2020-1 SOFTWARE ENGINEERING ***TEAM5***

Nutrition Cart

Design Specification

2020318170 Christian Klose

2016312860 Kim Seonji

2015310326 Kim Joohwan

2016310944 Lee Minyeong

2015310547 Lee Minho

2017310301 Kim Seungyoon

Contents

1. Preface

This chapter defines the expected readership of the document and document structure to briefly introduce each chapter.

1.1. Readership

The intended readers of this document are various. Therefore, an explanation of each part includes who is the intended reader of each part.

1.2. Document Structure

1) Introduction

   This chapter describes the tools and diagrams which are used for this project and defines the scope of this project.

2) System Architecture

   This chapter describes the overall structure of the system and the subsystem. For the better understanding of the reader, this chapter explains the system architecture with various diagrams such as class diagram and sequence diagram.

3) Protocol Design

   This chapter explains interfaces and protocols which controls the interaction between frontend system and backend system. The structure of the interfaces and the technology used for implementing it is described,

4) Database Design

   This chapter defines entities, attributes, and relationships between entities using ER diagram based on the database requirement defined in the requirement specification. Referencing the ER diagram, relational schema for the system database is confirmed. Also, this chapter includes SQL DDL sentences to build tables.

5) Testing Plan

For the verification and the validation of the system, this chapter describes the plan for the test.

6) Development Plan

This chapter includes the introduction of development environments such as development tools, programming language, library which is used to implement the system. This chapter also includes the development schedule.

7) Index

This chapter describes the picture, table, and diagram that used in this document.

2. Introduction

2.1. Objectives

This chapter describes the tools and diagrams which are used for this project and defines the scope of this project.

2.2. Applied Diagram

A. UML

UML, Unified Modeling Language, is a general-purpose, developmental modeling language in the field of software engineering. It is intended to provide a standard way to visualize the design of the system. It combines different aspects of the system to represent relations between sub-systems, processes, and results of the overall system. We use UML in this document to visualize the workflow of the system. – class diagram, sequence diagram, ER diagram.

B. Class Diagram

Class diagram is a type of static structure diagram that describes the structure of the system by visualizing the system's classes, attributes, methods, and the relationships among the objects. This diagram provides a clear distinction between each class and shows a clear relationship between them.

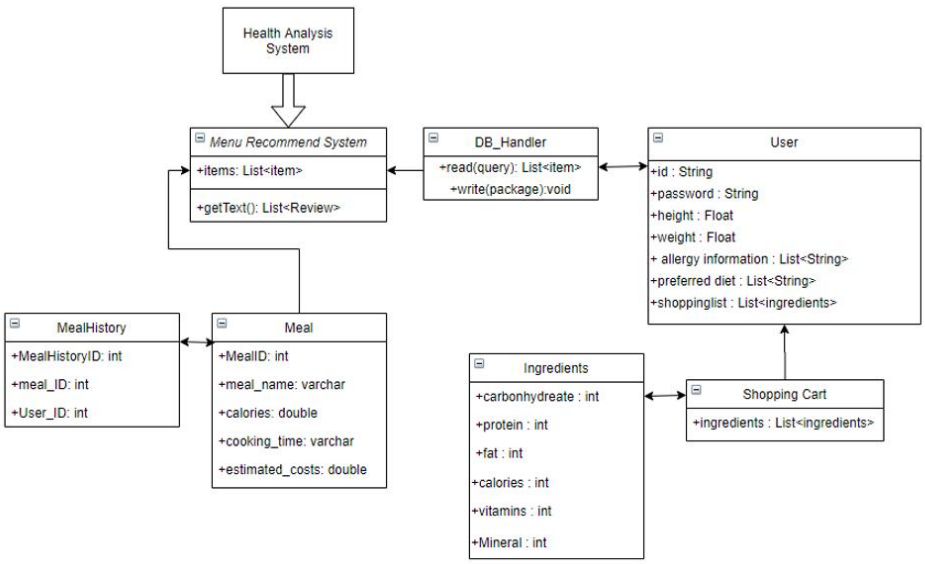

Figure 1 : Example of Class Diagram

C. Sequence Diagram

Sequence diagram, which is also called event diagram or event scenarios, is an interaction(dynamic) diagram that details how operation are carried out. It shows object interaction arranged in time sequence using parallel vertical lines and horizontal arrows. Vertical lines mean processes or objects that live simultaneously and horizontal arrows mean the messages exchanged between them in the order in which they occur. This diagram specifies the simple runtime scenario in a graphical manner.

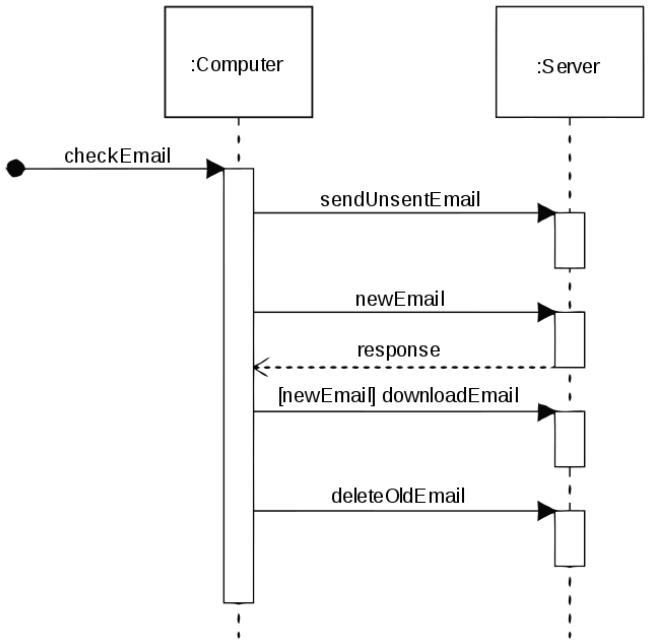

Figure 2 : Example of Sequence Diagram

D. ER Diagram

ER Diagram, Entity-Relationship diagram, visualizes the relationships of entity sets stored in a database. An entity in this context is an object which is a component of data. An entity set is a collection of similar entities. Entities have attributes that defines its properties. ER diagram illustrates the logical structure of database by drawing the entities, attributes of each entity, and relationships between them. Relational schema and SQL DDL are written based on this diagram.

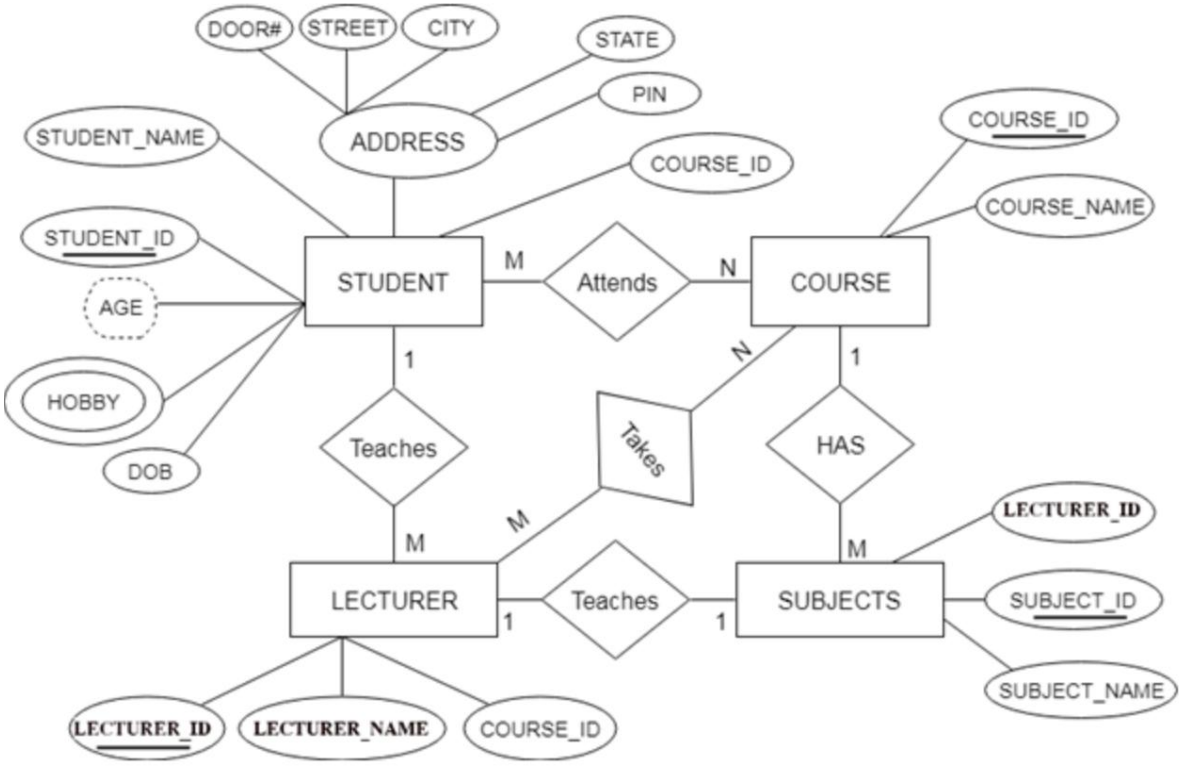

Figure 3 : Example of ER Diagram

E. Data Flow Diagram

Data flow diagram is a way of representing a flow of a data of a process or a system. It provides information about the outputs and input of each entity and the process itself, but it does not include control flow such as decision rule and loops. Data flow diagram consists of processes, flows, warehouses, and terminators. Processes represents a function which is a component of the system that transforms a inputs to outputs. Flows show the transfer of information from one part of the system to another using arrows to show the flow direction and it link processes, warehouses, and terminators. Warehouses are used to store data for later use. It can represent various things such as data file, folder with documents, or optical discs etc. The flow coming out from the warehouse is the reading of the data stored in it and the flow coming to the warehouse is data entry or updating. Terminator is an external entity that communicates with the system and stands outside of the system. It can represent various organizations, groups, or a department.

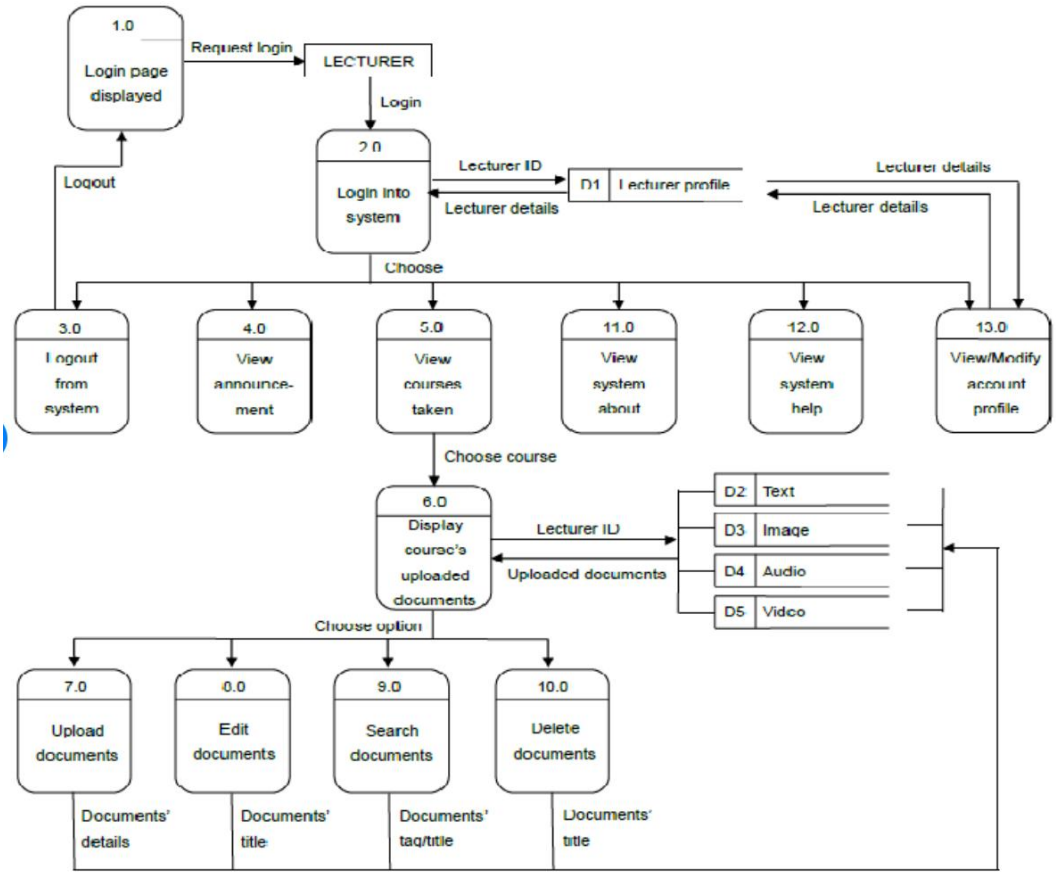

Figure 4 : Example of Data Flow Diagram

2.3. Project Scope

Begin with the strict social distancing because of the worldwide disease, this project focus on the situation that people evade to go offline market and rely on online shopping instead. With the increasing rate of online shopping, this project suggests a new platform that provide a convenient application for groceries.

<Nutrition Cart> is the system that recommend ingredients and menu for the user based on the user's personal information such as the ingredients that user puts in their cart, health condition, purchasing or viewing history. The major point of this system is that, unlike the other online grocery application, this system emphasizes a personalization service based on the user's personal information. Therefore, the main function of this system is item recommendation based on the user information. In addition to the recommendation function, this system provide meal planning function, list of cooking instruction and nutrition of ingredients, purchasing ingredients function, delivery service.

Frontend system directly interacts with the user, responses to the data request from backend system, and calls each subsystem - health analysis system, menu recommend system, ingredient merchandise system. Frontend system presents personal information section, menu plan section, menu recommend section, and ingredient merchandise section to the user. The user information such as weight, height, and preferred diet or meal category is stored in the user database and when the user purchase item through this platform, the purchasing history is also stored in the user database. A nutrition information of each ingredients and meals are stored in the ingredient database and the menu database. Ingredients which is sold in the merchandise section is stored in the product database.

The health analysis system utilizes the user's meal plan and personal information to analyse which nutrition in the status of the user's cart are lacking or overdosed. According to the user information and nutrition information, user is provided the ingredients recommendation to buy or to add to the cart. This recommendation system utilizes the result from the health analysis system using diet category that the user selects. The user can get a menu or ingredients recommendation which is suitable to

their taste and nutrition status. Through the recommendation, the platform lead the user to purchase ingredients in this platform. The user can add the ingredients of the menu they are recommended to their own shopping list and make an order in the ingredient merchandising system.

3. System Architecture – Frontend

3.1. Objective

In this section a detailed description of the frontend is given. The front-end consists of individual components, which in turn can consist of components. Services are provided for some main components. These services handle the data exchange with the backend. In the following, all main components and services are listed. Thereafter, they are represented using class diagrams. A short description is given for each class diagram.

3.2. Subcomponents

3.2.1. Component

- User Component
- Meal Component
  - Meal Component
    - Weekly Meal Plan
    - Recommended Meals
  - Meal Detail Component
    - Nutrition's
    - Ingredients
    - …
- Shopping Component (incl. List of all Items, Order Functionality and Payment)
- Delivery Component

3.2.2. Services

- User Service (Register, Password Reset, change personal Information)
- Auth Service (Login/logout, verify User Token)
- Meal- Plan, Detail Service (Handles the Data communication for Meal Component)
- Shopping Service (Handles the Data communication for Shopping Component)

- Delivery Service (Handles the Data communication for Delivery Component)

3.3.3. Class Diagram of Front-End Architecture

A. User Component & Service

The **UserComponent** allows the user to view his personal information. If the user wants to change his stored data, he can do so at any time. To do so, a request is sent to the backend via the **UserService** to transfer the changes to the database. Furthermore, general functions such as Q&A or customer service are available to the user

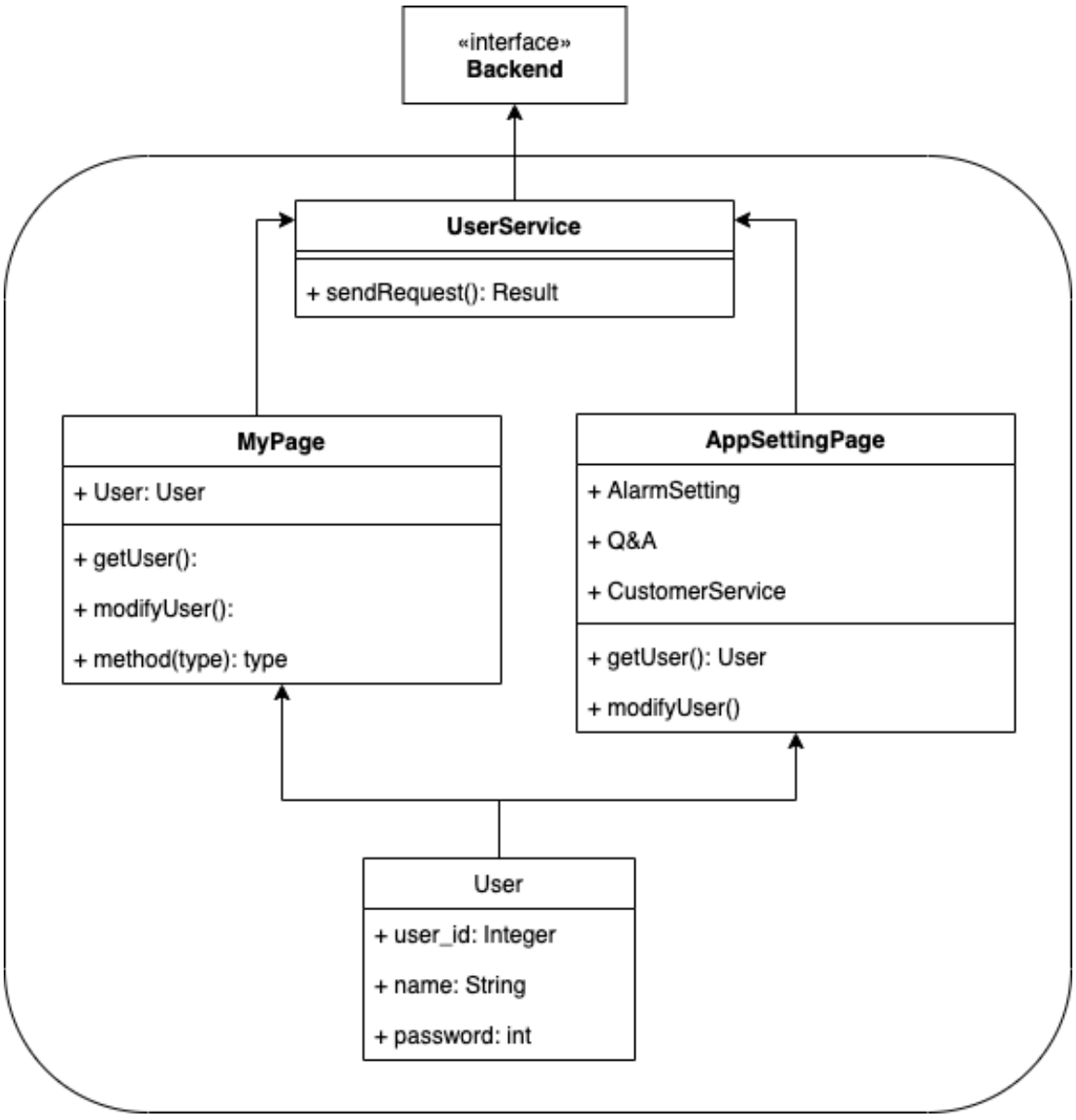

Diagram 1 : User Component and Service – Class Diagram

B. Meal Component

The Meal component consists of two subcomponents. The **Meal Plan Component** and the **Meal Detail Component**.

- Meal Plan Component

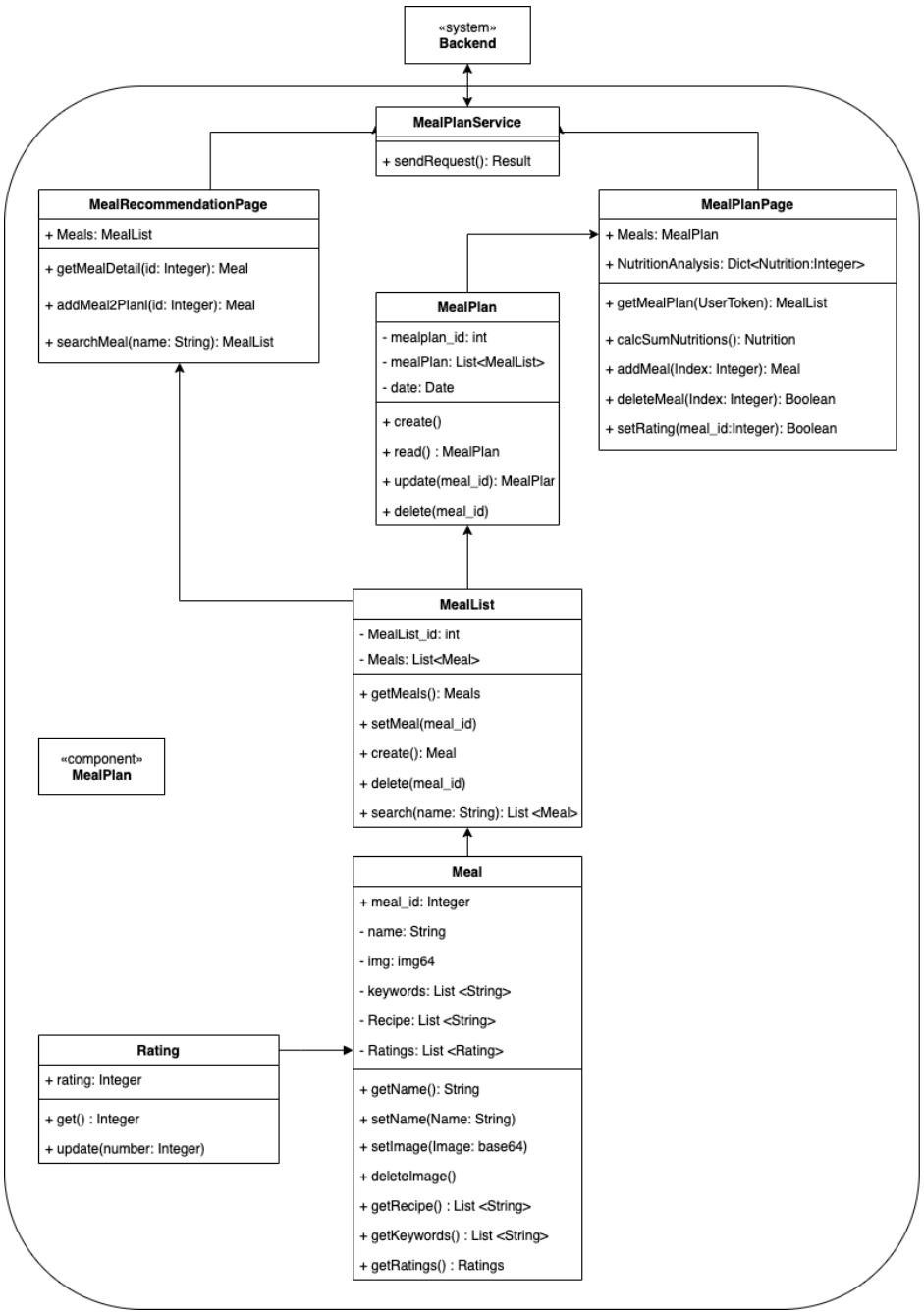

Diagram 2 : Meal Plan Component - Class Diagram

As shown in the diagram, the component consists of two pages. **Meal Plan Page** shows the main overview, **Meal Recommendation Page** shows the recommendations tailored to the user profile. Several classes based on each other are required for data storage, consisting of Meal, Meal List and Meal Plan.

- Meal Detail Component & Service

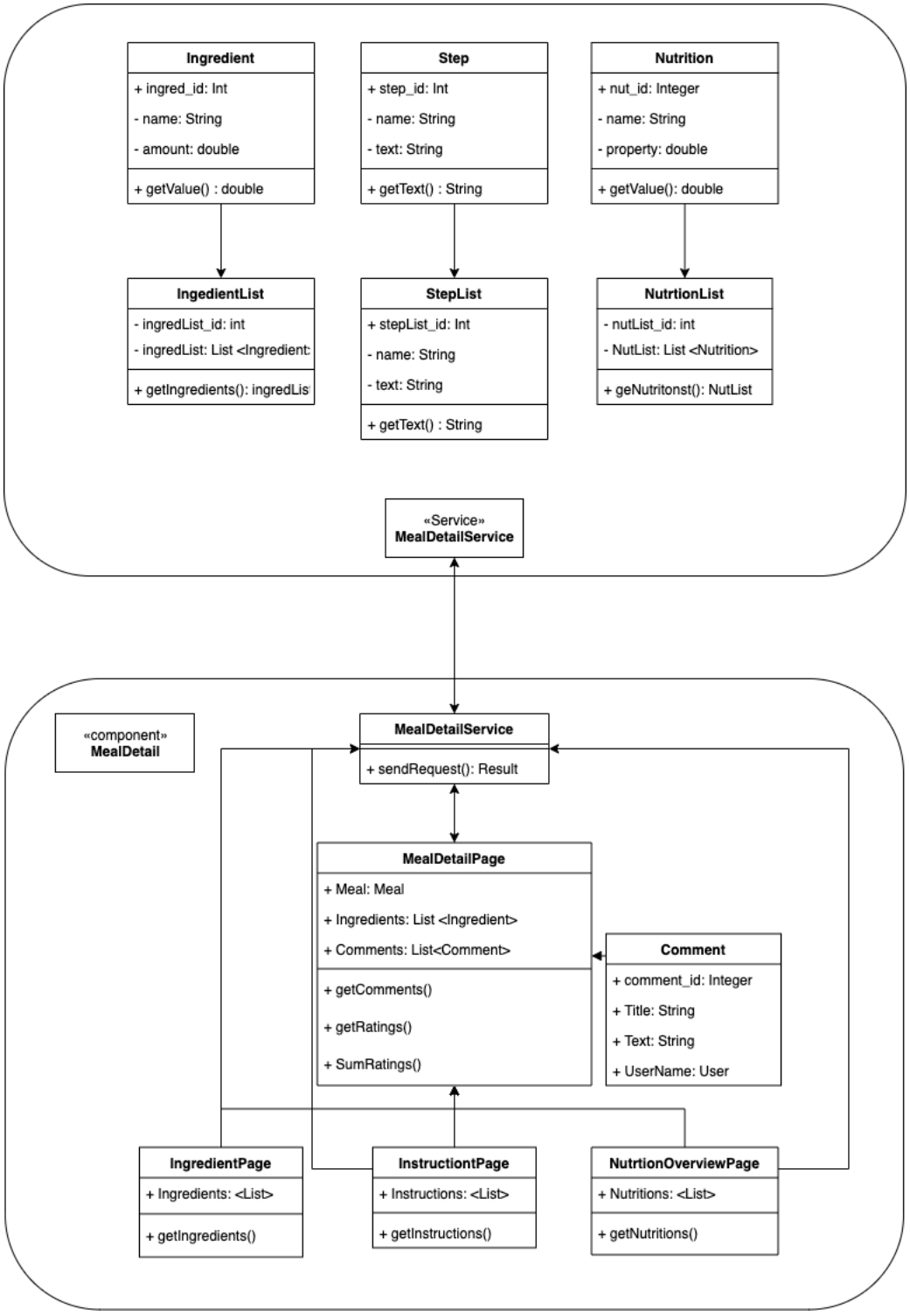

Diagram 3 : Meal Detail Component & Service - Class Diagram

In this diagram, the relationship between component and service is explained in detail. By using the same names for the functions, it is made clear how the function of a component is related to its counterpart of a service.

C. Shopping and Delivery Component

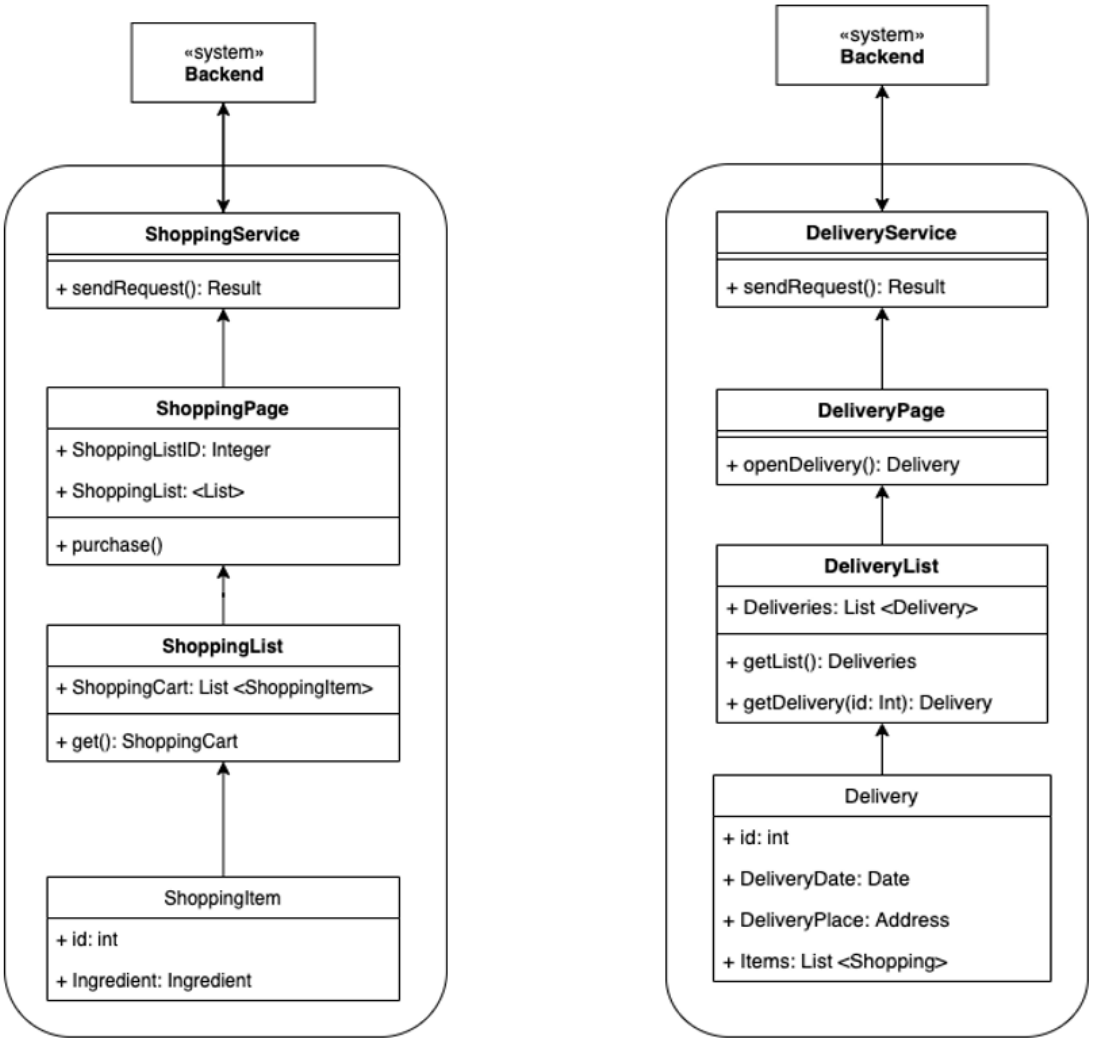

Diagram 4 : Shopping and Delivery - Class Diagram

Both Shopping and Delivery Component and Services are very similar; they both consist of a single page that is made up of a single class through a single list. For the data connection with the backend, the service bearing the same name as the page is available.

At present, both consist of a single page. However, this may change in the near future. This is also the reason why both pages are listed separately.

4.System Architecture – Backend

4.1. Objectives

This chapter deals with the backend system and the structures of each subsystem.

4.2. Overall Architecture

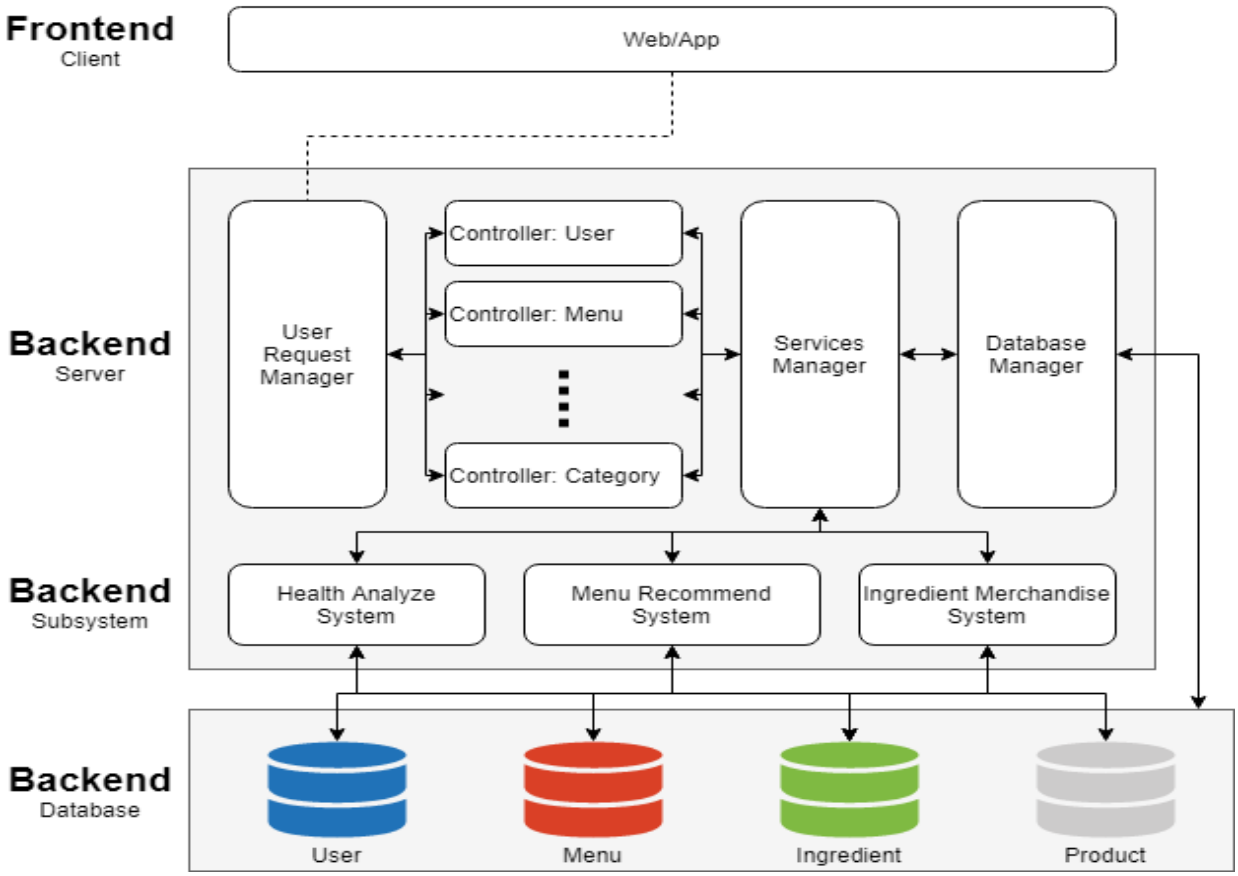

Diagram 5 : System Architecture - Backend – Overall

Above diagram is the overall structure of the backend system. The backend architecture is designed to handle the requests from users or admin. Server, which is in charge of a direct communication with frontend, includes user request manager, services manager, and database manager. User request manager in the server mediate the users' requests and responses to the requests. Requests from the users need several interactions to get a response to it. Service manager in the server interacts with subsystems which have various functions and database through the database manager, get responses to the request and send it to the user request manager.

5.3. Subcomponents

Subcomponents of the backend interacts with services manager in the server and database.

4.3.1. Application Server

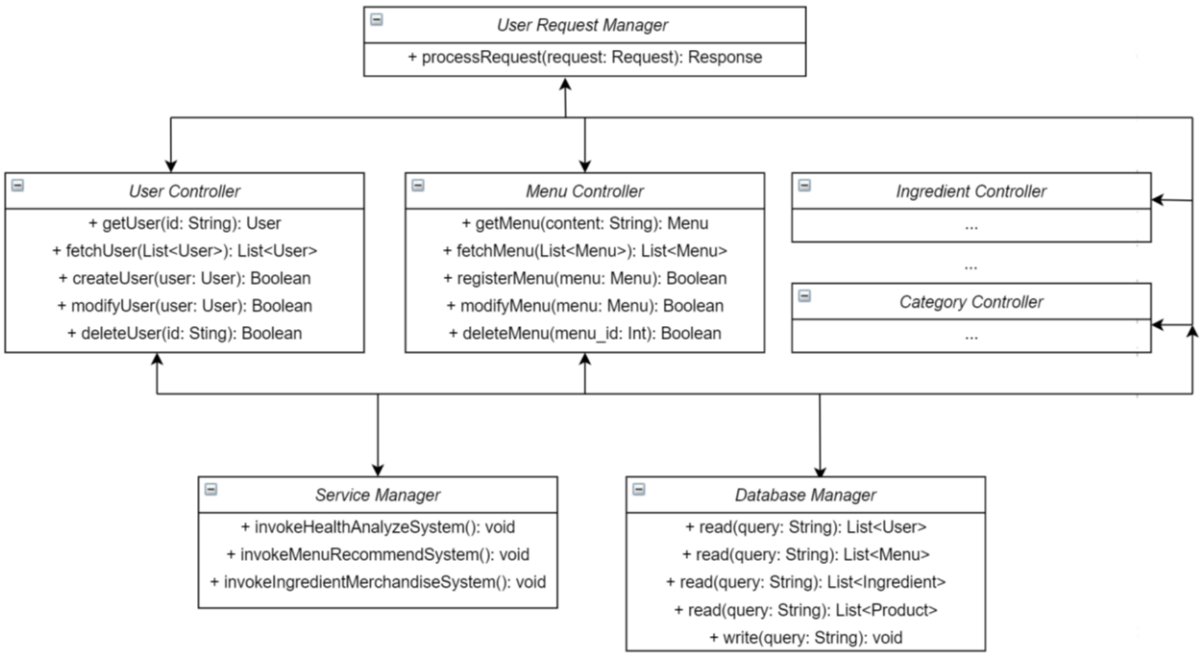

Diagram 6 : Application Service - Class Diagram

1) User Request Manager

   This entity handles requests from frontend

   A. Attributes: None

   B. processRequest(request: Request): Distribute the request comes to the server to each controller, handler the request, and return the result.

2) Controllers

   A. Attribute: None

   B. get[entity](entity_id): get entities

   C. fetch[entity](List<String>): get entity lists matching with the request

   D. create[entity](entity: Entity): create entity

   E. modify[entity](entity: Entity): modify entity

F. delete[entity](entity_id): delete entity

3) Services Manager

A. invokeHealthAnalysisSystem(): call health analysis system

B. invokeMenuRecommendSystem(): call menu recommend system

C. invokeIngredientMerchandiseSystem(): call ingredient merchandise system

4) Database Manager

A. read(query: String): read input query string from each database

B. write(query: String): write input query string to each database

5.3.2. Health Analyse System

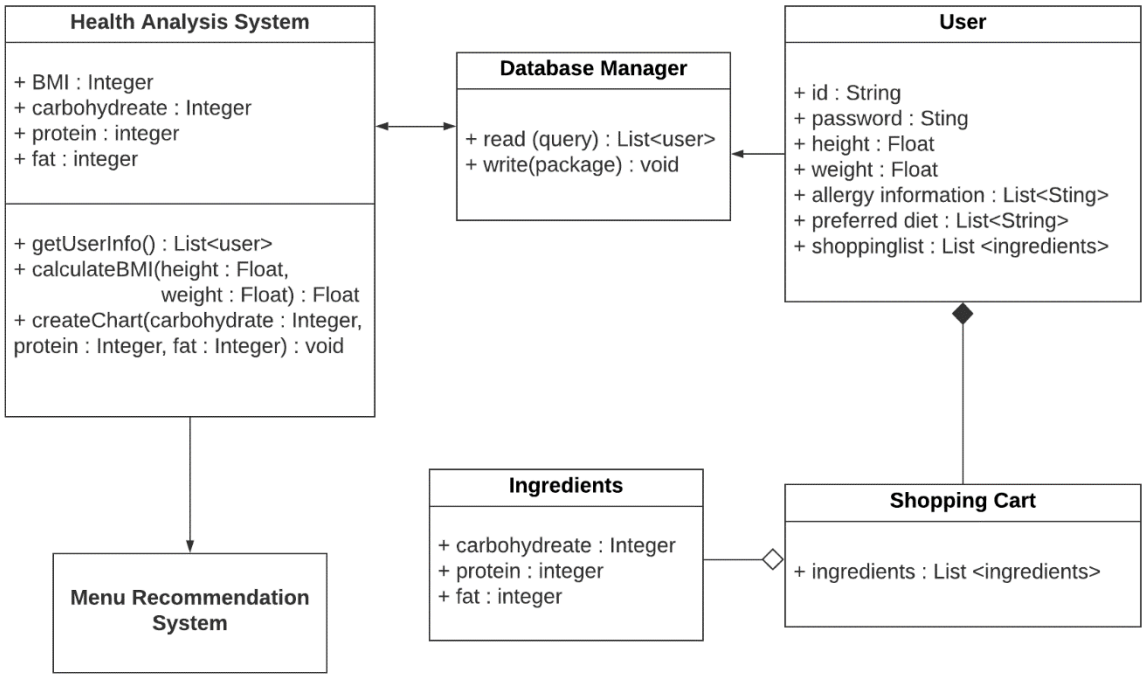

Diagram 7 : Health Analysis System – Class Diagram

Classes Description

1) Health Analysis System Class:

   This class works as a class for analyzing health based on the user's body information and nutrition information of ingredients. This class uses BMI value calculated by height and weight and carbohydrate, protein, and fat values of ingredients. It contains following methods. The results of this class is stored in the user database and used for the menu recommendation system.

   a. getUserInfo: get body information of the user
   b. calculateBMI: calculate BMI based on height and weight of the user
   c. createChart: create nutrition chart based on the nutrition information of the ingredients

2) Database manager:

This class interacts with the backend database through two main methods

a. read: this method retrieves user data from the user database. Health analysis system class need user's body information – weight and height – to calculate the BMI. According to the user's BMI, the system provides the chart that shows recommended nutrition intake.
b. write: this method updates data in the database

Sequence Diagram

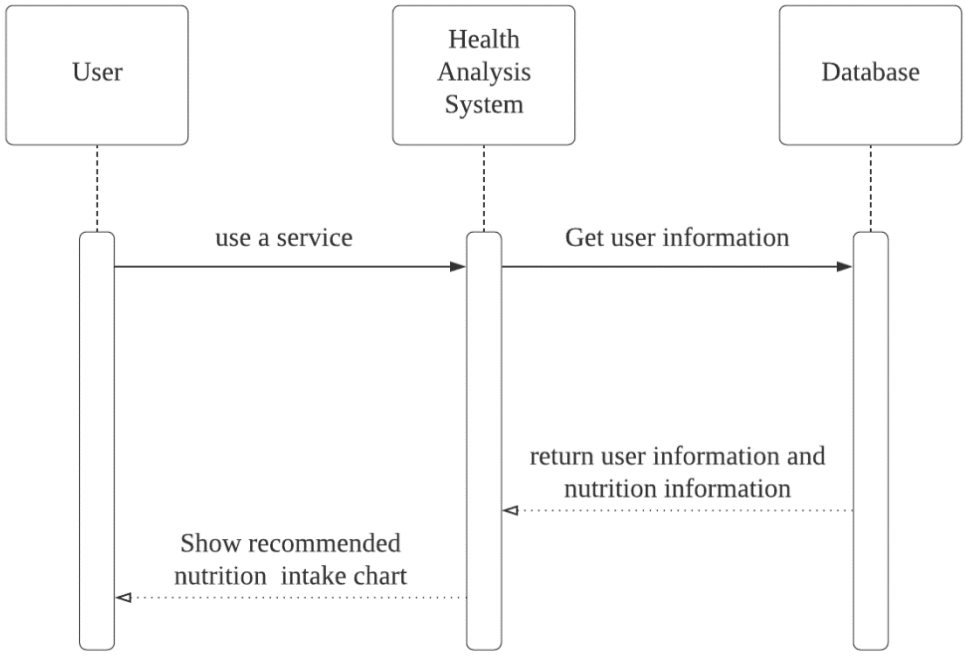

Diagram 8 : Health Analysis System - Sequence Diagram

4.3.3. Menu Recommend System

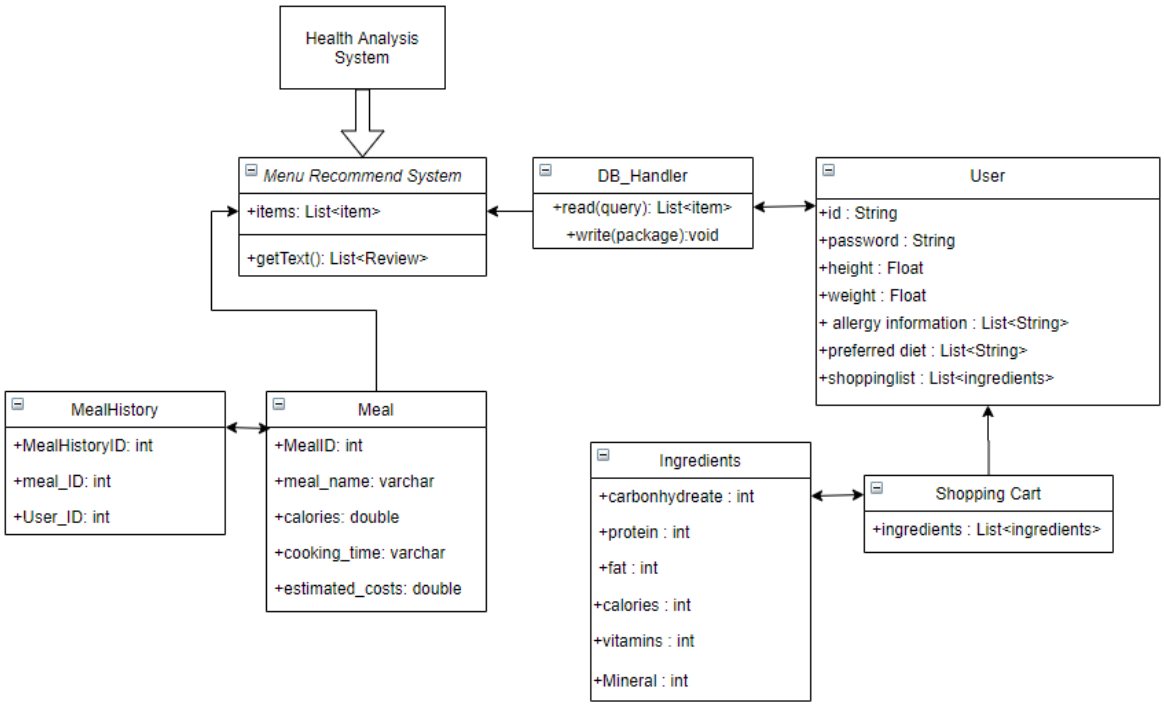

Diagram 9 : Menu Recommendation System – Class Diagram

Classes Description

1) User Class:

   This class stores information related to the user. In particular, this system recommends the user's health information and the list of ingredients in the shopping list as data.Cart Class:

2) Meal class:

   This class contains information on various dishes. This includes the name of the dish, calories, and cooking time.

3) MealHistoryClass:

   This is a class that stores a record of user selections. We provide a more appropriate recommendation by using both other users' records and their own records.

4) Database manager:

This class interacts with the backend database through two main methods.

a. read: This method retrieves user data from the user database. The menu recommendation system reads data from the User class to read the shopping list containing the ingredients desired by the user, the preferred diet, and basic body information.

b. Write: This method updates the data in the database

5) Shopping Cart Class:

This class contains a list of products the user has.

6) Ingredients Class:

This class holds information about materials. It contains information about the five major nutrients of ingredients, including calories, and these are used in recommendation systems for users' health.

4.3.4 Ingredient Merchandise System

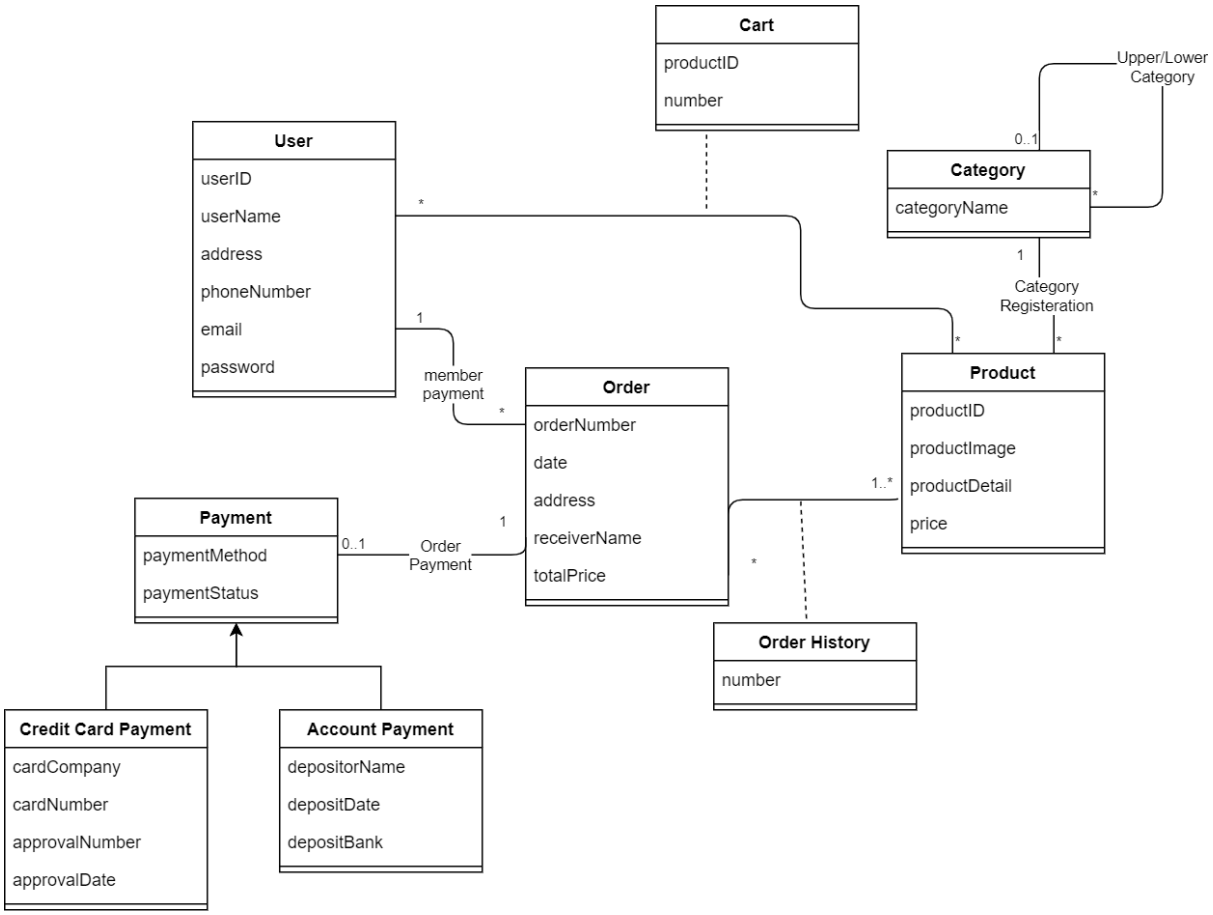

Diagram 10 : Ingredient Merchandising System - Class Diagram

Classes Description

7) User Class:

   This class is a class for storing individual user's personal information. The data of this class is stored in the user database and used for the health analysis system, menu recommendation system, and Ingredient Merchandising System.

8) Cart Class:

   This class is a class that displays items that the user wants to purchase. This class contains product ID and number of products in the cart.

9) Product Class:

This class is a class about the specific information of the product. This class contains product ID, product image, details about the product, and price of the product.

10) Order Class:

This class is the class about ordering products from the market. This class contains order number, ordered date, address to deliver, name of the receiver, and total price of the products. In this class, users could decide whether to order or not.

11) Payment Class

This class is a class to make payment. This class contains payment method and payment status. There are two payment methods, one is using a credit card, and the other one is deposit the exact money to the seller's account.

Sequence Diagram

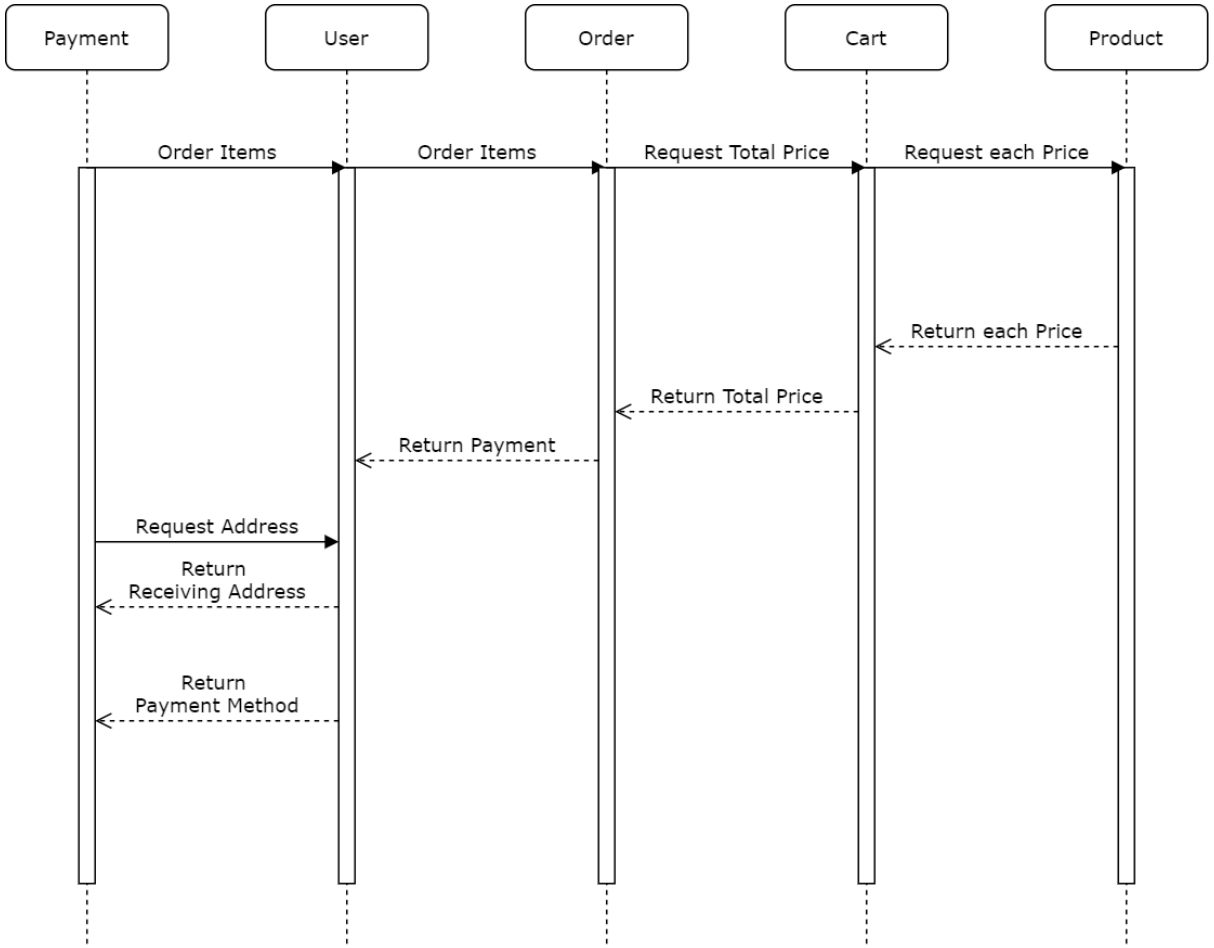

Diagram 11 : Ingredients Merchandising System - Sequence Diagram

5. Protocol Design

5.1. Objectives

This chapter explains how the system interacts with subsystems, how the interfaces are defined, and what kinds of protocols are used for connection between frontend and backend.

5.2. Application Programming Interface

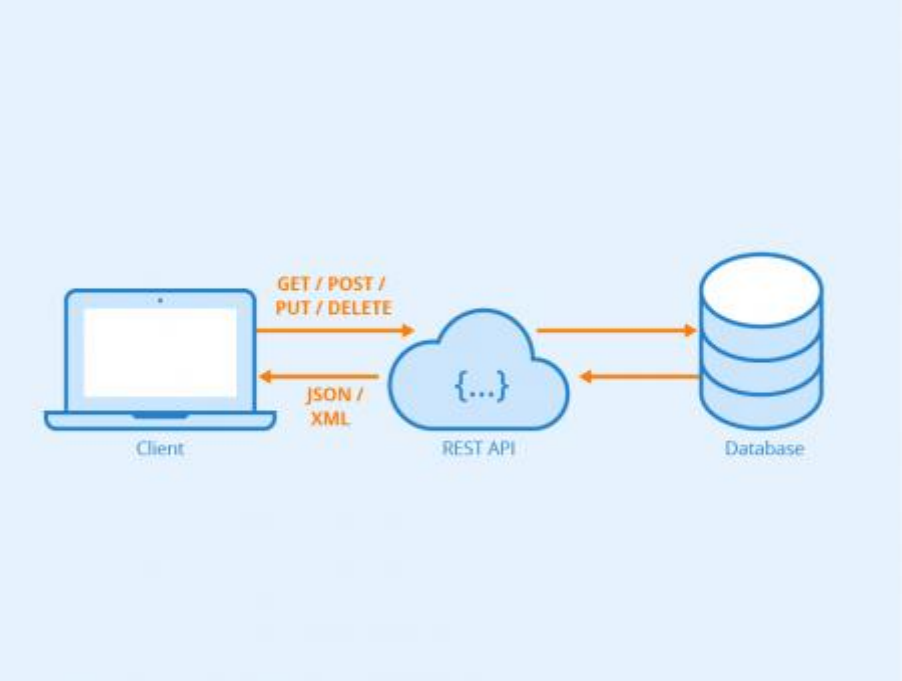

Figure 5 : Application Programming Interface

Rest API(Representational State Transfer) is a network-based software architecture, and will be implemented in the system for communication. It has six principles; '*Client-server architecture*', '*Statelessness*', '*Caching*', '*Uniform interface*', '*Layered system*' and '*Code-on-demand*'. It provides the complete separation of the user interface from server and database. From this, it improves portability of the interface to other types of platforms. As it is independent of the type of platform, it adapts to the type of platform being used. Its request type for resources is the HTTP method such as GET, POST, PUT, and DELETE and its response type is json for representation of resources.

5.3. Details

5.3.1. Authentification

A. Signup

Method	Request / POST	
URI	api/user/signup	
Header	user_email	User email address
	user_password	User password
	user_name	User name
Body	-	-

Table 1 : Signup

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Signup successful"
Failure Code	Response / 400 BAD REQUEST	
Failure Response Body	FAILURE	Reason of failure e.g. empty submission

Table 2 : Signup (2)

B. Login

Method	Request / POST	
URI	api/user/login	
Header	user_id	User id
	user_password	User password
Body	-	-

Table 3 : Login

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	Token and message "Login successful"
Failure Code	Response / 404 BAD REQUEST	
Failure Response Body	FAILURE	Reason of failure e.g. password wrong

Table 4 : Login (2)

C. Survey (On first login, afterwards to be changed in myPage)

Method	Request / PUT	
URI	api/user/:id	
Headers	name	User name
	birthdate	User birthdate
	sex	User sex
	height	User height
	weight	User weight
	allergies	User allergies
	For complete list …	… compare to DB-Schema.
Body	-	-

Table 5 : Survey

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Your profile was successfully set up."
Failure Code	Response / 400 BAD REQUEST	
Failure Response Body	FAILURE	Reason of failure e.g. empty

		submission

Table 6 : Survey (2)

5.3.2 User

A. Get user token

Method	Request / GET	
URI	api/user/:id	
Header	token	User Token
Body	-	-

Table 7 : Get User Token

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	User info e.g. id, password, ...
Failure Code	Response / 403 FORBIDDEN	
Failure Response Body	FAILURE	"Unauthorized access" if Token expired

Table 8 : Get User Token (2)

B. Modify user information

Method	Request / PUT	
URI	api/user/:id	
Header	token	User Token
	user_info	User information e.g. password, email
Body	-	-

Table 9 : Modify User Information

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Modification successful"
Failure Code	Response / 403 FORBIDDEN	
Failure Response Body	FAILURE	"Unauthorized access" if token expired

Table 10 : Modify User Information (2)

5.3.3. Meal & Ingredient

A. Get a nutrition information of a user's meal plan

Method	Request / GET	
URI	api/meal/:id/list	
Header	token	User Token
Body	-	-

Table 11 : Get a nutrition information of a user's meal plan

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	Meal list object. Proportion graph of nutritions in meal list in respect to nutrient intake standard. If the meal list is empty, an empty graph will be returned.

Table 12 : Get a nutrition information of a user's meal plan (2)

B. Get a list of meals

Method	Request / GET	
URI	api/meal/:id/list	
Header	Token	User Token
Body	-	-

Table 13 : Get a List of Meal

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	Meal list object. User specific Meal List including all meals of a user. If the user has no items in the meal list, an empty list will be returned.

Table 14 : Get a List of Meal (2)

C. Add a meal to a list

Method	Request / POST	
URI	api/meal/:id/list	
Header	Token	User Token
	meal_id	Meal ID
Body	-	-

Table 15 : Add a Meal to a List

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully added"
Failure Code	Response / 400 BAD REQUEST	
Failure Response Body	FAILURE	Reason of failure e.g. redundant meal exists in a list.

Table 16 : Add a Meal to a List (2)

D. Delete a meal from a list

Method	Request / DELETE	
URI	api/meal/:id/list	
Header	Token	User Token
	meal_id	Meal ID
Body	-	-

Table 17 : Delete a Meal from a List

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully deleted"
Failure Code	Response / 403 FORBIDDEN	
Failure Response Body	FAILURE	"Unauthorized access" if token expired

Table 18 : Delete a Meal from a List (2)

E. Get information of a meal

Method	Request / GET	
URI	api/meal/info	
Header	meal_id	Meal id
Body	-	-

Table 19 : Get Meal Information

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	Meal object. Information about a meal including ingredients, nutritions, calories, and recipe.

Table 20 : Get Meal Information (2)

F. Search a meal

Method	Request / GET	
URI	api/meal/info	
Header	Token	User Token for Auth
	keywords	Meal keyword e.g. name, preference.
Body	-	-

Table 21 : Search Meal

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	A list of meal objects. If there are no matched meals with keywords, an empty list will be returned.

Table 22 : Search Meal (2)

5.3.4. Shopping

A. Get all items of a shopping list

Method	Request / GET	
URI	api/shopping/:id	
Header	Token	User Token (for Auth)
Body	-	-

Table 23 : Get all Items of a Shopping List

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	Shopping list object. User specific Shopping List including all items of a user. If the user has no items in the shopping list, an empty list will be returned.

Table 24 : Get all Items of a Shopping List (2)

B. Add items to shopping list

Method	Request / CREATE	
URI	api/shopping/:id	
Header	Token	User Token (for Auth)
	meal_id	Meal id (in case of adding a meal)
Body	-	-

Table 25 : Add Items to a Shopping List

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully added"
Failure Code	Response / 400 BAD REQUEST	
Failure Response Body	FAILURE	Reasons of failure e.g. sold out.

Table 26 : Add Items to a Shopping List

5.3.5. Order & Delivery

A. Make an order

Method	Request / POST	
URI	api/order/:id/ api/shopping/:id	
Header	Token	User Token
	shopping_id	Shopping list id
	delivery_info	Delivery information e.g. address, purchase method, credit card number, password.
Body	-	-

Table 27 : Make an Order

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully ordered"
Failure Code	Response / 400 BAD REQUEST	
Failure Response Body	FAILURE	Reasons of a failure e.g. submission failed, empty submission.

Table 28 : Make an Order (2)

B. Cancel an order

Method	Request / DELETE	
URI	api/order/:id	
Header	Token	User Token
	order_id	Order id
Body	-	-

Table 29 : Cancel an Order

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully canceled"
Failure Code	Response / 401 UNAUTHORIZED or 403 FORBIDDEN	
Failure Response Body	FAILURE	Reasons of failure e.g. wrong token, already shipped.

Table 30 : Cancel an Order (2)

C. Get a list of order list

Method	Request / GET	
URI	api/order/:id	
Header	Token	User Token
Body	-	-

Table 31 : Get a List of Order List

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	A list of order objects. User specific Order List. If the user has no items in the order list, an empty list will be returned.

Table 32 : Get a List of Order List (2)

D. Modify delivery information (e.g name, address)

Method	Request / PUT	
URI	api/order/:id	
Header	Token	User Token
	delivery_info	Delivery information e.g. address, phone number.
Body	-	-

Table 33 : Modify Delivery Information

Success Code	Response / 200 OK	
Success Response Body	SUCCESS	"Successfully modified"
Failure Code	Response / 401 UNAUTHORIZED or 403 FORBIDDEN	
Failure Response Body	FAILURE	Reasons of failure e.g. wrong token, already shipped.

Table 34 : Modify Delivery Information (2)

6. Database Design

6.1. Objectives

Detailed database design is described based on the database requirements created in the requirements specification. Through ER Diagram, relationships between general entities are described, and relational schema and SQL DDL specification are created.

6.2. ER Diagram

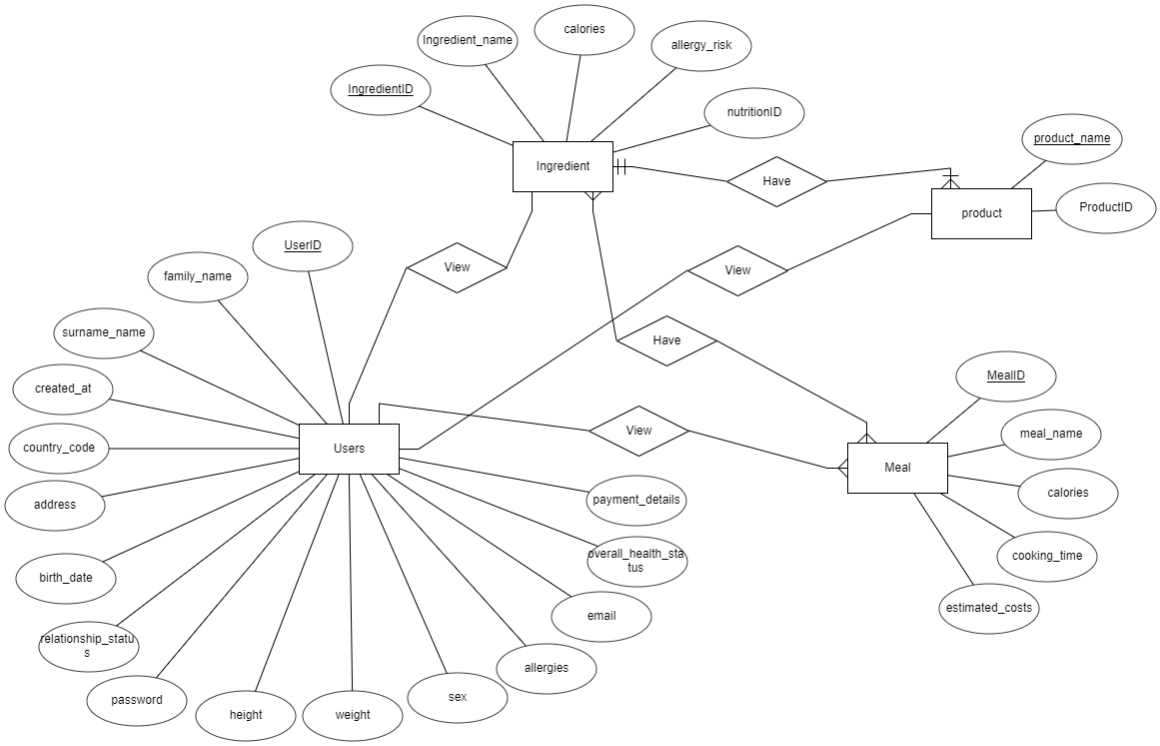

Diagram 12 : ER Diagram – overall system

There are 4 entities in this system: User, Category, Ingredients, and Meal. Each entity is expressed in the form of a square box, and the relationships between entities are expressed in a rhombus. When a specific entity can have multiple relationships with other entities, three trident-shaped lines are drawn on the corresponding entity, and if there is no specific entity, a small circle is displayed on the client entity. The attribute of each entity is represented by an oval. The unique key that identifies each entity is underlined on the label, and when multiple attributes are allowed, the border is doubled.

6.2.1 Entities

A. Users

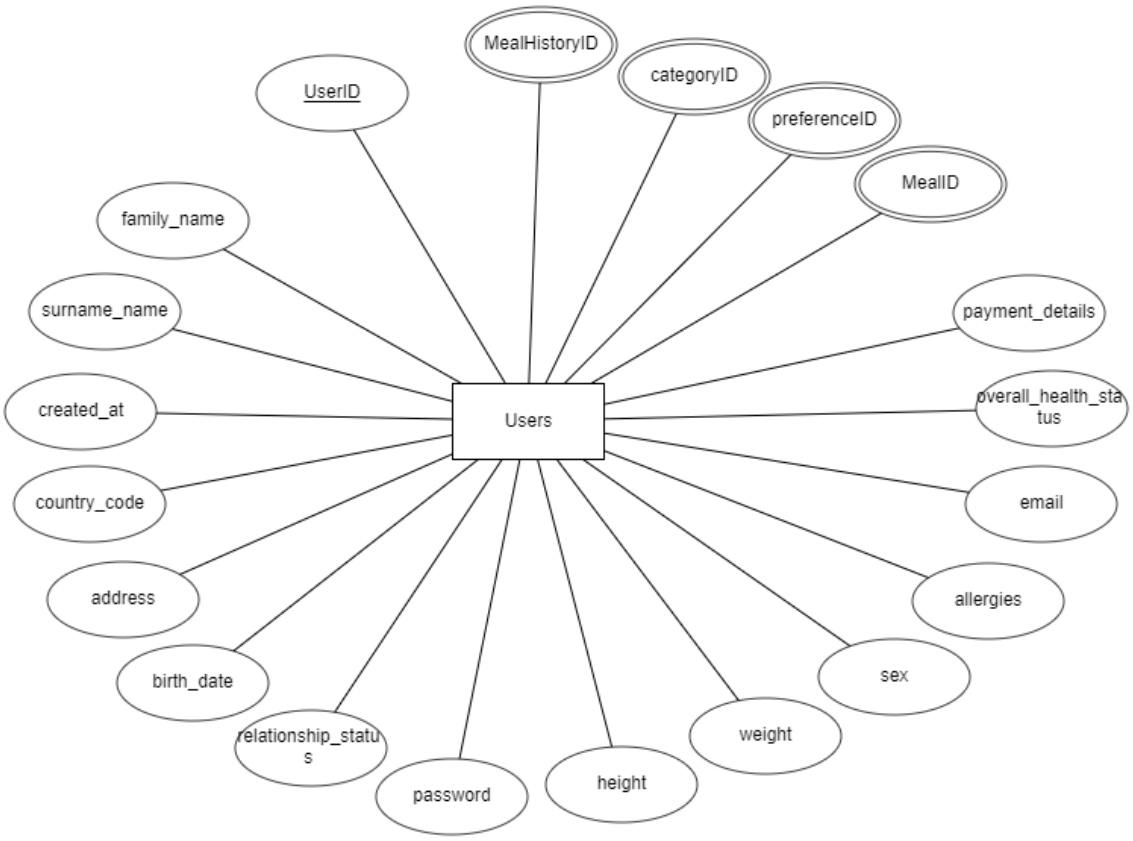

Diagram 13 : ER Diagram – User

An entity that expresses user information. The UserID attribute is the primary key, and has 15 other attributes such as name, password, address, and birthday. MealID, preferenceID, categoryID, and MealHistoryID have multiple values.

B. Ingredient

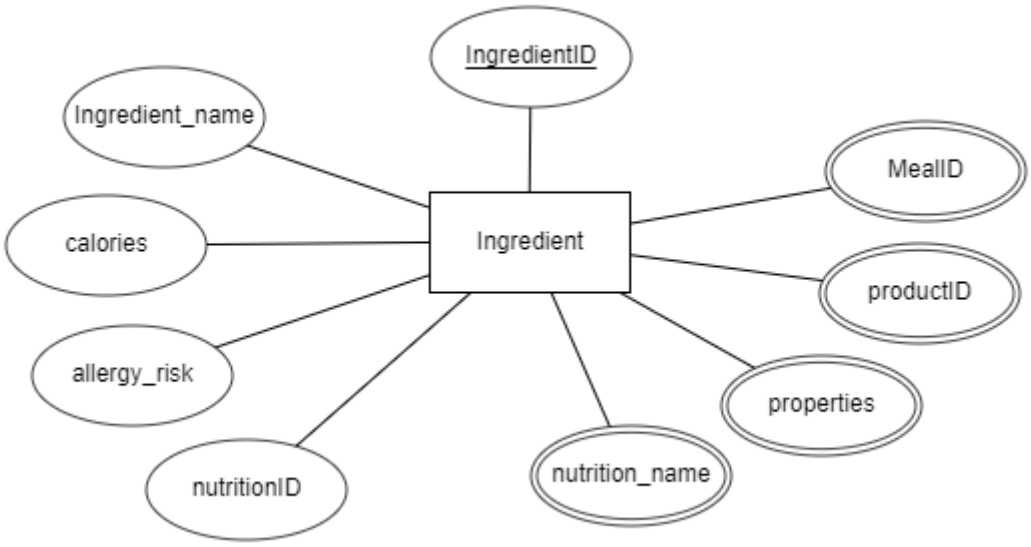

Diagram 14 : ER Diagram – Ingredient

In the case of Ingredient, it expresses information about ingredients. IngredientID is the primary key, and it also has information about the name, calories, allergies and nutrients. MealID, productID, properties, and nutrition_name have several values.

C. Meal

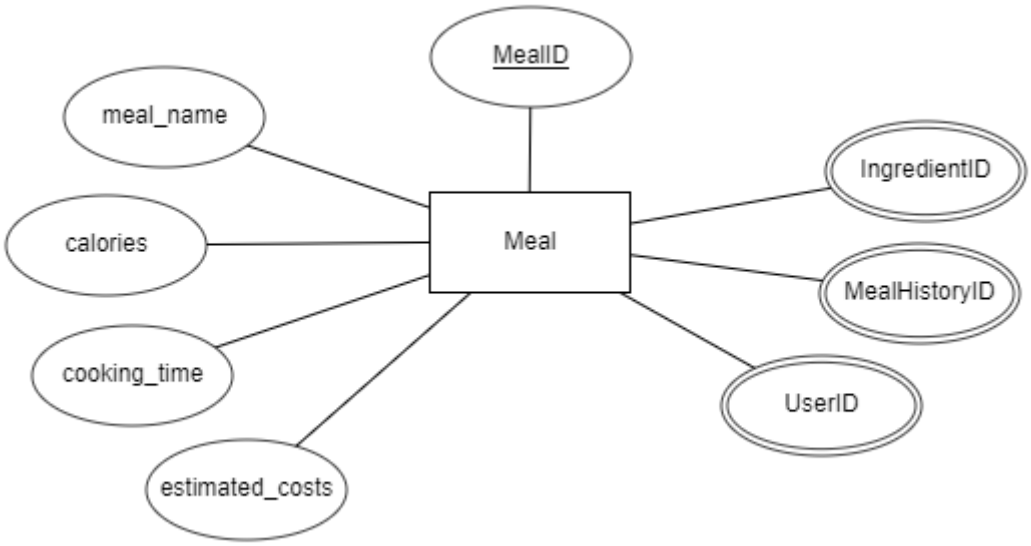

Diagram 15 : ER Diagram - Meal

In the case of Meal, information about cooking is presented. MealID is the primary key. Also, meal_name, calories, cooking_time, and estimated_costs are attributes. Ingrediient ID, MealHistoryID, and UserID have multiple values.

6.2. Relational Schema

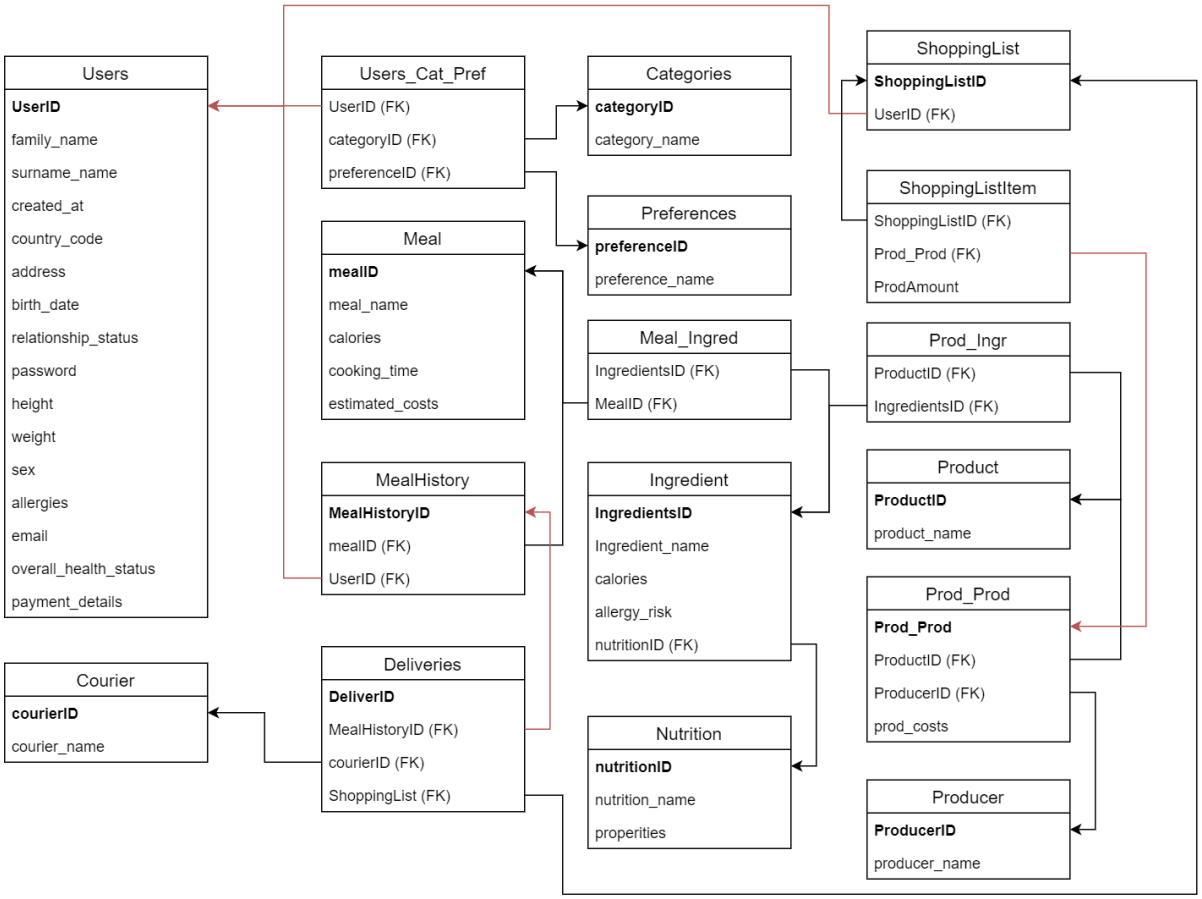

Diagram 16 : Relational Schema

6.3. SQL DDL

A. Users

```
CREATE TABLE Users
(
   UserID INT NOT NULL,
family_nameVARCHAR(10) NOT NULL,
surname_nameVARCHAR(10) NOT NULL,
created_at TIMESTAMP NOT NULL,
country_code INT NOT NULL,
   address VARCHAR(200) NOT NULL,
birth_date DATETIME NOT NULL,
relationship_statusVARCHAR(10) NOT NULL,
   password VARCHAR(20) NOT NULL,
   height DOUBLE(5,2) NOT NULL,
   weight DOUBLE(5,2) NOT NULL,
   sex VARCHAR(10) NOT NULL,
   allergies TEXT(50) NOT NULL,
   email VARCHAR(50) NOT NULL,
overall_health_statusVARCHAR(200) NOT NULL,
  payment_detailsVARCHAR(200) NOT NULL,
  PRIMARY KEY (UserID)
);
```

B. Users_Cat_Pref

```
CREATE TABLEUsers_Cat_Pref
(
UserID INT NOT NULL,
categoryID INT NOT NULL,
preferenceID INT NOT NULL,
   FOREIGN KEY (UserID) REFERENCES Users(UserID),
   FOREIGN KEY (categoryID) REFERENCES Categories(categoryID),
   FOREIGN KEY (preferenceID) REFERENCES Preferences(preferenceID)
);
```

C. Meal

```
CREATE TABLE Meal
(
   mealID INT NOT NULL,
meal_nameVARCHAR(30) NOT NULL,
   calories DOUBLE(6,2) NOT NULL,
cooking_timeVARCHAR(10) NOT NULL,
estimated_costs INT NOT NULL,
   PRIMARY KEY (mealID)
);
```

D. MealHistory

```
CREATE TABLEMealHistory
(
MealHistoryID INT NOT NULL,
mealID INT NOT NULL,
UserID INT NOT NULL,
   PRIMARY KEY (MealHistoryID),
   FOREIGN KEY (mealID) REFERENCES Meal(mealID),
   FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

E. Deliveries

```
CREATE TABLE Deliveries
(
DeliverID INT NOT NULL,
MealHistoryID INT NOT NULL,
courierID INT NOT NULL,
ShoppingListID INT NOT NULL,
   PRIMARY KEY (DeliverID),
   FOREIGN KEY (MealHistoryID) REFERENCES MealHistory(MealHistoryID),
   FOREIGN KEY (courierID) REFERENCES Courier(courierID),
   FOREIGN KEY (ShoppingListID) REFERENCES ShoppingList(ShoppingListID)
);
```

F. Categories

```
CREATE TABLE Categories
(
categoryID INT NOT NULL,
category_nameVARCHAR(20) NOT NULL,
   PRIMARY KEY (categoryID)
);
```

G. Preferences

```
CREATE TABLE Preferences
(
preferenceID INT NOT NULL,
preference_nameVARCHAR(20) NOT NULL,
   PRIMARY KEY (preferenceID)
);
```

H. Meal_Ingred

```
CREATE TABLEMeal_Ingred
(
IngredientsID INT NOT NULL,
mealID INT NOT NULL,
   FOREIGN KEY (IngredientsID) REFERENCES Ingredient(IngredientsID),
   FOREIGN KEY (mealID) REFERENCES Meal(mealID)
);
```

I. Ingredient

```
CREATE TABLE Ingredient
(
IngredientsID INT NOT NULL,
Ingredient_nameVARCHAR(20) NOT NULL,
   calories DOUBLE(6,2) NOT NULL,
allergy_riskVARCHAR(50) NOT NULL,
nutritionID INT NOT NULL,
   PRIMARY KEY (IngredientsID),
   FOREIGN KEY (nutritionID) REFERENCES Nutrition(nutritionID)
);
```

J. Nutrition

```
CREATE TABLE Nutrition
(
nutritionID INT NOT NULL,
nutrition_nameVARCHAR(20) NOT NULL,
    properties VARCHAR(50) NOT NULL,
    PRIMARY KEY (nutritionID)
);
```

K. ShoppingList

```
CREATE TABLEShoppingList
(
  ShoppingListID INT NOT NULL,
    UserID INT NOT NULL,
    PRIMARY KEY (ShoppingListID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

L. ShoppingListItem

```
CREATE TABLEShoppingList
(
  ShoppingListID INT NOT NULL,
    UserID INT NOT NULL,
    PRIMARY KEY (ShoppingListID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

M. Prod_Ingr

```
CREATE TABLEProd_Ingr
(
  ProductID INT NOT NULL,
    IngredientsID INT NOT NULL,
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (IngredientID) REFERENCES Ingredient(IngredientID)
);
```

N. Product

```
CREATE TABLE Product
(
  ProductID INT NOT NULL,
    product_nameVARCHAR(20) NOT NULL,
    PRIMARY KEY (ProductID),
);
```

O. Prod_Prod

```
CREATE TABLEProd_Prod
(
  Prod_Prod INT NOT NULL,
ProductID INT NOT NULL,
ProducerID INT NOT NULL,
prod_costs INT NOT NULL,
    PRIMARY KEY (Prod_Prod),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (ProducerID) REFERENCES Producer(ProducerID)
);
```

P. Producer

```
CREATE TABLE Producer
(
  ProducerID INT NOT NULL,
    producer_nameVARCHAR(50) NOT NULL,
    PRIMARY KEY (ProducerID),
);
```

Q. Courier

```
CREATE TABLE Courier
(
  courierID INT NOT NULL,
courier_nameVARCHAR(20) NOT NULL,
    PRIMARY KEY (courierID)
);
```

7. Testing Plan

7.1. Objectives

This chapter describes the test policy and test case for the tests being conducted to ensure that the 'Nutrition cart' system functions and its performance.

7.2. Testing Policy

In the development process of the 'Nitration cart' system, testing is carried out in three stages. It is divided into Development Testing, Release Testing, and User Testing.

A. Development Testing

This is a test performed in the development process, consisting of the following detailed testing:

1) Component Testing

It is a test that checks the normal operation of each component after developing it in component units.

2) Integrating Testing

It is a test to ensure proper operation while gradually integrating subsystems.

3) System Testing

It is a test that checks the proper functioning of the system after putting all the subsystems.

4) Acceptance Testing

It is a test to use the user's information to ensure that the user's requirements function properly in the system.

B. Release Testing

The final test of the system prior to release to the user is to ensure that all the requirements written in the Request Specification are reflected.

C. User Testing

Test whether the system operates normally in the user's actual environment.

7.3. Testing Case

A. Sign up and Login

1) Sign up

User: Enter personal information such as ID, PW and press 'Sign Up' button.

System operation: Check that data is entered according to the membership form and save it in the database.

Successful Case : Informs the user of successful membership in toast form and moves to the login screen.

Failure Case : If redundant data is entered in the database, no data is entered in the form of a form, or if the network is lost, the user is informed of the failure of membership in the form of toast.

2) Login

User: Enter personal information such as ID, PW and press 'Log in' button.

System operation: Make sure the data is entered according to the form and compare it with the data stored in the database.

Successful Case : Informs the user of successful login in the form of toast and moves to the Meal page.

Failure Case : Notifies users of failed login in the form of toast if data other than the database is entered or if the network is lost.

B. Get User information

User: After first login, enter the data on the personal information entry page and press the 'next' and 'confirm' buttons.

System operation: Check if the data entered by the user fits the form.

Successful Case:Send the entered data to the server's DB and save it.

Failure Case: Prompt the user to enter the data according to the specified form in popup format and wait for the next input.

C. Meals Detail

User: Select the menu and press one of the following buttons: ingredients, nutrition review, cooking structures, and user conditions.

System operation: Request data from server DB.

Successful Case: print the requested data on the user screen.

Failure Case: print the cause of the error (network problem, server problem, etc.) and the failure message.

8. Development Plans

8.1. Objectives

This chapter describes what technology and development environment will be used in the actual development phase and describes the development schedule and progress.

8.2. Frontend Environment

A. Ionic

Figure 6 : iconic logo

This is a front-end framework that compiles various access devices as targets and allows them to use UI styles tailored to their environment. Because developers can compile them for various environments with just one coding without developing UI components tailored to individual device environments productivity increases.

B. HTML

Figure 7 : HTML Logo

This is the dominant markup language for web pages. HTML provides a way to create structural documents with links, quotes, and other items as well as represent architectural meanings for texts such as titles, paragraphs, lists, etc. It can then be used to embed images and objects and generate interactive forms. HTML is written in the form of HTML elements, which are "tags" surrounded by angle brackets within web page content. HTML can include or recall JavaScripts that affect the behavior of HTML processing devices such as Web browsers, and scripts such as CSSs that define the appearance and placement of text and other items.

C. node. js

Figure 8 : node.js Logo

Node.js is a software platform used to develop scalable network applications (especially server-side). It utilizes JavaScript as a written language and has high throughput through non-blocking I/O and single thread event loop. The built-in HTTP server library allows web servers to operate without additional software such as Apache, which enables more control over web server behavior.

8.3. Backend Environment

A. Python

Figure 9 : python Logo

Python is an object-oriented programming language that is used to implement various Web application servers. In this system, it was used to create backend application servers because most of the team members are familiar with Python rather than Java programming.

B. Django

Figure 10 : Django Logo

Django is a free web application framework created with Python. This Web framework consists of components that help us develop websites quickly and easily. It is used because it is free open source, has active communities, and offers good reference materials and options for free and paid support.

C. SQL

Figure 11 : SQL Logo

SQL is a special purpose programming language designed to manage data in relational database management systems (RDBMS). It is designed to search and manage data in relational database management system, create and modify database schema, and manage database object access coordination. A large number of database-related programs are adopting SQL as the standard. The system will also manage the database through SQL.

8.4. Schedule

Phase	Expected	Actual
Concept & Proposal	**5/3**	**5/2**
Requirement & Design Documentation	**5/22**	**5/22**
Environment Setting	**5/29**	
Frontend Development **Meals, Shopping cart, Delivery, Settings**	**6/5**	
Backend – App server	**6/5**	
Backend – Database	**6/12**	
Backend – Other Systems	**6/12**	
System Integration and Testing	**6/17**	

Table 35 : Schedule

9. Index

9.1. Tables

9.2. Figures

9.3. Diagrams