<냉모밀>

Design Specification

소프트웨어공학개론 42분반

5조

이름	학번
김동현	**2015312044**
김주영	**2013310791**
백지연	**2016312607**
유대청	**2015312774**
채호정	**2016312915**

Index

1 Preface ..... **- 3 -**

1.1 Objective ..... - 3 -

1.2 Readership ..... - 3 -

1.3 Document Structure ..... - 3 -

2 Introduction ..... - 4 -

2.1 Objective ..... - 4 -

2.2. Applied Diagram ..... - 4 -

2.3. Applied Tool ..... - 6 -

2.4. Project Scope ..... - 8 -

3 System Architecture – Overall ..... - 9 -

3.1 Objectives ..... - 9 -

3.2 System Organization ..... - 9 -

4 System Architecture – Frontend ..... - 12 -

4.1 Objective ..... - 12 -

4.2 Subcomponents ..... - 12 -

5 System Architecture – Backend ..... - 20 -

5.1 Objectives ..... - 20 -

5.2 Overall Architecture ..... - 20 -

5.3 Subcomponents ..... - 21 -

6 Protocol Design ..... - 24 -

6.1 Objectives ..... - 24 -

6.2 REST API ..... - 24 -

6.3 JSON ..... - 25 -

6.4 Details ..... - 25 -

7 Database Design ..... - 32 -

7.1 Objectives ..... - 32 -

7.2 ER-Diagram ..... - 32 -

7.3 Relational Schema ..... - 37 -

7.4 SQL DDL ..... - 37 -

8 Testing Plan ..... - 41 -

8.1 Objectives ..... - 41 -

8.2 Testing Policy ..... - 41 -

8.3 Test Case ..... - 42 -

9 Development Plan ..... - 46 -

9.1 Objectives ..... - 46 -

9.2 Frontend Environment ..... - 46 -

9.3 Backend Environment ..... - 47 -

9.4 Version Management ..... - 49 -

9.5 Schedule ..... - 49 -

10 Figure Index ..... - 46 -

1 Preface

1.1 Objective

Preface에서는 본 문서의 독자를 정의하고, 문서의 목차를 간략하게 소개한다.

1.2 Readership

본 문서의 독자는 시스템의 설계와 개발, 유지 보수에 참여하는 모든 구성원으로 정의한다. Design 명세서를 통해 냉모밀과 외부 시스템, 그리고 냉모밀 내부 구조 간의 interaction 등 최종 시스템에 대한 대략적인 개요를 제공받을 수 있다.

1.3 Document Structure

1.3.1 Preface

본 문서의 독자를 정의하고, 문서의 목차를 간략하게 소개한다.

1.3.2 Introduction

시스템 설계 시 사용된 다양한 다이어그램과 표현 도구들에 대해 설명하고 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

1.3.3 System Architecture

본 시스템의 전반적인 구조를 Overall, 프론트엔드 어플리케이션, 백엔드 어플리케이션으로 나누어 설명한다. 시스템과 각 서브시스템의 구조를 개괄적으로 기술하고, 시스템의 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당되었는지 설명한다.

1.3.4 Protocol Design

시스템의 각 컴포넌트, 특히 프론트엔드 어플리케이션과 백엔드 어플리케이션 간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

1.3.5 Database Design

Requirement specification에서 기술한 데이터베이스 요구사항을 바탕으로 Database Design을 작성한다. 각 데이터 엔티티의 속성과 관계를 나타내는 ER diagram과 Relational Schema의

단계를 거쳐 SQL DDL을 작성한다.

1.3.6 Testing Plan

미리 작성된 Test를 이용해, Verification 과 Validation을 시행한다. 이 Test 작성에 대한 계획을 설명한다.

1.3.7 Development Plan

시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.

1.3.8 Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2 Introduction

2.1 Objective

이번 챕터에서는 본 시스템의 설계에 사용된 다양한 다이어그램과 도구를 소개하고, 본 시스템의 개발 범위를 기술한다.

2.2. Applied Diagram

2.1.1 UML

Figure 1: Unified Modeling Language(UML)

UML은 과거 개별적으로 존재하던 소프트웨어 설계 기법들을 하나로 모아 만든 통합된 소프트웨어 모델링 기법으로, 현재 객체 지향 소프트웨어 시스템을 개발하는 과정에서 각

단계의 결과를 도식적으로 시각화하고 문서화하기 위해 사용된다. 뿐만 아니라 시스템에 대한 시각적 정보를 제 공함으로써 시스템 개발 과정에서 요구되는 사용자와 개발자 간의 의사소통에 크게 기여하며, 13 개의 모델링 기법을 제공하기 때문에 프로젝트의 종류에 제한되지 않는다는 장점을 갖는다. 또, 각 모델이 사용하는 표기법은 해당 심벌(symbol) 마다 명확하고 고유한 정의를 가지고 있어 이를 이용하는 개발자들 간에 오류 없는 원활한 의사소통이 가능하다.

UML에서 제공하는 다이어그램(Diagram)의 종류는 총 13개로, 시스템의 구조(structure)를 도식화하는 Class Diagram, Object Diagram, Deployment Diagram, Composite Structure Diagram, Component Diagram, Package Diagram 등 6개, 시스템의 전개(flow)방식을 도식화하는 Activity Diagram, State Machine Diagram, Use Case Diagram 등 3개, 구성 요소 (components) 간의 상호작용(interaction) 방식을 표현하는 Communication Diagram, Sequence Diagram, Timing Diagram, Interaction Overview Diagram 등 4개다이어그램으로 구성된다.

2.1.2 Class Diagram

Class Diagram은 클래스 내부의 정적 요소나 클래스 간의 관계를 나타내는 다이어그램으로 시스템의 부분 혹은 전체 구조를 파악하는데 용이하다는 장점을 지닌다. Class Diagram은 각 클래스 사이의 의존 관계를 명확히 제시한다는 장점도 있다. 또한, 클래스 간의 links, 클래스 내의 attributes와 methods를 명시하는 필드도 가지고 있다.

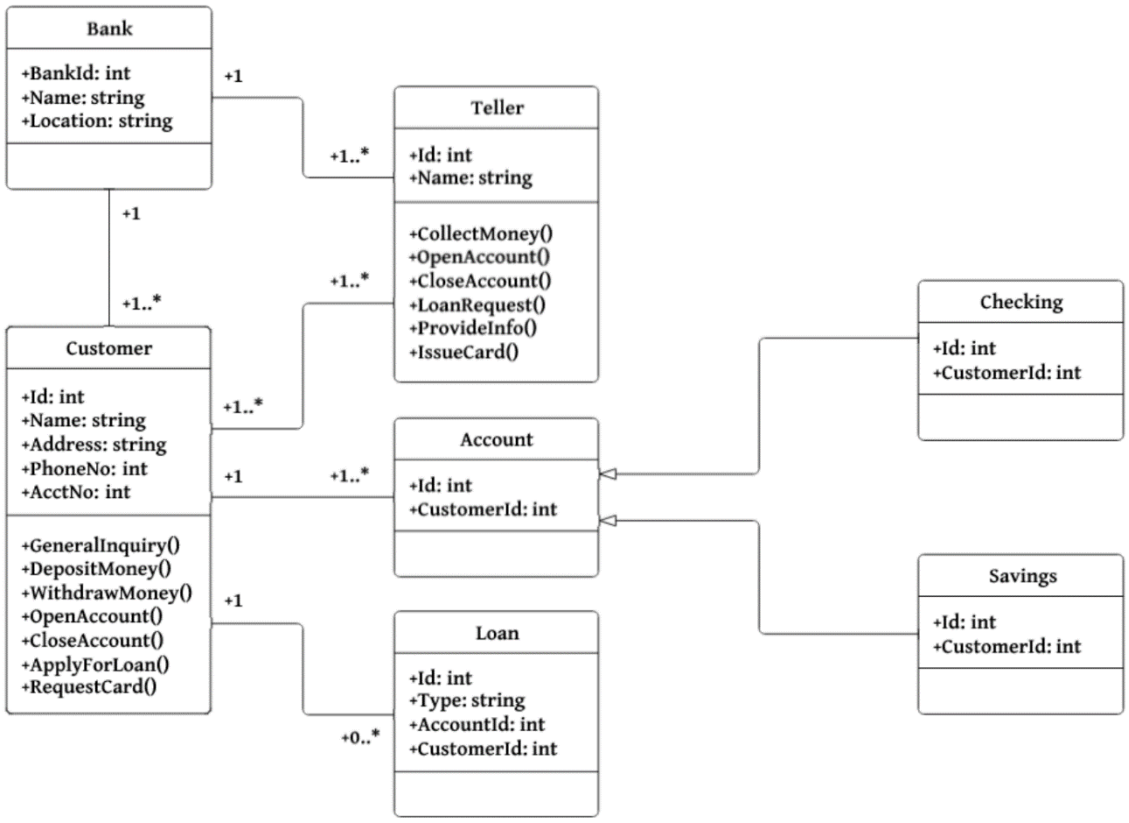

Figure 2: Example of Class Diagram

2.1.3 Sequence Diagram

Sequence Diagram의 목적은 어떤 결과를 만들어내는 사건(event)의 시퀀스(sequence)를 정의하는 것이다. 즉, 어떤 메시지가 발생했는가 보다는 해당 메시지가 발생하는 과정 혹은 순서에 집중한다. 대부분의 Sequence Diagram은 시스템 내부 객체들 간에 어떤 메시지들이 전달되는지, 어떤 순서로 전달되는지를 수직 축(axis)과 수평 축을 기준으로 표기한다. 수직 축에서는 메시지가 발생한 시간 순서를 축의 상단에서 하단으로 진행하는 방향으로 나타내며, 수평 축에서는 화살표의 방향에 따라 메시지의 송신객체와 수신객체를 표기한다.

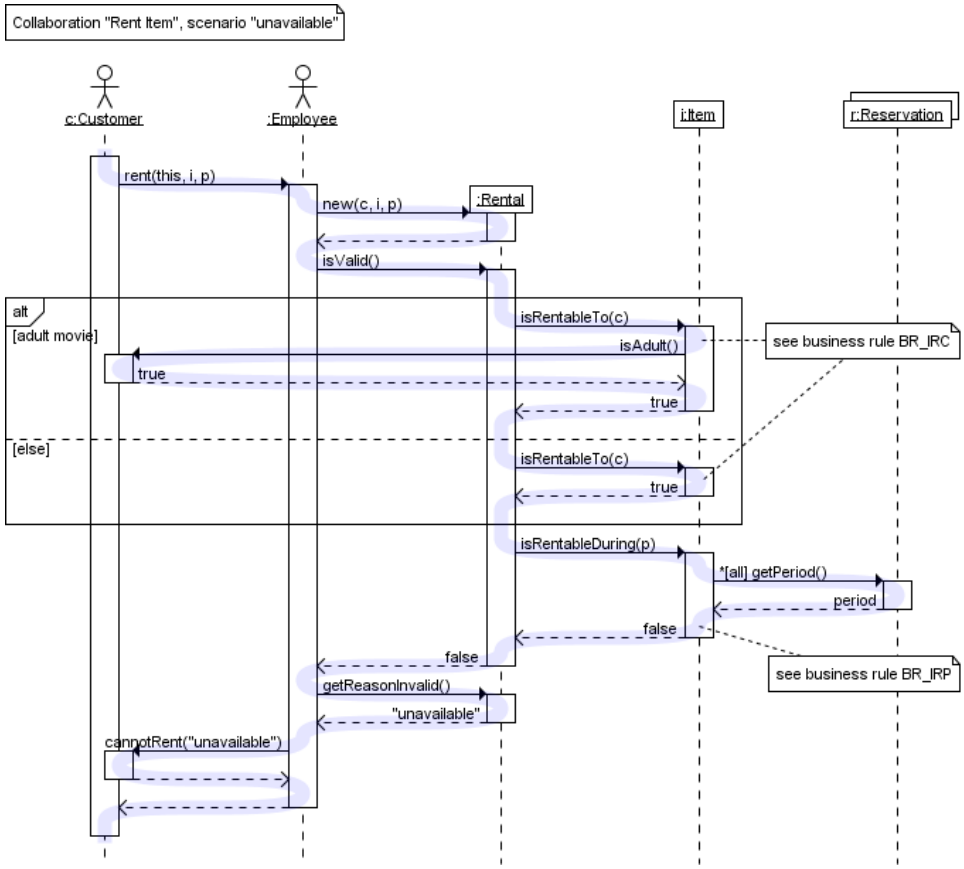

Figure 3: Example of Sequence Diagram

2.3. Applied Tool

2.1.4 Draw.io

Figure 4: draw.io Logo

Draw.io는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 직접

다이어그램에 사용하기 위해 도형을 만들 필요가 없다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있기 때문에 도형을 정렬하기 편리하다. 이 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

2.1.5 PowerPoint

Figure 5: PowerPoint Logo

Powerpoint는 그래픽 프레젠테이션 툴이다. 주로 발표용으로 사용되지만 내장된 도형 작성 기능이 매우 강력하기 때문에 draw.io에서 만들기 힘든 복잡한 다이어그램을 작성하기 위해 사용하였다.

2.1.6 ERDPlus

Figure 6: ERDPlus Logo

ERDPlus는 ER Diagram을 간단한 버튼 클릭으로 생성할 수 있게 해 주는 온라인 툴이다. 적은 노력으로 ER Diagram을 draw.io에 비해 간단하게 만들 수 있기 때문에 ER Diagram을 작성하는 데 사용하였다.

2.4. Project Scope

냉모밀은 지금 냉장고에 있는 재료를 이용해서 레시피를 추천 받고, 혹시 더 필요한 재료가 있다면 한 번에 구매를 할 수 있는 이커머스 시스템이다. 냉장고에 있는 재료들 중에서 현 레시피 추천에 포함할 것, 포함하지 않을 것을 선택하면 포함하는 재료는 포함하고 불포함 재료는 포함하지 않는 레시피들을 추천해준다. 사용자들은 레시피를 직접 등록할 수도 있고 레시피를 추천/검색 받을 수 있고 레시피에 좋아요 버튼을 이용하여 credit을 줄 수도 있다.

냉모밀은 크게 Join/Login, 냉장고 재료 관리, 재료 구매, 레시피 등록, 레시피 추천/검색의 5개의 시스템으로 구성되어 있다. 프론트엔드는 안드로이드 어플리케이션이고 백엔드 서버와 JSON으로 통신한다. 백엔드 서버는 프론트엔드에서 받은 요청을 처리하는 각 Controller가 있고 DB에서 원하는 정보를 검색하거나 새로운 데이터를 추가하는 Database Controller가 있다. 백엔드 어플리케이션에 Recipe Recommendation System에서 앞서 언급한 냉장고 재료 기반 레시피 추천이 이루어 진다.

3 System Architecture – Overall

3.1 Objectives

이번 챕터에서는 본 시스템의 전체적인 구조를 설명한다. 시스템 전체의 구조와 각 서브시스템의 개략적인 구조, 서브시스템 간의 관계를 서술하며 각 구조는 다이어그램을 첨부하여 이해를 돕는다.

3.2 System Organization

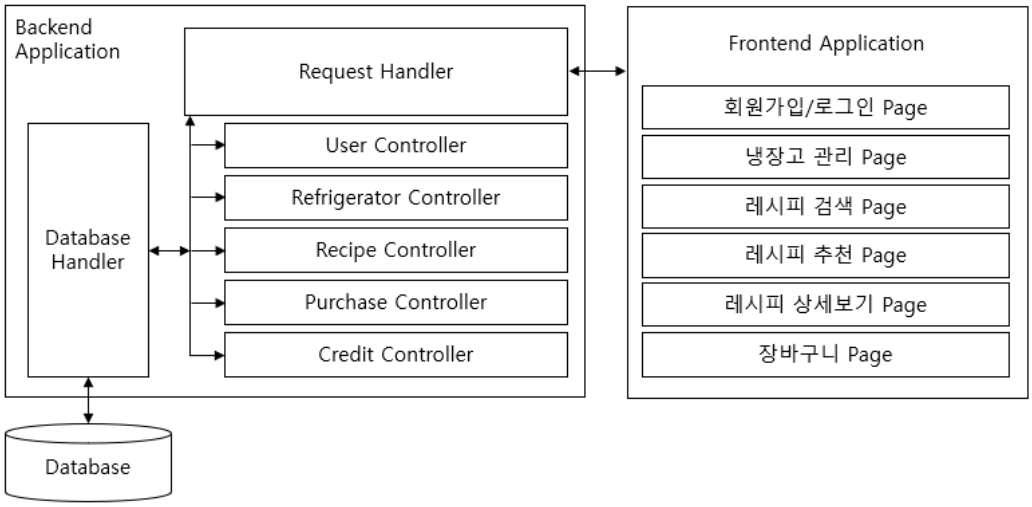

Figure 7: Overall System Organization

본 서비스는 프론트엔드 어플리케이션이 사용자와의 모든 상호작용을 맡고, 프론트엔드 어플리케이션과 백엔드 어플리케이션은 JSON을 기반으로 한 HTTP 통신으로 데이터를 송수신한다. 백엔드 어플리케이션은 프론트엔드로부터 들어오는 각 요청을 컨트롤러로 분배하고, 필요한 객체 정보를 데이터베이스로부터 가져와 JSON 포맷으로 가공한 뒤 전달한다.

프론트엔드에는 사용자가 어플리케이션을 이용하는 Page들이 있고 관리자가 접근하는 Page가 있다. 관리자 Page는 사용자들이 레시피 등록 Page에서 등록한 레시피들을 검수하는 작업을 한다. 새로운 레시피가 등록될 만한 퀄리티를 갖추고 있는지 확인하고 DB에서 관리하기 쉽게 메뉴 카테고리를 설정한다. 검수 작업을 마친 레시피를 백엔드에 Database Controller에게 전달하여 최종적으로 DB에 저장한다.

사용자가 각각 포함하는 재료와 포함하지 않는 재료를 선택하고 레시피 추천을 요청하면 Recipe Recommendation System은 해당 재료들을 바탕으로 Recipe DB를 검색하여 적절한 추천 레시피들을 프론트엔드에 제공한다.

3.2.1 Frontend Application

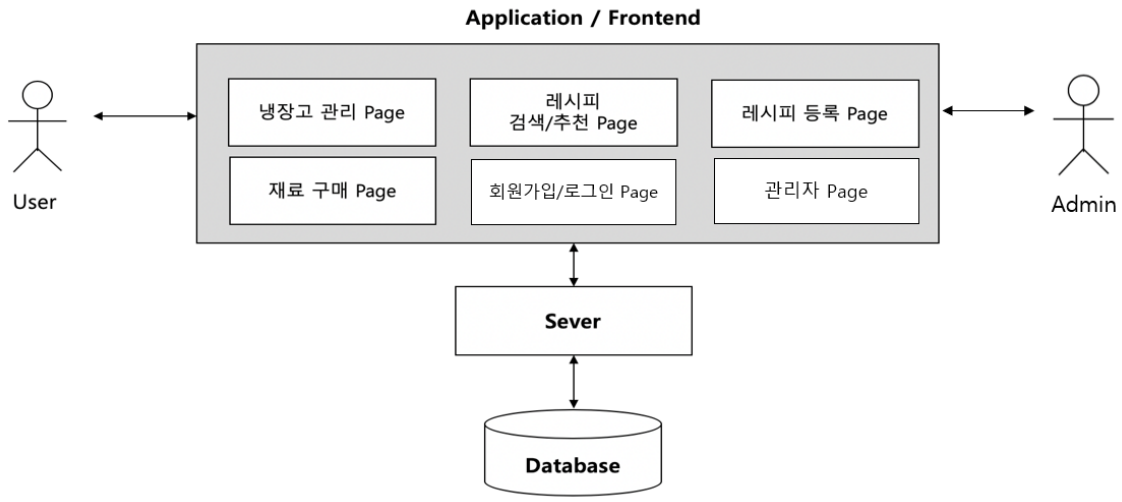

Figure 8: System Architecture – Frontend

사용자와의 상호작용을 전담하는 시스템으로 안드로이드 어플리케이션이다. 회원가입/로그인 Page, 냉장고 관리 Page, 재료 구매 Page, 레시피 검색/추천 Page, 레시피 등록 Page는 본 시스템의 사용자가 이용한 컴포넌트들이다. 관리자는 프론트엔드에 관리자 Page에서 사용자가 등록한 레시피의 검수를 진행한다. 검수에 통과한 레시피들은 백엔드에 Database Handler를 통해 DB에 최종 저장된다. 각 컴포넌트가 백엔드와의 통신을 전담하는 Request Handler에게 필요한 정보나 서비스를 요청하면, Request Handler는 미리 정해 둔 적절한 프로토콜로 요청을 변형해 백엔드에 전달하게 된다.

3.2.2 Backend Application

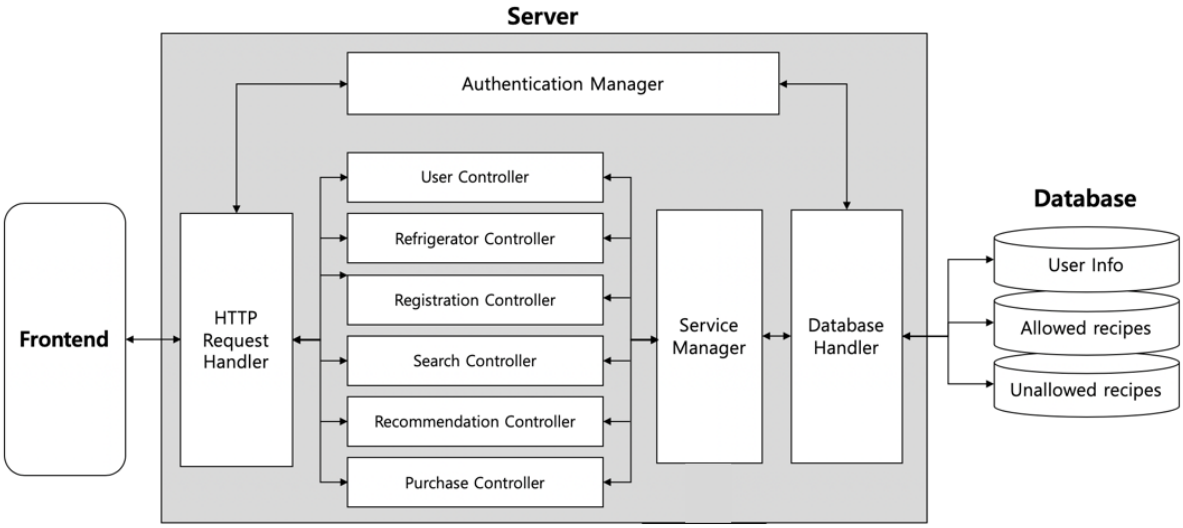

Figure 9: System Architecture – Backend

프론트엔드에서 전달받은 요청들을 알맞은 Controller에 보내 처리한다. Database Handler는 DB에 query를 보내 원하는 데이터를 검색, 추가, 수정, 삭제 등을 하고 DB로부터 받은 결과를 다시 프론트엔드에 보낼 수 있게 JSON 형태로 바꾸어 준다. Authentication Manager는 회원가입/로그인 요청이 왔을 때 해당 요청을 accept 할 수 있는지 없는지 확인한 후, 필요하다면 Database Handler를 통해 User DB에 반영한다.

4 System Architecture – Frontend

4.1 Objective

전체 시스템 아키텍처 중 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트 간의 관계를 서술한다.

4.2 Subcomponents

4.2.1 Join/Login

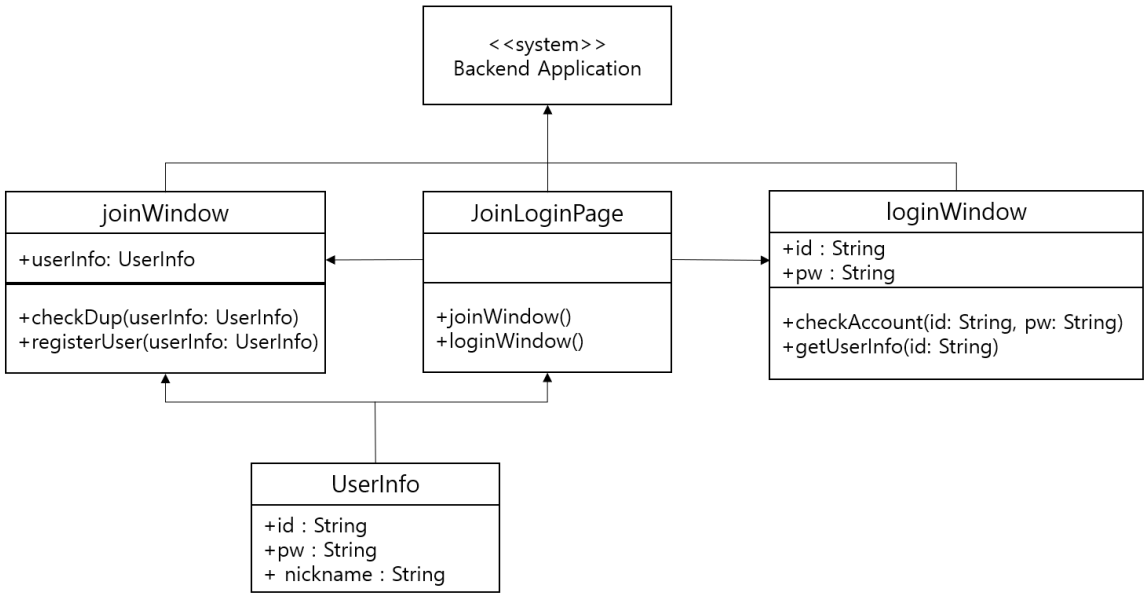

Figure 10: System Architecture – Frontend – Join/Login

1. JoinLoginPage – 회원가입 또는 로그인 선택하는 페이지
   A. attributes
   B. methods
      + joinWindow( ): 회원 가입 절차를 위한 화면을 띄워준다.
      + LoginWindow( ): 로그인 절차를 위한 화면을 띄워준다.
2. joinWindow
   A. attributes
      +userInfo: 회원 가입할 회원 정보
   B. methods
      +checkDup(userInfo: UserInfo): id와 nickname의 중복 확인을 한다.
      +registerUser(userInfo: UserInfo): 해당 회원 정보로 회원 가입을 한다.

3. loginWindow

A. attributes

+id: 입력 받은 로그인 할 id

+pw: 입력 받은 위 id에 대한 pw

B. methods

+checkAccount(id: String, pw: String): id와 pw가 일치하는지 확인한다.

+getUserInfo(id: String): 해당 id의 회원 정보를 받아온다.

4.2.2 Refrigerator Management

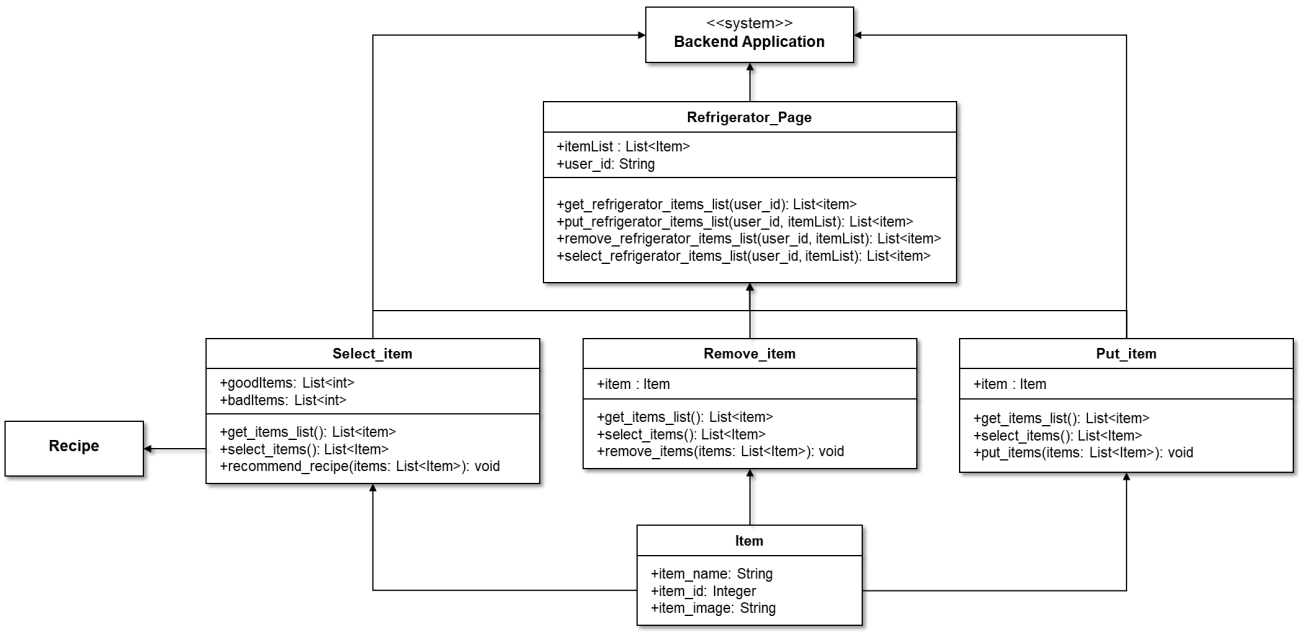

Figure 11: System Architecture – Frontend – RefrigeratorPage

1. Refrigerator_Page – user가 냉장고 재료를 관리하는 화면 객체

A. attributes

+item: 재료의 정보를 저장하는 재료 객체

+user_id : 사용자 아이디를 저장하는 객체

B. methods

+get_refrigerator_items_list(user_id): user의 냉장고에 있는 재료 목록을 Backend 에서

가져온다.

+put_refrigerator_items_list(user_id, itemList): user의 냉장고에 넣을 재료 리스트를 선택하여 서버에 입력을 요청한다.

+remove_refrigerator_items_list(user_id, itemList): List<item>

user의 냉장고에서 제거할 재료 리스트를 선택하여 서버에 삭제를 요청한다.

+select_refrigerator_items_list(user_id, itemList): List<item> user의 냉장고에 있는 재료 중에서 good, bad를 선택하여 레시피 추천 요청을 한다.

2. Put_item – 재료 입력 객체

A. attributes

+item: 재료의 정보를 저장하는 재료 객체

B. methods

+get_items_list(): Backend 로부터 item의 리스트를 불러온다.

+select_items(): 화면에 표시된 item 리스트에서 사용자가 선택한 item 들의 리스트를 반환한다.

+put_items(items: List<Item>): 냉장고에 입력할 item 리스트를 Backend로 보낸다.

3. Remove_item – 재료 입력 객체

A. attributes

+item: 재료의 정보를 저장하는 재료 객체

B. methods

+get_items_list(): Backend 로부터 사용자의 냉장고에 있는 item의 리스트를 불러온다.

+select_items(): 화면에 표시된 item 리스트에서 사용자가 선택한 item 들의 리스트를 반환한다.

+remove_items(items: List<Item>): 냉장고에서 제거할 item 리스트를 Backend로 보낸다.

4. Select_item – 재료 입력 객체

A. attributes

+goodItems: 레시피 추천시 포함되기를 원하는 item을 선택한 리스트

+badItems: 레시피 추천시 제외되기를 원하는 item을 선택한 리스트

B. methods

+get_items_list(): Backend 로부터 사용자의 냉장고에 있는 item의 리스트를 불러온다.

+select_items(): 화면에 표시된 item 리스트에서 사용자가 선택한 item 들의 리스트를 반환한다.

+recommend_recipe(items: List<Item>): 선택된 item으로 레시피를 추천받는 페이지로 이동한다.

5. Item – 재료 객체

A. attributes

+item_name: 재료의 이름

+item_id: 재료의 id

+item_num: 재료 개수

4.2.3 Overall Recipe Recommendation & Recipe Searching

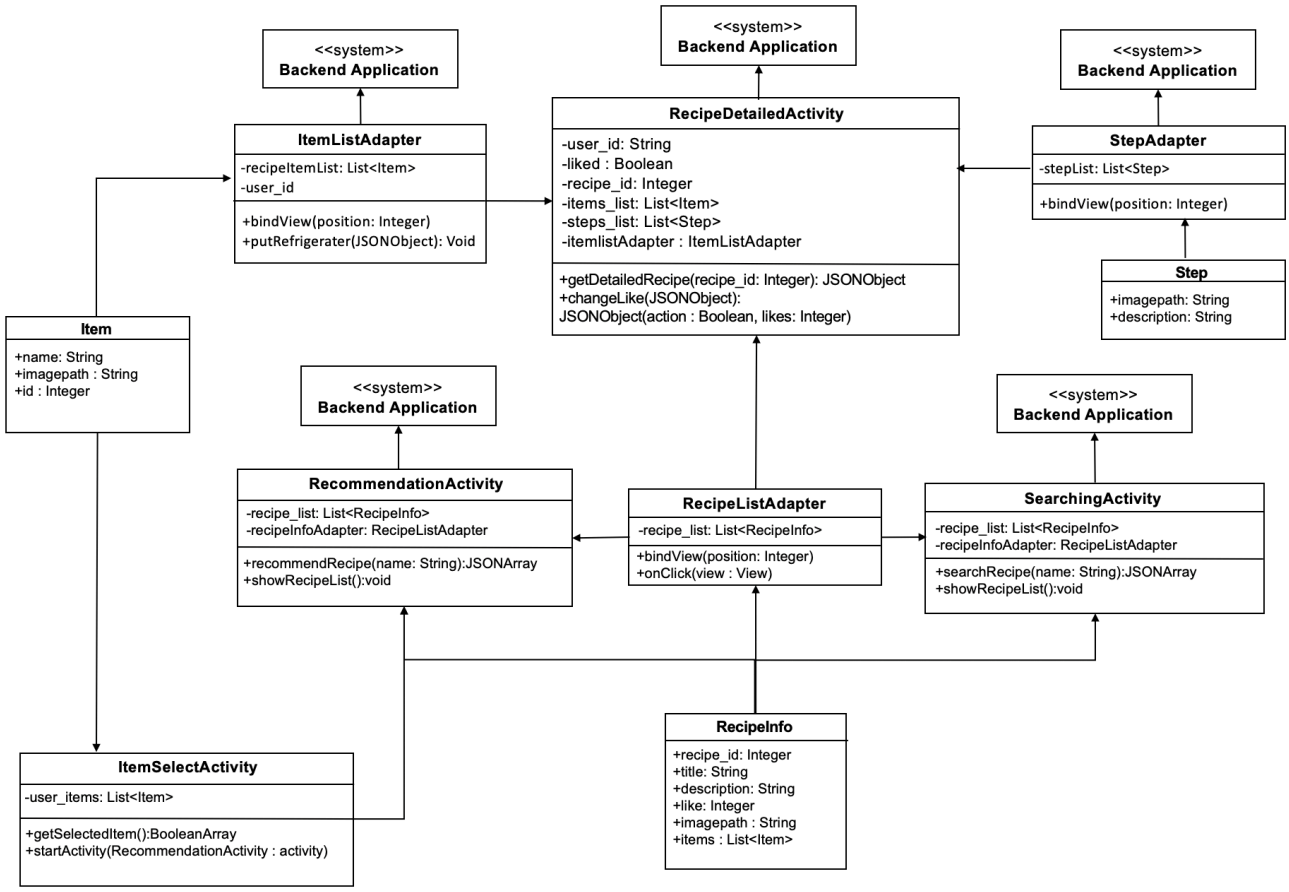

Figure 12: System Architecture – Frontend – Overall Recipe Recommendation & Recipe Searching

1. Item – 아이템 객체
   A. attributes

      +name: 아이템의 이름

      +imagepath : 아이템 이미지 리소스

      +id: 아이템 아이디

2. ItemSelectActivity – 추천아이템 선택 객체
   B. attributes

      +user_items: 사용자의 아이템이 담기는 리스트

   B. methods

      +getSelectedItem(): 유저가 선택한 아이템의 정보를 받는다

      +startActivity(src: String) 유저가 선택한 아이템의 정보를 Recommendation activity로

넘겨준다.

3. RecipeInfo – 리스트로 보이는 레시피 객체
   A. attributes

   +recipe_id: 레시피 아이디

   +title: 레시피 이름

   +description: 레시피 소개

   +like: 레시피 추천 수

   +imagepath: 레시피 메인 이미지 주소

   +items: 레시피에 필요한 아이템이 담기는 리스트

4. RecommendationActivity – 추천 액티비티 객체
   A. attributes

   -recipe_list : RecipeInfo 리스트

   -recipeInfoAdapter : RecipeInfo를 리스트에 붙여주는 adapter

   B. method

   +recommendRecipe(name: String): 추천받고 싶은 아이템과 아닌 아이템을 문자열로 보내어 JSON 객체를 받는 함수

   +showRecipeList(): 변경된 레시피 리스트를 보여주는 함수

5. SearchingActivity – 검색 액티비티 객체
   A. attributes

   -recipe_list : RecipeInfo 리스트

   -recipeInfoAdapter : RecipeInfo를 리스트에 붙여주는 adapter

   B. method

   +recommendRecipe(name: String): 추천받고 싶은 아이템과 아닌 아이템을 문자열로 보내어 JSON 객체를 받는 함수

   +showRecipeList(): 변경된 레시피 리스트를 보여주는 함수

6. RecipeListAdapter – 레시피 리스트를 보여주는 adapter
   A. attributes

-recipe_list : RecipeInfo 리스트

B. method

+bindView(position: Integer): list의 position에 해당하는 recipeinfo를 뷰에 매핑한다.

+onClick(view View): 클릭시 RecipeInfo객체의 상세보기 페이지로 이동한다.

7. RecipeDetailedActivity – 레시피 상세정보 액티비티

A. attributes

-user_id: 현재 보고 있는 사용자 id
-liked : 해당 사용자가 추천을 누른 레시피인지
-recipe_id: 레시피 아이디
-items_list: Item 객체 리스트
-steps_list: Step 객체 리스트
-itemlistAdapter : Item 을 리스트에 붙여주는 adapter

B. method

+getDetailedRecipe(recipe_id: Integer): 서버로부터 recipe id 로 상세 정보 jsonobject 를 가져온다

+changeLike: 사용자가 추천을 누른 경우 추천 수를 업데이트한다.

8. ItemListAdapter – 레시피 리스트를 보여주는 adapter

A. attributes

-recipeItemList: Item 객체 리스트
-user_id: 사용자의 아이디

B. method

+bindView(position: Integer): list의 position에 해당하는 item을 뷰에 매핑한다.

+putRefrigerator(JSONObject): item 클릭 시, 사용자의 냉장고로 아이템을 넣는다.

9. StepListAdapter – 레시피의 과정을 보여주는 adapter

A. attributes

-stepList: Step 객체 리스트

B. method

+bindView(position: Integer): list의 position에 해당하는 step을 뷰에 매핑한다.

10. Step – 레시피 과정 객체

C. attributes

-imagepath: 과정 이미지 주소

-description: 과정 설명 문자열

4.2.4 Basket Page

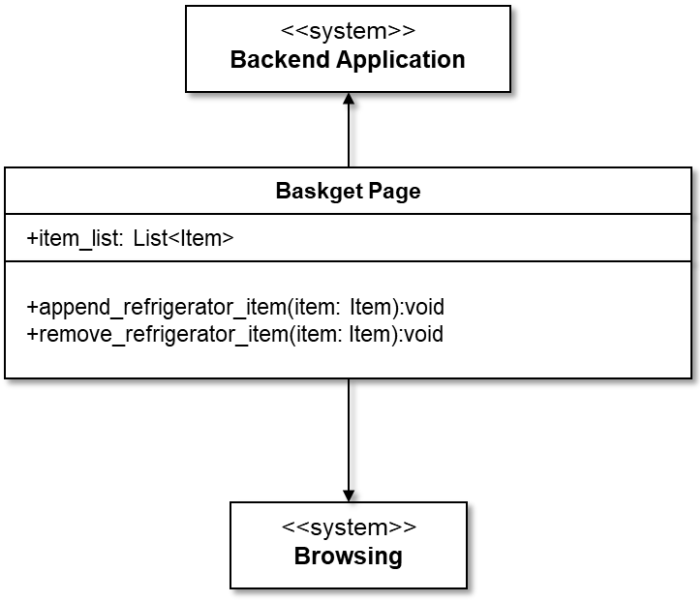

Figure 16: System Architecture – Frontend – Basket

1. Purchase – 구매 객체

A. attributes

+item_list: 장바구니에 있는 재료 리스트

B. methods

+append_refrigerator_items_list(products: List<Product>): 재료 리스트에서 item id와 num을 읽어서 사용자 냉장고에 반영한다.

+remove_refrigerator_items_list(products: List<Product>): 재료 리스트에서 제거한다.

5 System Architecture – Backend

5.1 Objectives

이 챕터에서는 전체 시스템 중 사용자와 상호작용을 하는 프론트 엔드를 제외한 백 엔드 시스템과 각 서브시스템의 구조에 대해 설명한다.

5.2 Overall Architecture

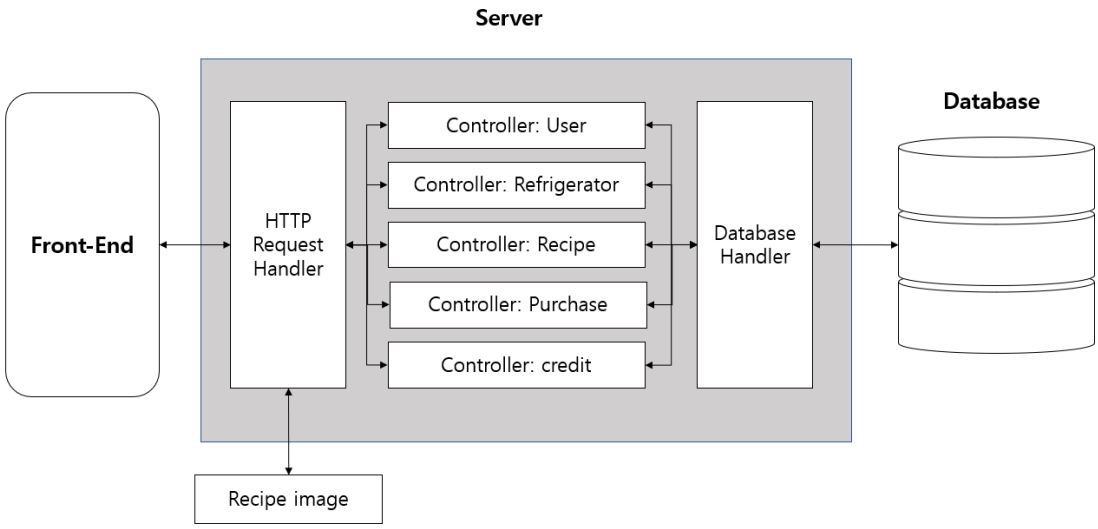

Figure 18: Overall Backend Architecture

백엔드 시스템의 전체적인 구조는 위의 그림과 같다.

Front-End와 직접적인 통신을 하는 Server와 각종 데이터를 저장하는 Database로 이루어져 있다. 각 HTTP 통신 url과 방법에 따라 User, Refrigerator, Recipe, Purchase, Credit Controller가 나누어서 request를 처리한다.

5.3 Subcomponents

5.3.1 Server

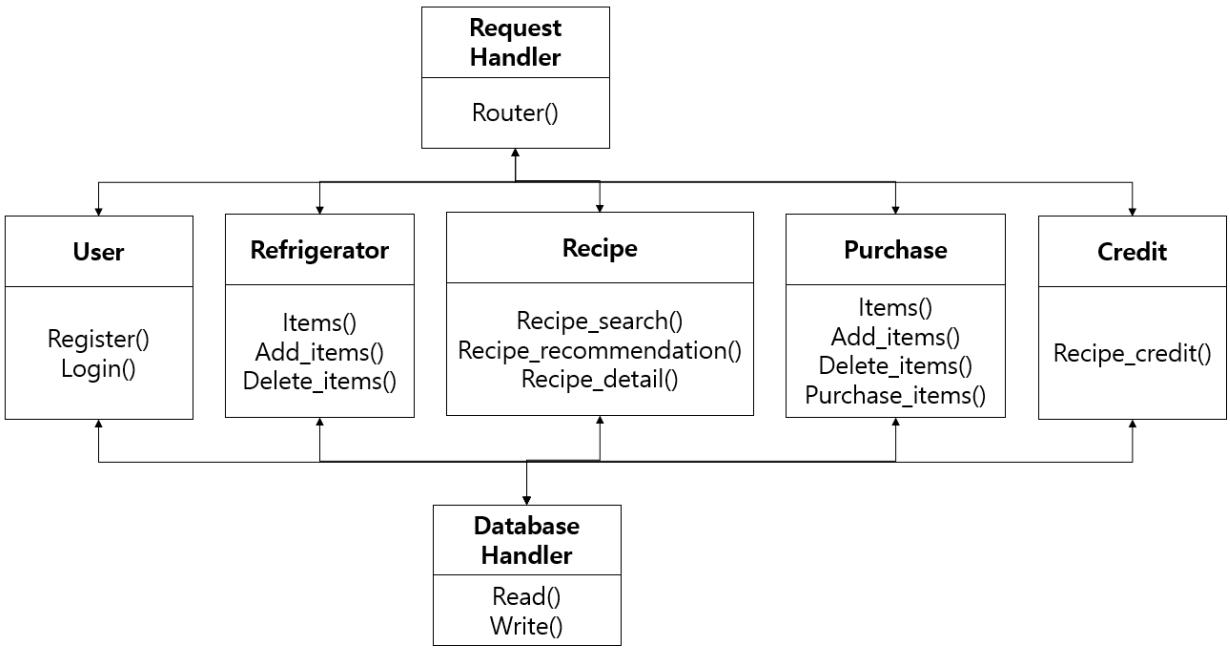

Figure 19: Backend-Server

1. Request Handler

프론트엔드에서 오는 요청을 처리한다.

A. Router()

Node.js에 express router를 이용하여 서버로 들어온 요청을 적절한 controller에 분배해서 요청을 처리한다.

2. User Controller

사용자 정보 처리와 관련된 부분을 조정한다.

A. User_id: string

사용자가 가입 할 때 사용하는 아이디 정보

B. User_password: string

사용자가 가입 할 때 사용하는 비밀번호 정보

C. register(): 회원가입을 처리하는 function

D. login(): 로그인을 처리하는 function

3. Refrigerator Controller

사용자의 냉장고 정보를 관리한다.

A. Items: JSON Array

사용자 냉장고에 존재하는 재료 정보이다.

B. add_items(재료: JSON Array)

사용자가 냉장고에 추가하고자 하는 재료에 대해서 데이터베이스에 추가할 수 있도록 한다.

C. remove_items(재료: JSON Array)

사용자가 냉장고에서 제거하고자 하는 재료에 대해서 데이터베이스에서 삭제할 수 있도록 한다.

4. Recipe Controller

레시피에 관련된 검색, 추천, 상세보기 요청을 처리한다.

A. recipe_search(키워드: String)

사용자가 검색하고자 하는 레시피 이름을 포함하는 레시피들을 보내준다.

B. recipe_recommendation(good: 재료ID List, bad: 재료ID)

메인 재료에 good list에 있는 재료가 포함되어 있고, 전체 재료에 bad list에 있는 재료가 없는 레시피들을 1차적으로 선별한다. 만일, 1차 선별에 해당하는 레시피가 없다면 전체 재료에 good list에 있는 재료가 포함된 레시피들을 선별한다. 선별된 레시피의 간단 요약 정보(레시피 ID, 레시피 이름, 메인 재료, 메인 사진, 메인 설명, 좋아요 개수)을 보내준다.

C. recipe_detail(username: 사용자 ID, recipe_id: int)

상세보기를 원하는 레시피의 ID를 받아 상세정보(좋아요 개수, 스텝 별 설명과 사진, 사용자가 해당 레시피에 좋아요를 눌렀는지 안 눌렀는지)를 보내준다.

5. Purchase Controller

사용자의 장바구니 정보를 관리한다. 장바구니 조회, 추가, 삭제, 구매처리의 요청을 처리한다.

A. items(username: 사용자 ID)

사용자의 장바구니에 담겨있는 재료들의 ID list를 보내준다.

B. add_items(username: 사용자 ID, items: 재료 ID List)

해당 사용자의 장바구니에 items에 있는 재료들을 추가한다.

C. delete_items(username: 사용자 ID, items: 재료 ID List)

해당 사용자의 장바구니에서 items에 있는 재료들을 삭제한다.

D. purchase_items(username: 사용자 ID, items: 재료 ID List)

해당 사용자의 장바구니에서 items에 있는 재료들을 삭제하고, 그 재료들을 사용자의 냉장고에 추가한다.

6. Credit Controller

사용자가 레시피에 '좋아요' 버튼을 누르는 요청을 처리한다.

A. recipe_credit(username: 사용자 ID, recipe_id: 레시피 ID, like: 호/불호)

like가 1이면 좋아요를 누르는 액션임을 나타낸다. 좋아요를 누르는 요청이면 recipe_credit 테이블에 user_id와 recipe_id를 추가하고 해당 레시피의 credit을 1 증가시킨다. 그리고 사용자에게 새로 update된 좋아요 개수를 보내준다.

Like가 0이면 좋아요를 취소하는 액션이다. 좋아요를 취소하는 요청이면 recipe_credit 테이블에서 해당 user_id와 recipe_id를 삭제하고 해당 레시피의 credit을 1 감소시킨다. 그리고 사용자에게 새로 update된 좋아요 개수를 보내준다.

6 Protocol Design

6.1 Objectives

이번 챕터에서는 각 서브시스템 간의 상호작용, 특히 프론트엔드 시스템과 백엔드 애플리케이션 서버 시스템간 상호작용에 이용되는 프로토콜에 어떤 구조가 사용되는지 설명하고, 각 인터페이스가 어떻게 정의되어 있는지를 기술한다.

6.2 REST API

본 시스템은 프론트엔드와 백엔드 사이의 통신에 웹 인터페이스, 즉 HTTP 를 이용하며,요청과 응답 형식은 REST API 에 따른다. REST API 란 Representational State Transfer 의 약자로서, 서버에 저장되어 있는 각 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 API 의 설계 형식을 의미한다. REST API 는 크게 다음 세 부분으로 구성되어 있다.

6.2.1 자원(Resource): URI

서버가 보관하고 있는 데이터를 나타내며, 각 자원은 고유한 URI 를 가진다.

6.2.2 행위(Verb): HTTP Method

서버의 자원에 접근해 상태를 조작하기 위한 요청 행위로서, 각 조작 행위는 HTTP Method 를 통해 표현된다. (POST, GET, PUT, DELETE)

6.2.3 표현(Representation): JSON

클라이언트의 요청에 대한 서버의 응답 형식으로, 주로 JSON 이 사용된다.

클라이언트와 서버 간의 통신에 REST API 를 사용할 경우 서버와 클라이언트 간의 의존성이 줄어들고, 프로토콜이 이해하기 쉬워지는 장점이 있으며, 이를 통해 본 시스템에서 개발하는 프론트엔드 클라이언트뿐 아니라 다른 시스템에서 서버의 자원을 쉽게 이용할 수 있기 때문에 확장성이 높다

6.3 JSON

Figure 22: JSON

JSON(제이슨, JavaScript Object Notation)은 속성-값 쌍(attribute–value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX 가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

6.4 Details

6.4.1 Login

- Request

Method	POST	
URI	/user/login	
Request Body	username	사용자 아이디
	password	사용자 비밀번호

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Failure Response	message	ID/password mismatch

6.4.2 Signup

- Request

Method	POST	
URI	/user/signup	
Request Body	username	사용자 아이디
	password	사용자 비밀번호

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Failure Response	message	Duplication of ID

6.4.3 Get items in user's refrigerator

- Request

Method	GET	
URI	/user/refrigerator	
Parameter	-username	-
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	items	Item Object List
Failure Response	message	Database error

6.4.4 Put/Remove items in refrigerator

- Request

Method	POST/DELETE	
URI	/user/refrigerator	
Request Body	username	사용자 아이디
	items	추가/삭제 Item Object List
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	400 Bad Request

6.4.5 Recipe Recommendation

- Request

Method	POST	
URI	/recipe/recommendation	
Request Body	good	선호하는 재료 리스트
	bad	선호하지 않는 재료 리스트
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	recipe_main	Recipe 간략한 정보 리스트
Failure Response	message	Database error

6.4.6 Recipe Search

- Request

Method	GET	
URI	/recipe/search	
Parameters	keyword	검색할 레시피 이름
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	recipe_main	Recipe 간략한 정보 리스트
Failure Response	message	Database error

6.4.7 Recipe Detailed Information

- Request

Method	GET	
URI	/recipe/detail	
Parameters	username	사용자 아이디
	recipe_id	레시피 아이디
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	info	레시피 상세정보 리스트
Failure Response	message	Database error

6.4.8 Get Item Basket

- Request

Method	GET	
URI	/user/basket	
Parameters	username	사용자 아이디
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	info	장바구니에 담긴 재료 리스트
Failure Response	message	Database error

6.4.9 Put items in Basket

- Request

Method	POST	
URI	/user/basket	
Request Body	username	사용자 아이디
	items	추가 Item Object List
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	400 Bad Request

6.4.10 Remove items in refrigerator

- Request

Method	DELETE	
URI	/user/basket/remove	
Request Body	username	사용자 아이디
	items	추가/삭제 Item Object List
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	400 Bad Request

6.4.11 Move items in Basket to Refrigerator

- Request

Method	DELETE	
URI	/user/basket/move	
Request Body	username	사용자 아이디
	items	추가/삭제 Item Object List
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	400 Bad Request

6.4.12 Click Like button of Recipe

- Request

Method	GET	
URI	/recipe/detail/like	
Parameters	username	사용자 아이디
	recipe_id	레시피 아이디
	like	1이면 like, 0이면 dislike
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Failure Response	message	Database error

7 Database Design

7.1 Objectives

요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER-Diagram을 통해 개괄적인 Entity 간의 관계를 기술하고, Relational Schema, SQL DDL 명세를 만든다.

7.2 ER-Diagram

본 시스템에는 User, Items, Products, Authentic Recipe, Unauthentic Recipe 총 5개의 Entity가 존재한다. 각각의 Entity는 네모 박스의 형태로 표현되고, Entity 간의 관계는 마름모꼴로 표현된다. 각 Entity가 가지는 Attribute는 타원형으로 표현되는데, 각 Entity를 식별하는Unique key는 라벨에 밑줄을 그어 표시한다.

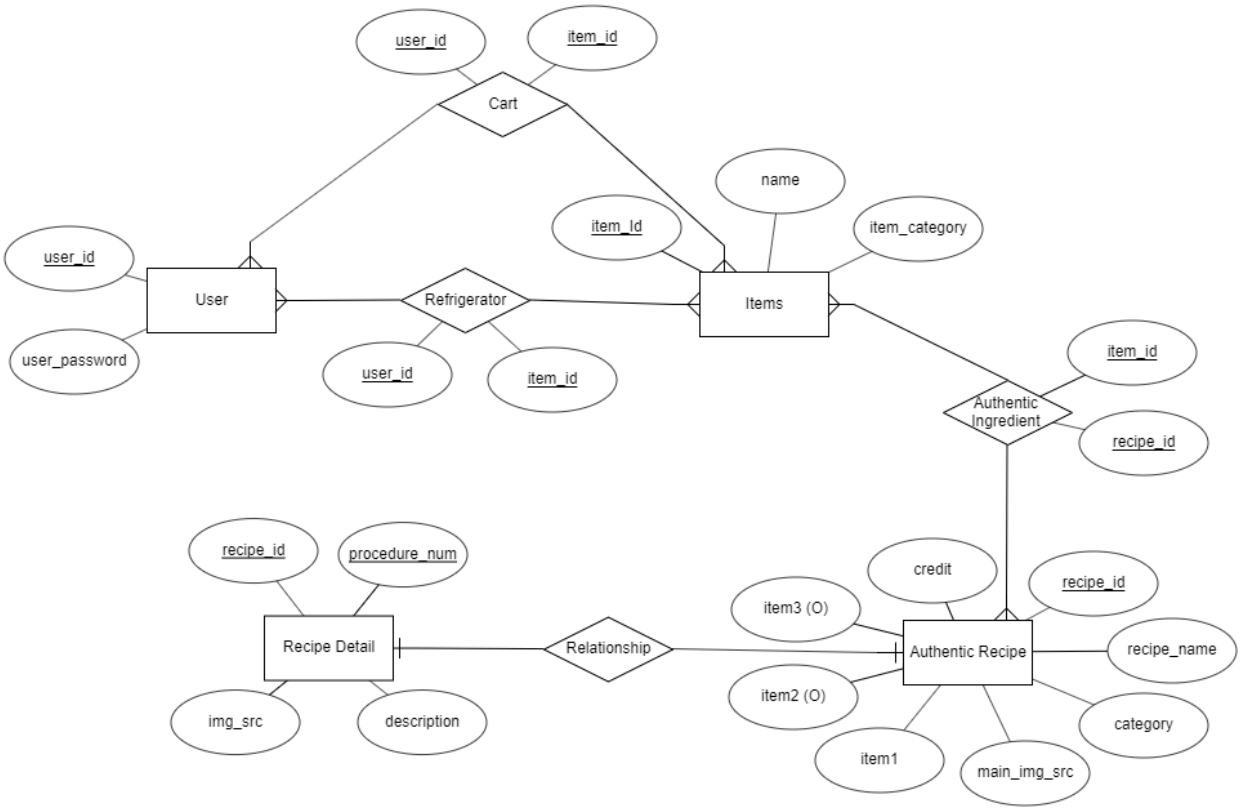

Figure 23: ER-Diagram

7.2.1 User

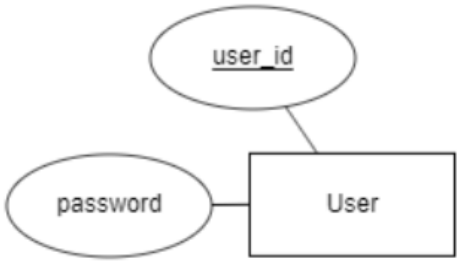

Figure 24: User

User Entity는 이용자의 정보를 저장한다. 이용자의 ID, PASSWORD를 저장한다. User_id가 primary key이다.

7.2.2 Items

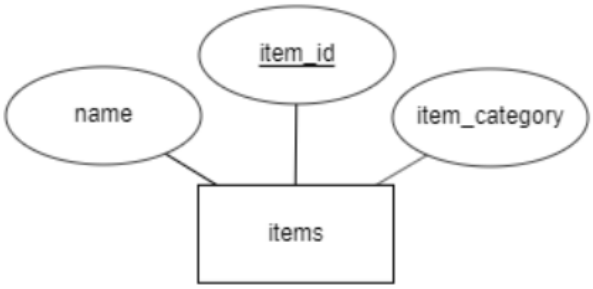

Figure 25: Items

Items Entity는 재료 정보를 저장한다. 재료의 고유 아이와 이름, 그리고 재료의 카테고리 (ex: 유제품, 육류)를 저장한다. Item_id 가 primary key이다.

7.2.3 Refrigerator

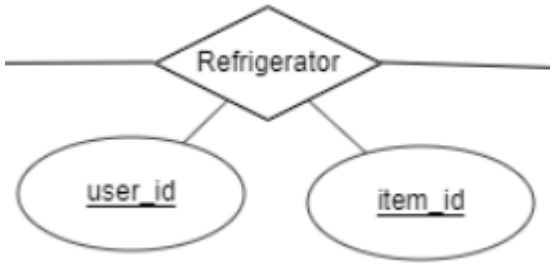

Figure 27: Refrigerator

Refrigerator Entity는 User Entity와 Items Entity 사이 관계에서 나온 Entity이다. Refrigerator Entity에서는 이용자의 아이디, 재료의 아이디, 재료의 개수를 저장함으로써 이용자의 냉장고 정보를 저장한다.

7.2.4 Authentic Recipe

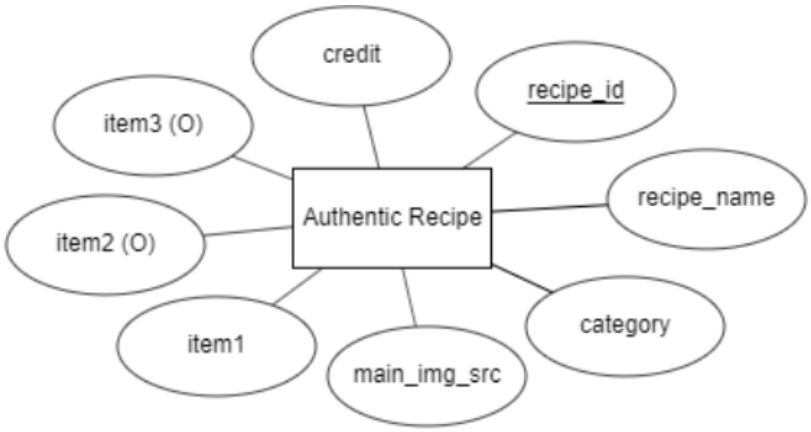

Figure 28: Authentic Recipe

Authentic Recipe Entity는 관리자에 의해 인증 받은 레시피들을 저장한다. 레시피 아이디, 레시피 이름, 카테고리, 주요 재료 3가지, 메인 사진 경로, 좋아요 개수를 저장한다. recipe_id가 primary key이다.

7.2.5 Authentic Ingredient

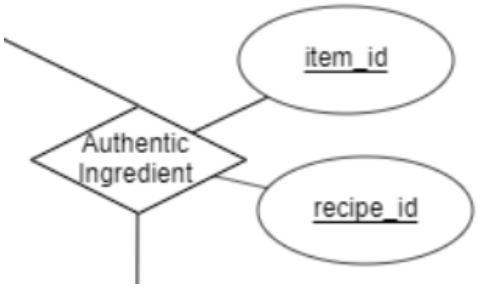

Figure 30: Authentic Ingredient

Authentic Ingredient Entity는 Items Entity와 Authentic Recipe Entity 사이 관계에서 나온 Entity이다. Authentic Ingredient Entity에는 레시피 아이디, 레시피에 들어간 재료 아이디를 저장한다.

7.2.6 Recipe Detail

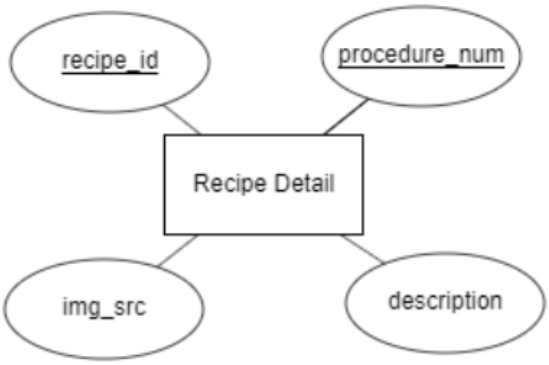

Figure 32: Recipe Detail

레시피의 자세한 정보를 담는 테이블이다. Recipe_id를 FK로 가져오고 각 레시피 별로 하나의 Recipe Detail 테이블을 가진다. Procedure_num은 절차 번호, img_src는 레시피 절차에 해당하는 이미지 주소, description은 절차 설명 글을 가진다.

7.2.7 Cart

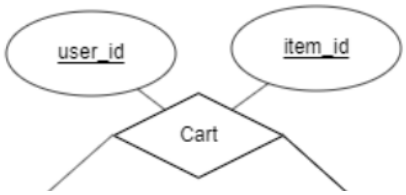

Figure 33: Cart

Cart는 user가 어떤 재료를 장바구니에 담았는지에 대한 정보를 담는 테이블이다. User_id는 User 테이블에서, item_id는 Items 테이블에서 가져온 FK이다.

7.3 Relational Schema

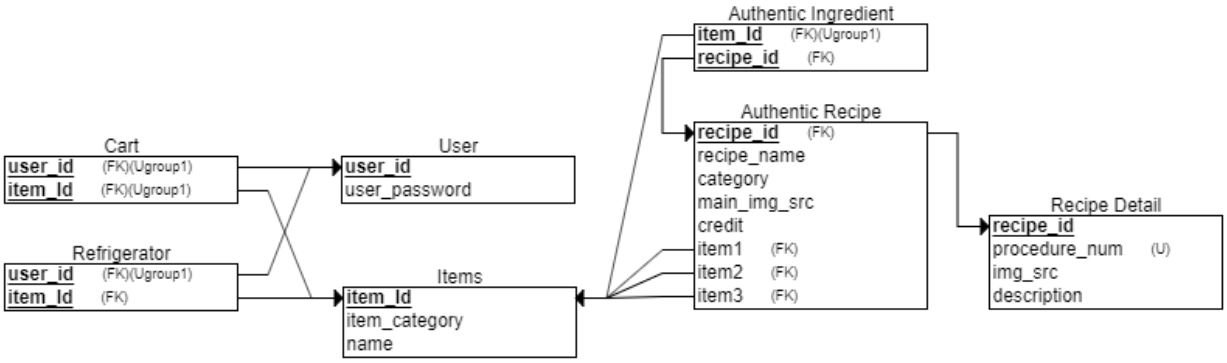

Figure 34: Relational Schema

7.4 SQL DDL

7.4.1 User

```
CREATE TABLE user
(
  user_id VARCHAR(16) NOT NULL,
  user_password VARCHAR(16) NOT NULL,
  PRIMARY KEY(user_id)
);
```

7.4.2 Items

```
CREATE TABLE items
(
  item_id INT NOT NULL,
  name VARCHAR(24) NOT NULL,
  item_category VARCHAR(24) NOT NULL,
  PRIMARY KEY (item_id)
);
```

7.4.3 Refrigerator

```
CREATE TABLE refrigerator
  (
  user_id VARCHAR(16) NOT NULL,
  item_id INT NOT NULL,
  FOREIGN KEY(user_id) REFERENCES
user(user_id) ON DELETE CASCADE,
  FOREIGN KEY(item_id) REFERENCES
items(item_id) ON DELETE CASCADE
  PRIMARY KEY(user_id, item_id)
);
```

7.4.4 Authentic Recipe

```
CREATE TABLE authentic_recipe
(
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(60) NOT NULL,
  main_img_src VARCHAR(30) NOT NULL,
  credit INT,
  category VARCHAR(24) NOT NULL,
  item1 INT NOT NULL,
  item2 INT,
  item3 INT,
  description VARCHAR(500) NOT NULL,
  PRIMARY KEY (recipe_id)
);
```

7.4.5 Authentic Ingredient

```
CREATE TABLE authentic_ingredient
(
  item_id INT NOT NULL,
  recipe_id INT NOT NULL,
  PRIMARY KEY(item_id, recipe_id),
  FOREIGN KEY(item_id) REFERENCES
items(item_id) ON DELETE CASCADE,
  FOREIGN KEY(recipe_id) REFERENCES
authentic_recipe(recipe_id) ON DELETE
CASCADE
);
```

7.4.6 Recipe Detail

```
CREATE TABLE recipe_detail
(
  recipe_id INT NOT NULL,
  procedure_num INT NOT NULL,
  description VARCHAR(500) NOT NULL,
  img_src VARCHAR(30),
  PRIMARY KEY(recipe_id, procedure_num),
  FOREIGN KEY(recipe_id) REFERENCES
authentic_recipe(recipe_id) ON DELETE
CASCADE
);
```

7.4.7 Cart

```
CREATE TABLE cart
  (
  user_id VARCHAR(16) NOT NULLl,
  item_id INT NOT NULL,
  FOREIGN KEY(user_id) REFERENCES
user(user_id) ON DELETE CASCADE,
  FOREIGN KEY(item_id) REFERENCES
items(item_id) ON DELETE CASCADE,
  PRIMARY KEY(user_id, item_id)
);
```

8 Testing Plan

8.1 Objectives

시스템이 작성한 문서대로 설계되어야하며, 요구사항에 맞게 작동해야합니다. 따라서 시스템이 의도한 방향으로 실행되는지 확인하고 시스템 내부의 결함을 찾기 위해 Testing을 진행합니다. Testing Plan 에서는 Testing Policy와 Test Case에 대해 서술합니다.

8.2 Testing Policy

시스템을 Development Testing, Release Testing, 그리고 User testing 세 파트로 나누어 Testing을 진행합니다.

8.2.1 Development Testing

Development Testing는 소프트웨어 개발 위험을 줄이기 위해 광범위한 결함 예방 및 탐지 전략의 적용을 동기화하기위한 것입니다. 이 테스트를 통해 미리 버그를 감지하여 시간과 비용 측면에서 모든 종류의 위험을 피할 수 있도록 합니다. 이 단계에서는 code analysis, code review, data analysis, unit testing, 그리고 system testing을 진행합니다. 또한 안정성, 보안, 성능 및 규정 측면에서 문제가 없는지 확인합니다.

Development Testing은 Component Testing, Integrating Testing, System Testing, 그리고 Acceptance Testing 의 4단계로 진행됩니다. Component Testing을 통해 각각의 Component에 대하여 Test를 진행하고, Sub-System을 통합하여 Integration Testing을 진행합니다.Test Case를 만들어 System Testing을 진행한 후, User 환경에서 Acceptance Testing을 진행합니다.

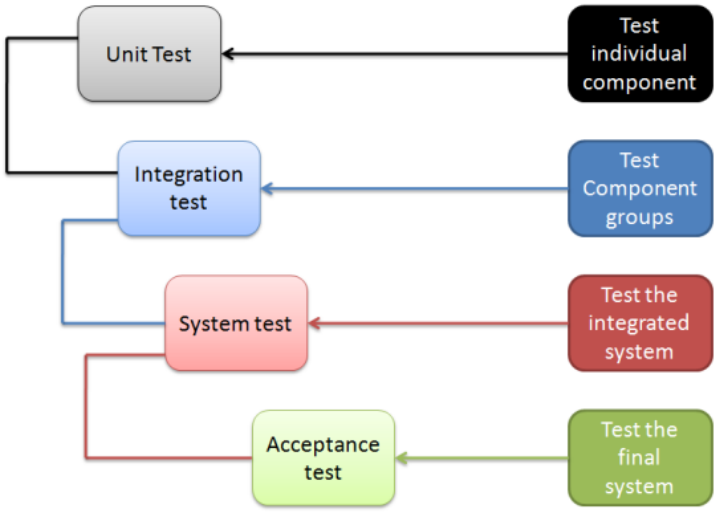

Figure 35: Testing process

8.2.2 Release Testing

사용자에게 Release 하기 전에 최종 시스템을 테스트해야 하며, 작성된 요구 사항이 올바르게 반영되는지 확인해야합니다. 이를 위해 Release Testing을 진행하며, Release Testing은 소프트웨어의 버전 및 빌드 상태를 테스트하여 오류가 없는지, 의도 한대로 작동하는지 확인하는 과정을 거칩니다.

8.2.3 User Testing

다지인한 시스템이 User가 사용할 경우 여러 가지 변수가 있을 수 있습니다. 그래서 User Testing을 통해 User의 환경에서 Testing을 진행합니다.

8.3 Test Case

8.3.1 Sign-up

User	회원가입 버튼을 통해 회원가입을 시도한다. 아이디, 패스워드, 패스워드 확인 칸을 입력한다. 최종적으로 회원가입 요청을 한다.
System	User DB 에 있는 ID 와 중복된 아이디를 입력하였는지 확인한다. 모든 입력요소가 입력되었는지 확인한다.
Success	회원가입이 되었다는 문구를 사용자에게 출력하고, 사용자 정보를 DB 에 저장한다.
Failure	시스템은 유저에게 이미 사용중인 ID 를 입력했거나 모든 요소를 입력하지 않았음을 알린다.

8.3.2 Login

User	사용자가 아이디와 비밀번호로 로그인을 시도한다
System	입력 받은 ID/PW 가 User DB 에 있는지 확인한다. 비밀번호가 일치하는지 확인한다.
Success	해당 아이디로 냉모밀 서비스에 로그인을 하고, 냉모밀 메인 화면 중 첫번째 탭인 '냉장고'를 보여준다.
Failure	시스템이 비밀번호가 일치하지 않다고 알린다.

8.3.3 Add Ingredients into refrigerator

User	'추가' 버튼을 눌러 뜨는 아이템 리스트 중에서 냉장고에 추가하고 싶은 재료를 누르고 '추가' 버튼을 누른다.
System	사용자로부터 입력 받은 user_id, item_id 를 Refrigerator DB 에 추가한다.
Success	추가한 재료가 들어간 업데이트된 냉장고 화면을 보여준다.
Failure	냉장고 화면에 변화가 없다.

8.3.4. Delete Ingredients from refrigerator

User	냉장고 화면에서 지우고 싶은 재료를 누르고 '제거' 버튼을 누른다.
System	사용자로부터 입력 받은 user_id, item_id 를 Rerigerator DB 에서 삭제한다.
Success	삭제한 재료가 사라진 업데이트된 냉장고 화면을 보여준다.
Failure	냉장고 화면에 변화가 없다.

8.3.5. Recipe Searching

User	사용자가 검색하기 위한 키워드를 입력하고 검색 버튼을 누른다.
System	사용자로부터 입력 받은 데이터를 확인하여, Recipe 관련 DB 에서 입력 받은 데이터와 일치하는 모든 데이터를 불러온다. 이때 레시피는 좋아요 숫자가 큰 순서로 정렬된다.
Success	정렬된 레시피의 간략한 정보(이름, 메인 사진, 메인 설명, 주요 재료)를 사용자에게 출력한다.
Failure	빈 결과 화면을 보여준다.

8.3.6. Recipe Recommendation

User	사용자가 포함할 재료와 포함시키지 않을 재료를 선택한다.
System	메인 재료에 포함할 재료가 있고, 전체 재료에 포함시키지 않을 재료가 없는 레시피를 필터링하여 좋아요 개수로 내림차순 하여 보여준다.
Success	정렬된 레시피의 간략한 정보(이름, 메인 사진, 메인 설명, 주요 재료)를 사용자에게 출력한다.
Failure	빈 결과 화면을 보여준다.

8.3.7. Recipe Detail

User	레시피 목록에서 상세보기를 원하는 레시피를 누른다.
System	해당 레시피의 상세 정보 (메인 정보, 재료, 요리 절차, 좋아요 개수)를 보여준다.
Success	정렬된 레시피의 간략한 정보(이름, 메인 사진, 메인 설명, 주요 재료)를 사용자에게 출력한다.
Failure	빈 결과 화면을 보여준다.

8.3.8. Click Like of Recipe

User	레시피 상세보기 제일 아래에 있는 좋아요 버튼을 누른다.
System	이미 해당 레시피에 좋아요를 누른 사용자라면 좋아요 취소를, 좋아요를 누르지 않은 사용자라면 좋아요를 해준다. 이 정보는 recipe_credit DB 에 저장되고, 좋아요 여부는 좋아요 버튼의 색깔 차이로 사용자에게 알려준다.
Success	좋아요 버튼이 눌러지고 업데이트 된 좋아요 개수를 반영한다.
Failure	좋아요 버튼이 눌러지지 않는다.

8.3.9. Add items to cart

User	(1) 레시피 상세보기에 재료 옆 '담기' 버튼을 눌러 원하는 재료를 장바구니에 담는다. (2) 하단에 Basket 탭으로 이동해 장바구니에 담고 싶은 재료를 선택해서 장바구니에 담는다.
System	사용자로부터 입력 받은 데이터를 확인하여, user_id 와 item_id 를 cart 테이블에 추가한다.
Success	(1) 레시피 상세보기에 재료 옆 '담기' 버튼이 눌려진 상태가 되고, 장바구니에 담은 재료가 담겨있다. (2) 담기를 원하는 재료가 추가된 update 된 장바구니 화면을 보여준다.
Failure	사용자의 장바구니에 해당 재료가 담기지 않는다.

8.3.10. Delete items from basket

User	삭제하고 싶은 재료 옆에 '삭제' 버튼을 누른다.
System	해당 재료를 Cart DB 에서 제거하고 업데이트 된 장바구니 화면을 보여준다.
Success	삭제를 누른 재료가 사라진다.
Failure	재료가 삭제되지 않는다.

8.3.11. Show Web browser for shopping

User	장바구니 하단에 '구매하러 가기' 버튼을 누른다.
System	식재료를 살 수 있는 사이트를 어플 내 브라우저창으로 띄워주고, 장바구니 내용을 상단에 함께 보여준다.
Success	브라우저와 장바구니 내용이 같이 뜬다.
Failure	해당 브라우저 창이 열리지 않는다.

8.3.12. Purchase items in the basket

User	냉장고로 이동하고 싶은 재료 옆에 '담기' 버튼을 누른다.
System	해당 재료를 Cart DB 에서 삭제하고 Refrigerator DB 에 추가한다.
Success	담기를 누른 재료가 사라지고 냉장고에 해당 재료가 추가되어 나타난다.
Failure	재료가 이동되지 않는다.

1 Development Plan

1.1 Objectives

실제 개발 단계에서 어떠한 기술과 개발 환경을 사용할 것인지를 기술하고, 개발 일정과 진행 상황을 설명한다.

1.2 Frontend Environment

1.2.1 IDE

- Android Studio

Figure 36: Android Studio

안드로이드 스튜디오는 구글에서 발표한, 안드로이드 및 안드로이드 전용 어플리케이션 제작을 위한 공식 통합 개발 환경 (IDE)이다.

1.2.2 Language

- JAVA

Figure 37: JAVA language

Java는 객체 지향 프로그래밍 언어로서, 안드로이드 어플리케이션을 구현하는데 사용된다. 안드로이드 어플리케이션은 코틀린으로도 구현할 수 있지만, 팀원들이 JAVA에 익숙하여 JAVA로 language를 정하였다.

1.3 Backend Environment

1.3.1 Server Platform & Language

- Node.js

Figure 38: Node.js

확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있으며, JavaScript로 브라우저 밖에서 서버를 구축하는 등의 코드를 실행할 수 있게 해주는 런타임 환경이다

- JavaScript

Figure 39: JavaScript

객체 기반의 스크립트 프로그래밍 언어이며, 웹 브라우저, 또는 Node.js와 같은 런타임 환경과 같이 서버 사이드 네트워크 프로그래밍에도 사용되고 있다

1.3.2 DBMS(Database Management System)

- MySQL

Figure 40: MySQL

세계에서 가장 많이 쓰이는, 오픈 소스의 관계형 데이터베이스 관리 시스템(RDBMS)이다.

1.4 Version Management

- GitHub

Figure 41: GitHub

깃허브는 분산 버전 관리 툴인 깃(Git)을 사용하는 프로젝트를 지원하는 웹 서비스이며, 이를 이용하여 협업과 소스코드 관리를 할 예정이다.

1.5 Schedule

		10월	11월				12월	
		2주차	1주차	2주차	3주차	4주차	5주차	1주차	2주차
Concept & Proposal									
Requirement Specification									
Design Specification									
Frontend	login/sign_up								
	레시피 등록/추천/검색 페이지								
	구매/관리 페이지								
Backend	login/sign_up								
	등록/추천/검색								
	구매/ 레시피 관리								
System Integration/Testing									
Presentation									

2 Figure Index

Figure 1: Unified Modeling Language(UML) ..... - 4 -

Figure 2: Example of Class Diagram ..... - 5 -

Figure 3: Example of Sequence Diagram ..... - 6 -

Figure 4: draw.io Logo ..... - 6 -

Figure 5: PowerPoint Logo ..... - 7 -

Figure 6: ERDPlus Logo ..... - 7 -

Figure 7: Overall System Organization ..... - 9 -

Figure 8: System Architecture – Frontend ..... - 10 -

Figure 9: System Architecture – Backend ..... - 10 -

Figure 10: System Architecture – Frontend – Join/Login ..... - 12 -

Figure 11: System Architecture – Frontend – RefrigeratorPage ..... - 13 -

Figure 12: System Architecture – Frontend – Overall Recipe Recommendation & Recipe Searching ..... - 16 -

Figure 16: System Architecture – Frontend – Basket ..... - 19 -

Figure 18: Overall Backend Architecture ..... - 20 -

Figure 19: Backend-Server ..... - 21 -

Figure 22: JSON ..... - 25 -

Figure 23: ER-Diagram ..... - 32 -

Figure 24: User ..... - 33 -

Figure 25: Items ..... - 33 -

Figure 27: Refrigerator ..... - 34 -

Figure 28: Authentic Recipe ..... - 34 -

Figure 30: Authentic Ingredient ..... - 35 -

Figure 32: Recipe Detail ..... - 35 -

Figure 33: Cart ..... - 36 -

Figure 34: Relational Schema ..... - 37 -

Figure 35: Testing process ..... - 41 -

Figure 36: Android Studio ..... - 46 -

Figure 37: JAVA language ..... - 47 -

Figure 38: Node.js ..... - 47 -

Figure 39: JavaScript ..... - 48 -

Figure 40: MySQL ..... - 48 -

Figure 41: GitHub ..... - 49 -