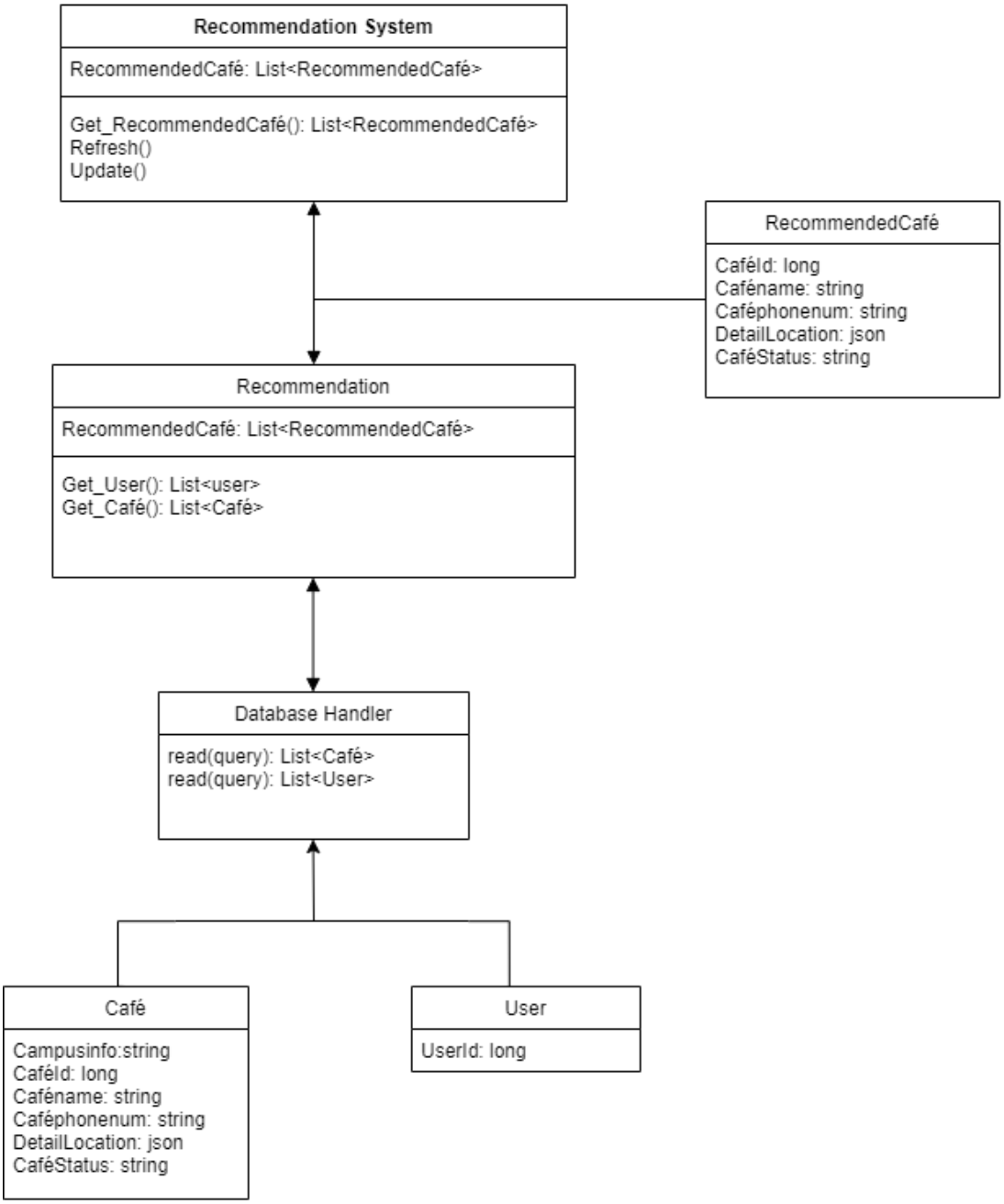

FIGURE 19 RECOMMENDATION SYSTEM CLASS DIAGRAM

5.3.2.2 Sequence Diagram

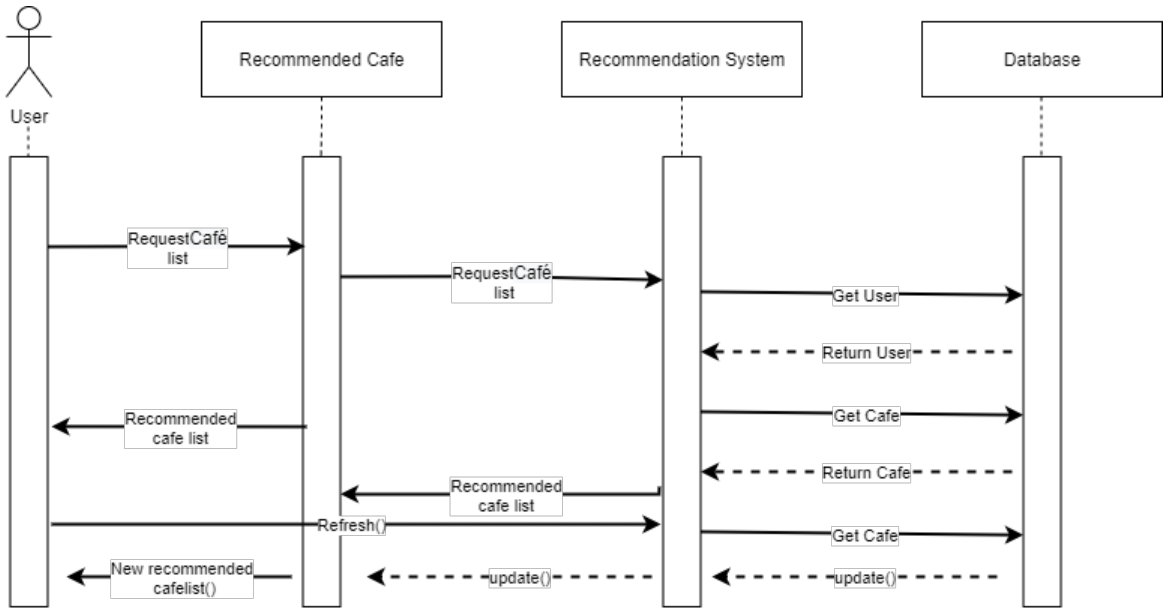

FIGURE 20 RECOMMENDATION SEQUENCE DIAGRAM

5.3.3 ORDERING SYSTEM

Interface for ordering. If the user selects the desired menu and puts it in the cart, the payment information is connected to the payment method and requests the user to purchase it.

5.3.3.1 Class Diagram

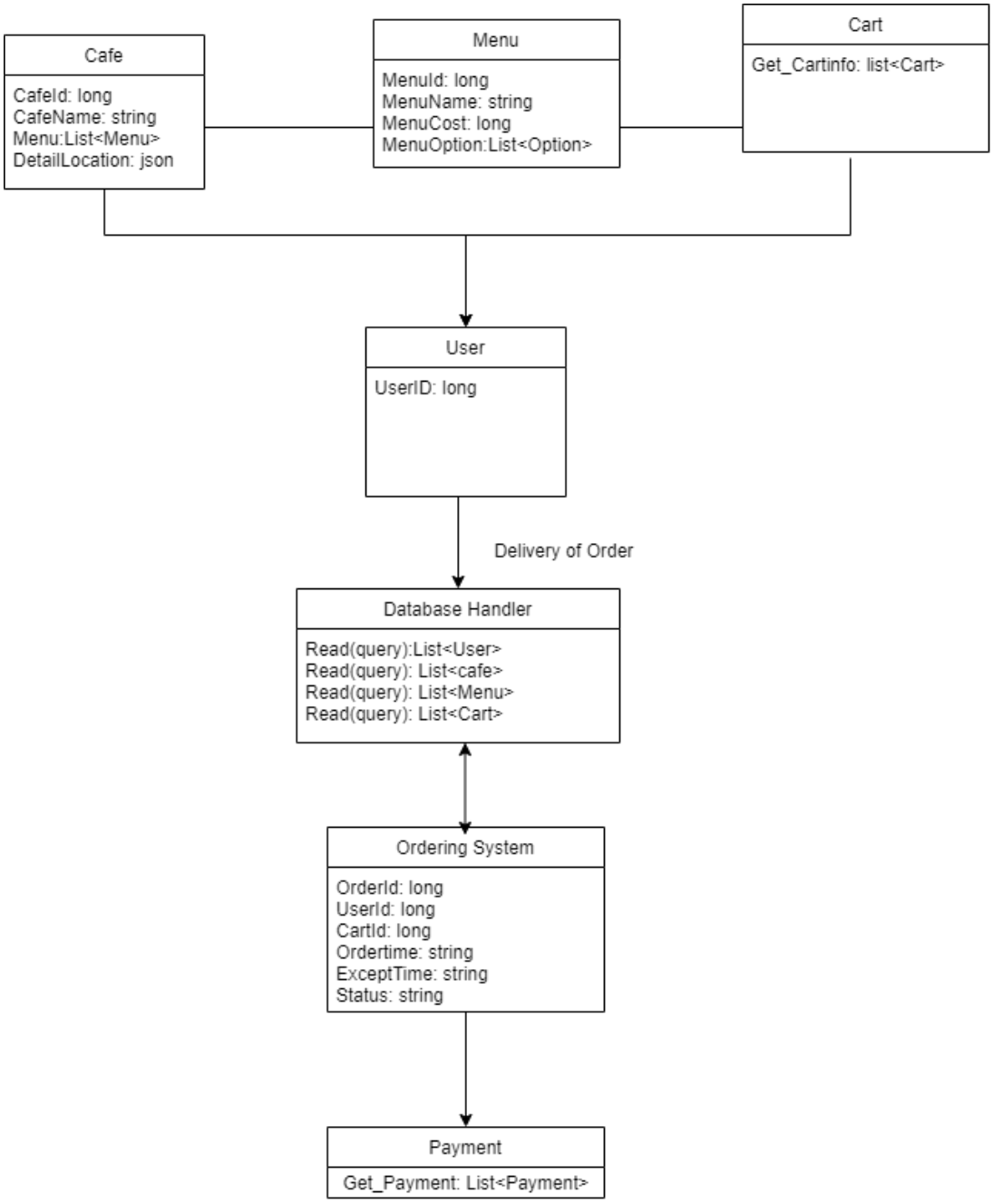

FIGURE 21 ORDERING SYSTEM CLASS DIAGRAM

5.3.3.2 Sequence Diagram

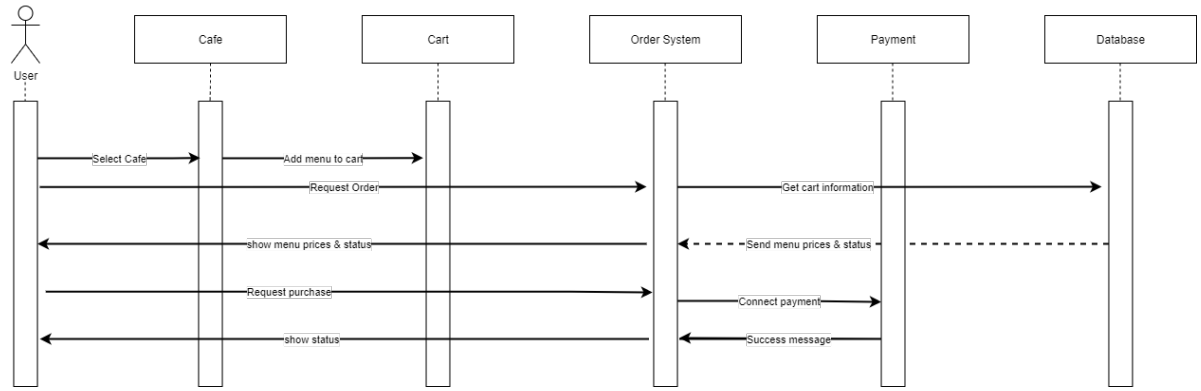

FIGURE 22 ORDERING SYSTEM SEQUENCE DIAGRAM

6. PROTOCOL DESIGN

6.1 OBJECTIVES

A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information. This chapter describes communication protocols used for interaction between KINGO COFFEE application and the server. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods.

6.2 REST API

REST stands for the term “Representational State Transfer”. It refers to displaying resources through URI, determining the behavior of the resource using HTTP METHOD, and receiving the result. An API designed by following the REST-based rules is called REST API or RESTful API.

6.2.1 URI RESOURCES

The URI should represent the resource of information. Resource names use nouns rather than verbs. Since URIs should be focused on representing resources, there should be no expressions for actions. (The HTTP METHOD and verb expressions for actions should not be included in the URI.)

6.2.2 HTTP METHODS

TABLE 1 HTTP METHODS

Method	Action
POST	A POST request is used to send data to the server.
GET	The GET method is used to retrieve information from the given server using a given URI.
PUT	Replaces all current representations of the target resource with the uploaded content.
DELETE	Removes all current representations of the target resource given by a URI.

6.2.3 STATUS CODE

TABLE 2 STATUS CODE

Status Code	Meaning
200 – OK	Everything is working.
201 – CREATED	A new resource has been created.
204 – NO CONTENT	The resource was successfully deleted, no response body.
304 – NOT MODIFIED	The date returned is cached data (data has not changed).
400 – BAD REQUEST	The request was invalid or cannot be served. The exact error should be explained in the error payload.
401 – UNAUTHORIZED	The request requires user authentication.
403 – FORBIDDEN	The server understood the request but is refusing it or the access is not allowed.
404 – NOT FOUND	There is no resource behind the URI.
500 – INTERNAL SERVER ERROR	If an error occurs in the global catch blog, the stack trace should be logged and not returned as a response.

6.3 AUTHENTICATION

6.3.1 REGISTER

6.3.1.1 Request

TABLE 3 TABLE OF REGISTER REQUEST

Attribute	Detail	
Method	POST	
URI	/authentication/register	
Parameter	name	User name
	phone_number	User phone number
	email	User Email
	password	User password
	campus	Selected campus

6.3.1.2 Response

TABLE 4 TABLE OF REGISTER RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (Already existing Email)

6.3.2 LOGIN

6.3.2.1 Request

TABLE 5 TABLE OF LOGIN REQUEST

Attribute	Detail	
Method	POST	
URI	/authentication/login	
Parameter	email	User Email
	password	User password

6.3.2.2 Response

TABLE 6 TABLE OF LOGIN RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (ID or password is not right)

6.4 MAP

6.4.1 MAP VIEW

6.4.1.1 Request

TABLE 7 TABLE OF MAP VIEW REQUEST

Attribute	Detail	
Method	GET	
URI	https://maps.googleapis.com/maps/api/staticmap?	
Parameter	center	Current location(lat, lan)
	key	Google map API key
	campus	Selected campus (optional)

6.4.1.2 Response

TABLE 8 TABLE OF MAP VIEW RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	403 FORBIDDEN (Invalid API key)

6.4.2 RECOMMENDATION

6.4.2.1 Request

TABLE 9 TABLE OF RECOMMENDATION REQUEST

Attribute	Detail

Method	GET	
URI	/map/recommendation	
Parameter	arrival	Estimated arrival(moving) time
	waiting	Estimated waiting time

6.4.2.2 Response

TABLE 10 TABLE OF RECOMMENDATION RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	404 NOT FOUND

6.4.3 SELECT CAFE

6.4.3.1 Request

TABLE 11 TABLE OF SELECT CAFE REQUEST

Attribute	Detail	
Method	POST	
URI	/map/café/{café_id}	
Parameter	-	-

6.4.3.2 Response

TABLE 12 TABLE OF SELECT CAFE RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (Invalid Request format)

6.5 CAFE

6.5.1 WAITING STATUS

6.5.1.1 Request

TABLE 13 TABLE OF WAITING STATUS REQUEST

Attribute	Detail	
Method	GET	
URI	/status/{café_id}	
Parameter	-	-

6.5.1.2 Response

TABLE 14 TABLE OF WAITING STATUS RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	404 NOT FOUND

6.5.2 VIEW ITEMS

6.5.2.1 Request

TABLE 15 TABLE OF VIEW ITEMS REQUEST

Attribute	Detail	
Method	GET	
URI	/menu/{café_id}	
Parameter	-	-

6.5.2.2 Response

TABLE 16 TABLE OF VIEW ITEMS RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	404 NOT FOUND

6.5.3 ORDERING STATUS

6.5.3.1 Request

TABLE 17 TABLE OF ORDERING STATUS REQUEST

Attribute	Detail	
Method	GET	
URI	/ordering/{user_id}	
Parameter	-	-

6.5.3.2 Response

TABLE 18 TABLE OF ORDERING STATUS RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	404 NOT FOUND

6.6 CART

6.6.1 ORDER

6.6.1.1 Request

TABLE 19 TABLE OF ORDER REQUEST

Attribute	Detail	
Method	POST	
URI	/order	
Parameter	order_id	Order ID
	user_id	User ID
	cost	Total cost

6.6.1.2 Response

TABLE 20 TABLE OF ORDER RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (Invalid Request format)

6.6.2 PAYMENT

6.6.2.1 Request

TABLE 21 TABLE OF PAYMENT REQUEST

Attribute	Detail	
Method	POST	
URI	https://kapi.kakao.com/v1/payment/ready	
Parameter	cid	Cafe code
	partner_order_id	Order code
	item_name	Item name
	quantity	Item quantity
	total_amount	Total price
	approval_url	Redirect url if payment is approved
	cancel_url	Redirect url if payment is canceled
	fail_url	Redirect url if payment is failed
Header	Authorization	Kakao admin key

6.6.2.2 Response

TABLE 22 TABLE OF PAYMENT RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (이미 진행중인 거래가 있습니다.)

6.6.3 Cancel Payment

6.6.3.1 Request

Table 23 Table of Cancel Payment Request

Attribute	Detail	
Method	POST	
URI	https://kapi.kakao.com/v1/payment/cancel	
Parameter	cid	Cafe code
	tid	Payment code
	cancel_amount	Price to be cancelled
Header	Authorization	Kakao admin key

6.6.3.2 Response

Table 24 Table of Cancel Payment Response

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (원거래 없음)

6.7 Profile

6.7.1 Get Profile

6.7.1.1 Request

Table 25 Table of Get Profile Request

Attribute	Detail	
Method	GET	
URI	/profile/:id	
Parameter	-	-

6.7.1.2 Response

TABLE 26 TABLE OF GET PROFILE RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	404 NOT FOUND

6.7.2 SET PROFILE

6.7.2.1 Request

TABLE 27 TABLE OF SET PROFILE REQUEST

Attribute	Detail	
Method	POST	
URI	/profile/:id	
Parameter	name	User name
	phone_number	User phone number
	email	User Email
	password	User password
	campus	Selected campus

6.7.2.2 Response

TABLE 28 TABLE OF SET PROFILE RESPONSE

Attribute	Detail
Success Code	200 OK
Failure Code	400 Bad Request (Already existing Email)

7. DATABASE DESIGN

7.1 OBJECTIVES

This section describes the system data structures and how these are to be represented in a database. It first identifies entities and their relationship through ER-diagram (Entity Relationship diagram). Then, it generates Relational Schema and SQL DDL (Data Description Language) specification.

7.2 ER DIAGRAM

The system consists of five entities which are User, Order, Product, Café and Total Orders. ER-diagram expresses each entity as rectangular and their relationship as rhombus. Cross line means start of the relationship and trident (three lines) means end of the relationship. For example, in below diagram, User entity has relationship with

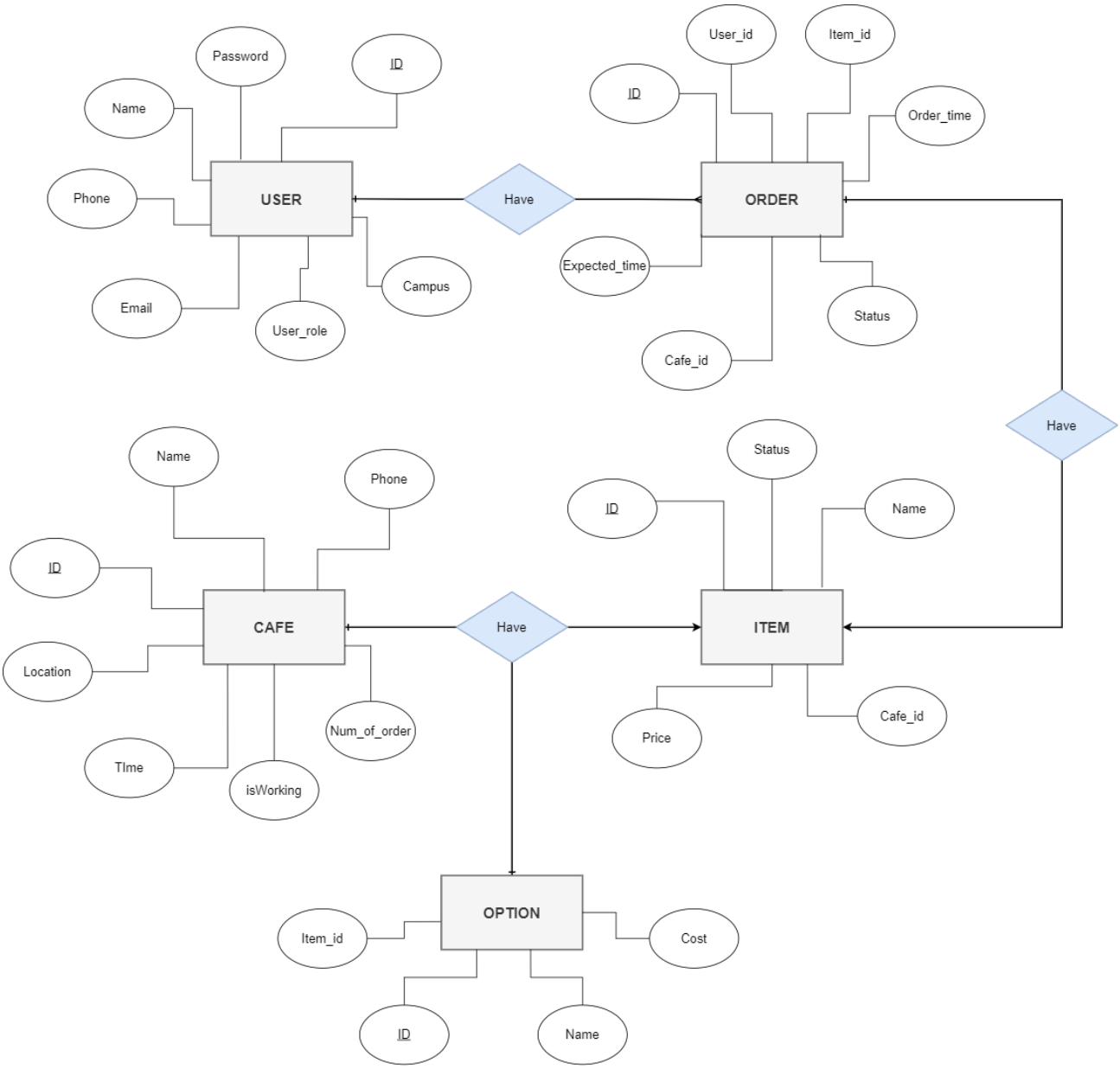

FIGURE 23 ER DIAGRAM

Order entity. And it means User entity have Order entity. The attribute of an entity is expressed as an ellipse. The unique attribute which uniquely identifies an entity (primary key) is underlined.

7.3 ENTITIES

7.3.1 USER

User entity consists of total 7 attributes which are ID, User_role, Campus, Email, Phone, Name, Password. ID is id of user and it is a primary key of User entity. User role means the role of user like customer or cafe manager. Campus means which campus does user prefer or which campus is user occupied. Email is user's email for login, Phone is user's phone number, name is user's name, and Password is user id's password for login.

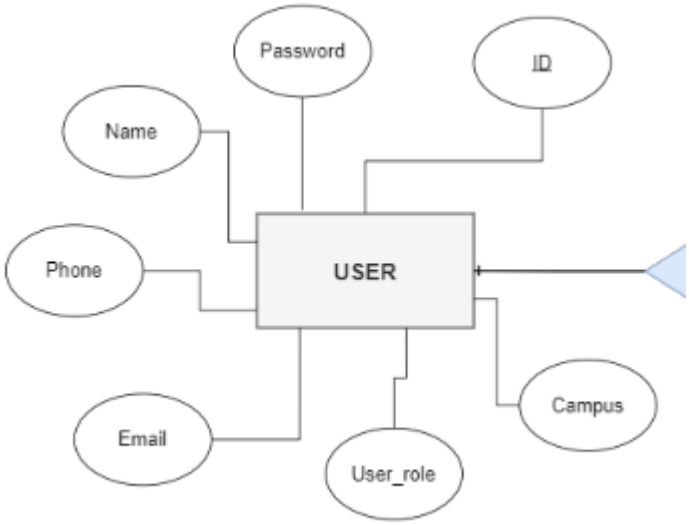

FIGURE 24 ER DIAGRAM, ENTITY, USER

7.3.2 ORDER

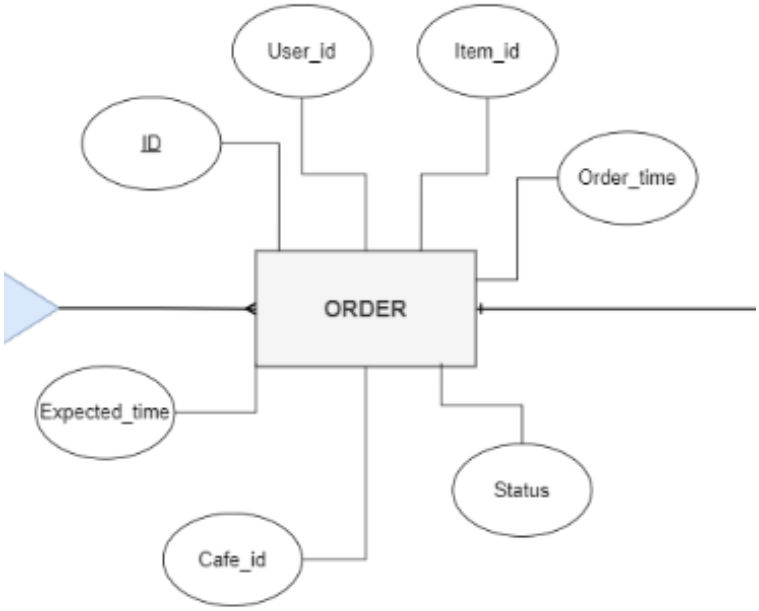

FIGURE 25 ER DIAGRAM, ENTITY, ORDER

Order entity consists of total 7 entities which are ID, User Id, Item Id, Order time, Status, Café Id, and Expected time. ID means order Id which is the distinct number of order and can distinguish between the orders and it is the primary key. User id means id of user who order. Item id is the id of item,Order time is the time when customer order this menu, Status is the complete status which represented by 1,0. 1 means it is not yet taken by customer, and 0 means it is taken. If the status is 0, the order is removed from the database. Café id means the id of café which takes the order. Expected time is expected time of order of the item which is an input by cafe manager.

7.3.3 ITEM

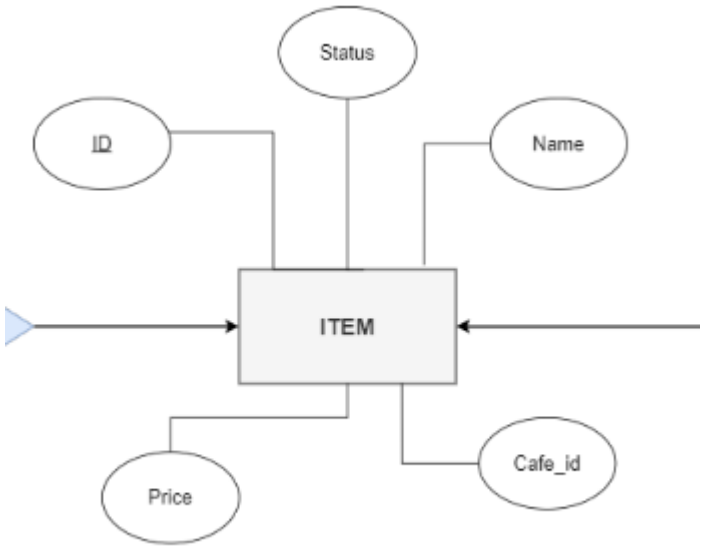

FIGURE 26 ER DIAGRAM, ENTITY, ITEM

Item entity consists of total 5 attributes which are ID, Name, Price, Café Id and Status. ID means the distinct ID of the product which can distinguish product between products. And it is the primary key. Name is the name of the product, Price is the price of the product, Café Id is the id of the café which sales this product. Status means whether it can be selled or not.

7.3.4 OPTION

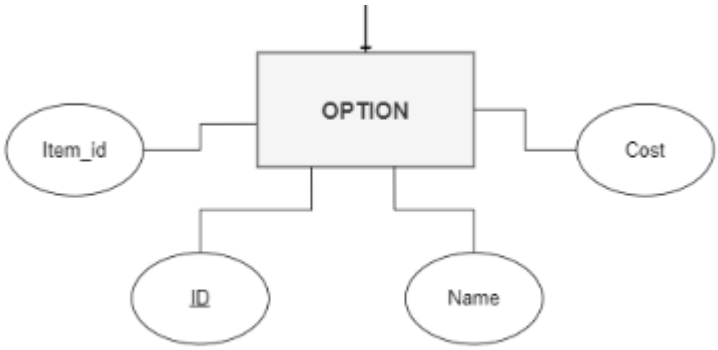

FIGURE 27 ER DIAGRAM, ENTITY, OPTION

Option entity consists of total 4 attributes which are ID, Name, Cost, Item_id. Option is option of menu. For example, like an additional shot for an americano. ID is option id, name is the name of option, Cost is the cost of option and Item id is item id which item can have this option.

7.3.5 CAFÉ

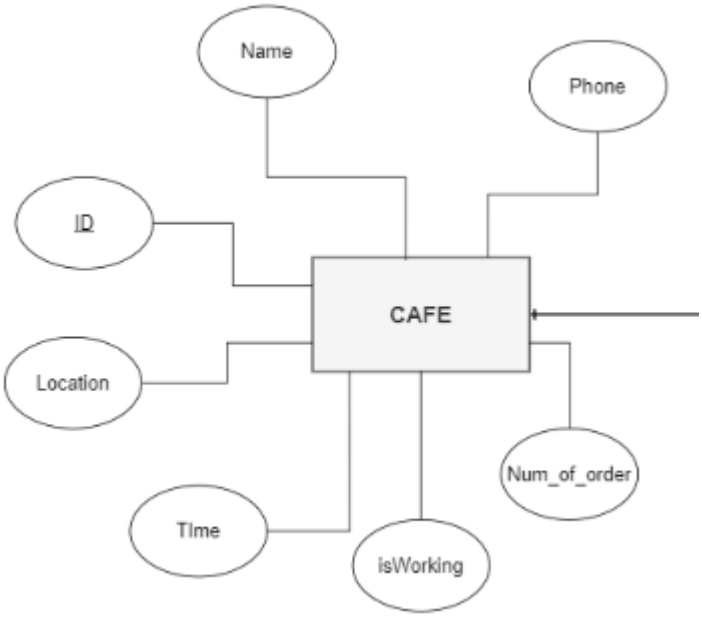

FIGURE 28 ER DIAGRAM, ENTITY, CAFE

Café entity consists of total 8 attributes which are ID, Name, Phone, Num of Order, waiting time, is Working, Time and Location. ID is the distinct ID of the café and is primary key. Name is the name of the café, Phone is the phone number of the café, Num of order is the current total number of orders in the café, waiting time is the current waiting time of the café which calculated from the num of order. Is working means whether the café is working or not. Time is the working hour of the café and Location is the location of the café.

7.4 RELATIONAL SCHEMA

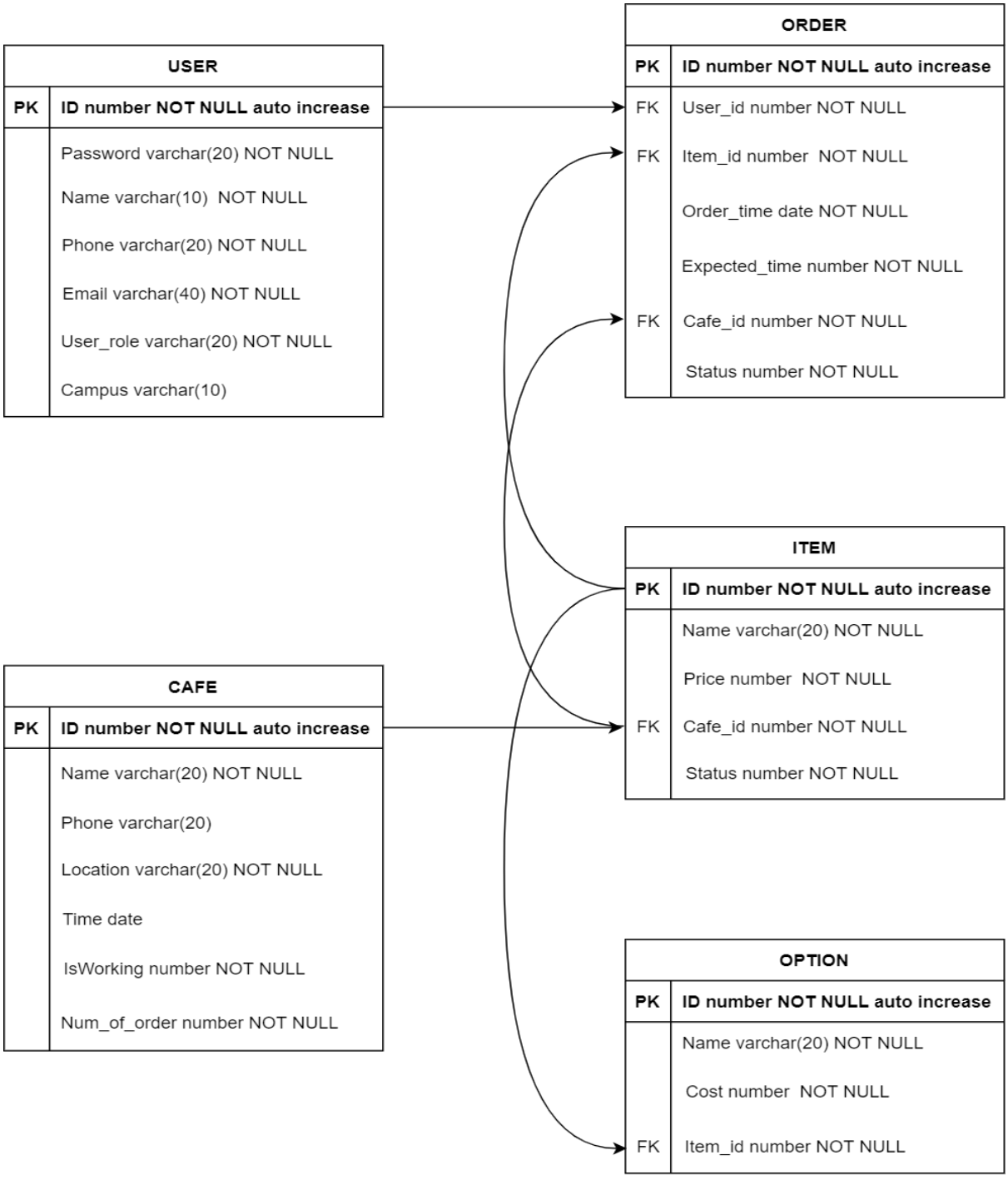

FIGURE 29 RELATIONAL SCHEMA

7.5 SQL DDL

7.5.1 USER

```
CREATE TABLE USER
```

```
(
    ID number  NOT NULL auto increase,
    Password varchar(20) NOT NULL,
    Name varchar(10) NOT NULL,
    Phone varchar(20) NOT NULL,
    Email varchar(40) NOT NULL,
    User_role  varchar(20)  NOT NULL,
    Campus varchar(10)
) PRIMARY KEY(ID)
```

7.5.2 ORDER

```
CREATE TABLE ORDER
(
    ID number  NOT NULL auto increase,
    User_id number  NOT NULL,
    Item_id number  NOT NULL,
    Order_time  date NOT NULL,
    Expected_time  number NOT NULL,
    Cafe_id number  NOT NULL,
    Status number NOT NULL
    FOREIGN KEY (User_id) REFERENCES USER(ID),
    FOREIGN KEY (Item_id, Cafe_id) REFERENCES ITEM(ID, Cafe_id)
) PRIMARY KEY(ID)
```

7.5.3 ITEM

```
CREATE TABLE ITEM
(
    ID number  NOT NULL auto increase,
```

```
    Name varchar(20) NOT NULL,
    Price number NOT NULL,
    Cafe_id number  NOT NULL,
    Status number NOT NULL,
    FOREIGN KEY (Cafe_id) REFERENCES CAFE(ID)
) PRIMARY KEY(ID)
```

7.5.4 OPTION

```
CREATE TABLE OPTION
(
    ID number  NOT NULL auto increase,
    Name varchar(20) NOT NULL,
    Cost number NOT NULL,
    Item_id number  NOT NULL,
    FOREIGN KEY (Item_id) REFERENCES ITEM(ID)
) PRIMARY KEY(ID)
```

7.5.5 CAFÉ

```
CREATE TABLE CAFE
(
    ID number  NOT NULL auto increase,
    Name varchar(20) NOT NULL,
    Phone varchar(20),
    Time datetime,
    Location varchar(20) NOT NULL,
    isWorking number NOT NULL,
    Num_of_order number NOT NULL
) PRIMARY KEY(ID)
```

8. Testing Plan

8.1. Objectives

This chapter describes plans for development testing, release testing, and user testing to be implemented in our system. Through testing, we check that our system runs in the intended direction, and detect that there are no internal potential errors and defects.

Figure 30 Testing Plan

8.2. Testing Policy

8.2.1. Development Testing

It is a process to reduce the risk, development time, and development cost of software and to increase the quality of software and the efficiency of the overall development process. Development testing performs several tasks, including static code analysis, data flow analysis, peer code reviews, and unit testing, corresponding to component testing, integration testing, and system testing.

8.2.1.1. Performance

Our system stores most of the data on a server and obtains and renews data through communication with the server, so the latency of this communication is a large part of the performance. Therefore, it is necessary to test the configuration and communication of efficient databases to ensure an appropriate level of stable latency. In addition, algorithms for implementing recommendation systems should also be designed efficiently and should not take long to present recommendation cafes to

users. Therefore, the efficiency of the recommendation algorithm should also be tested. For characteristics such as Reliability, security, and maintainability in a trade-off relationship with Performance, it is necessary to test whether performance has been sacrificed excessively and to improve if excessive performance degradation occurs.

8.2.1.2. Security

It is necessary to test that the user authentication process is properly performed to prevent unauthorized users from accessing and using the system to cause damage to the system. It should also be tested to ensure that the personal information of users stored in the server's database is well encrypted and protected. Additionally, it should be tested to ensure that there are no loopholes in communication between the payment API and the system and that there are no security issues.

8.2.1.3. Reliability

It should provide accurate and reliable information and data about the user's request. Therefore, it should be tested how consistent the actual cafe status is with the current status shown in the system. In order for the system to function without failure, the component testing in the development process and the integration testing in the integration process of the components must be thoroughly tested and repeatedly tested gradually.

8.2.1.4 Maintainability and Sustainability

The system utilizes already commercialized cloud services to maximize the ease of server management and maintenance. In addition, the system should be organized to make it easier to reflect various changes, such as adding new cafes, changing menus, and changing prices. Therefore, we check that the server is designed to be easy to maintain and fluctuate.

8.2.2 Release Testing

The main goal of the release testing is to convince the customer that the system is sufficient for use. Therefore, release testing should show that the system provides specified features, performance, and characteristics and does not fail during normal use. Scenario testing is used to validate and demonstrate the system. We devise general usage scenarios and use them to conduct release testing. Scenarios should be

realistic and in practice similar to users using the system. Scenarios in the requirements engineering process can also be utilized for scenario testing. Release testing begins with the alpha version, where the basic implementation of the software has been completed.

8.2.3 User Testing

Usability is one of the key factors in this system, so it should be easy for anyone to use and intuitive to instinctively use all functions without a system manual. Thus, in user testing, Usability, as well as characteristics such as Performance and Reliability, is an important test perspective. We release the Beta version to a restricted group of users to test based on the user's actual usage data and also receive feedback.

8.2.4 Test cases

Test cases are generated according to the characteristics of performance, reliability, security, maintenance, and sustainability. Additionally, the test is conducted by generating a test case to verify the system's fulfillment of the required functions. Prepare an evaluation sheet to verify the results of the test.

9. Development Plan

9.1. Objectives

This chapter describes the technology, environment, language, etc. necessary for the development of the system.

9.2. Environment

9.2.1. Kotlin

FIGURE 31 KOTLIN LOGO

Kotlin is a programming language that operates on Java Virtual Machine (JVM) and is currently officially designated by Google's Android app development. It has quite concise grammar compared to Java and is 100% supported for interoperation with Java. Therefore, it is a language with a dramatic decrease in the amount of code and a significant improvement in productivity.

9.2.2. ADOBE PHOTOSHOP

FIGURE 32 ADOBE PHOTOSHOP LOGO

Adobe Photoshop is a graphical tool developed and published by Adobe that enables image synthesis, editing, and drawing.

9.2.3. ADOBE XD

FIGURE 33 ADOBE XD LOGO

Adobe Xd is a UI/UX design software developed and published by Adobe that supports vector design, wireframe and prototyping.

9.2.4. Android Studio

Figure 34 Android Studio Logo

Android Studio is an official integrated development environment (IDE) for Android and Android-only applications. It has advantages such as making the developing system executable immediately, intelligent code editors for productive code writing, and supporting multiple emulators.

9.2.5 Amazon Web Services

Figure 35 Amazon Web Services Logo

Amazon Web Services provides all kinds of services needed to build IT infrastructure. All services provided by AWS are natively HTTP, REST, and SOAP, characterized by API control. It also provides a variety of environments for development, testing, and apps, as well as various types of databases available at the end. It is easy to maintain as it integrates the back-end into AWS as a single cloud.

9.2.6 Spring boot

FIGURE 36 SPRING BOOT LOGO

Spring boot can easily create commercially viable, spring-based applications that only need to be run and run alone. With minimal settings, spring platforms and third-party libraries are available. Spring boot Create stand-alone Spring applications, Embed Tomcat, Jetty or Undertow directly, Automatically configure Spring whenever possible and Absolutely no code generation and no requirement for XML configuration.

9.2.7 MYSQL

FIGURE 37 MYSQL LOGO

MySQL is the world's most popular open-source relational database management system. It is used not only to manage client-internal data on the system, but also to manage Amazon relational database service in conjunction with AWS.

9.3. VERSION CONTROL SYSTEM

9.3.1. GIT

FIGURE 38 GIT LOGO

Git is a version control system for tracking file changes and coordinating the operations of these files among multiple users. It features very fast speed and distributed storage support. It has advantages such as offline work through local storage, fast speed, easy and stable branch and merge, and free use. In our system, version management is done using Github server.

9.4. CONSTRAINTS

The system is designed and implemented based on the content of the document. Other details shall be designed and implemented in the direction preferred by the developer, and the following restrictions shall be observed:

- Use technology that has already been widely demonstrated.
- Avoid using technology or software that requires a license or financial costs.
- Unauthorized parts should not be used arbitrarily because information on facilities in schools is used.
- When using external APIs, the relevant constraints and regulations should be observed.
- It should be designed and implemented considering the characteristics of the system's Performance, Security, Reliability, Maintenance, and Sustainability.
- Refactoring source code should be executed frequently.
- The application develops with minimum Android version 6 and target Android version 10.

9.5. ASSUMPTIONS AND DEPENDENCIES

All systems in this document are designed and implemented based on Android devices. (minimum Android version 6 and target Android version 10) Therefore, the system may not work if it does not fit other operating systems or versions. It also

assumes, designs and implements development permits for cafes in schools and permits for point schemes.

10. SUPPORTING INFORMATION

10.1. SOFTWARE DESIGN SPECIFICATION

This software design specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).