

PROJECT FINAL REPORT

오픈소스 SW 기초 3 분반

SUBMITTED ON: 06/03/2025

OWNER: 하경준(32204781)

MEMBERS: 곽태훈(32210214), 한승진(32204837), 이예권(32223343)

CONTENTS

1. TEAM.....	- 3 -
1.1 MEMBERS	- 3 -
1.2 TOOLS FOR COOPERATION.....	- 3 -
2. PROJECT OVERVIEW	- 4 -
2.1 BACKGROUND AND OBJECTIVES	- 4 -
2.2 KEY FEATURES	- 5 -
2.3. SYSTEM ARCHITECTURE AND USER FLOW	- 8 -
2.3.1 System Architecture.....	- 8 -
2.3.2 User Flow.....	- 9 -
3. SYSTEM DESIGN AND IMPLEMENTATION.....	- 11 -
3.1 TECHNOLOGY STACK	- 11 -
3.2 DATABASE DESIGN AND ERD	- 12 -
3.2.1 Entity Relation Diagram	- 12 -
3.2.2 Table Details	- 12 -
3.2.2 RELATION DETAILS.....	- 12 -
3.3 CORE FUNCTIONAL MODULES.....	- 14 -
3.3.1 Receipt Upload and OCR Processing	- 14 -
3.3.2 Inventory and Stock Management.....	- 17 -
3.3.3 Recipe Recommendation using GPT API	- 22 -
3.4 RESTFUL API SPECIFICATION.....	- 23 -
4. TESTING AND ANALYSIS.....	- 29 -
4.1 TEST SCENARIOS AND CASES	- 29 -
4.2 TEST RESULTS AND ANALYSIS	- 30 -
5. TERMINOLOGY AND REFERENCES.....	- 44 -
5.1 TERMS.....	- 44 -
5.2 REFERENCES.....	- 44 -

1. TEAM

본 장에서는 프로젝트 수행을 위해 구성된 팀의 역할 분담과 협업 방식에 대해 간략히 소개한다.

1.1 MEMBERS

본 프로젝트는 총 4 명의 팀원으로 구성되어 있으며, 크게 프로젝트 매니저(PM), UX/UI 디자인 및 프론트엔드(FE) 개발, 데이터베이스 설계 및 백엔드(BE) 개발로 업무를 나누어 담당하였다.

NAME	POSITION
곽태훈	UX/UI Design & Frontend (FE)
이예권	UX/UI Design & Frontend (FE)
하경준	PM & Backend (BE)
한승진	Database Design & Backend (BE)

1.2 TOOLS FOR COOPERATION

팀은 FIGMA 를 통해 와이어프레임 및 사용자 인터페이스(UI) 설계를 공유하고 상호 피드백을 통해 프로젝트 기간동안 지속적으로 개선하였다. 또한 전체 소스코드 및 문서를 공유하고 통합 관리하기 위한 목적으로 GITHUB 를 활용하였다.

TOOL	PURPOSE
FIGMA	UX/UI 및 와이어프레임 공유
GITHUB	프로젝트 전체 소스코드 공유 및 버전 관리
GITHUB LINK	https://github.com/junjinju/opensource-project

2. PROJECT OVERVIEW

본 장에서는 프로젝트의 기획 배경, 개발 목적, 핵심 기능, 그리고 시스템 아키텍처와 사용자 사용 흐름에 대해 상세히 설명한다.

2.1 BACKGROUND AND OBJECTIVES

현대 사회에서 식재료 낭비와 가정 내 음식물 쓰레기는 막대한 경제적 손실과 환경 문제를 유발하는 주요 이슈로 떠오르고 있다. 서울특별시 통계에 따르면, 한국 가정에서 발생하는 음식물 쓰레기의 약 34%는 유통기한이 지난 식재료에서 비롯되며, 이로 인해 매년 약 1조 원 이상의 처리 비용이 발생하고 있다. 그럼에도 불구하고 많은 가정에서는 여전히 냉장고 속 식재료를 체계적으로 관리하지 못한 채 방치하는 경우가 많으며, 결국 불필요한 폐기와 함께 자원 낭비, 탄소 배출 증가 등 환경적 악순환이 반복되고 있다.

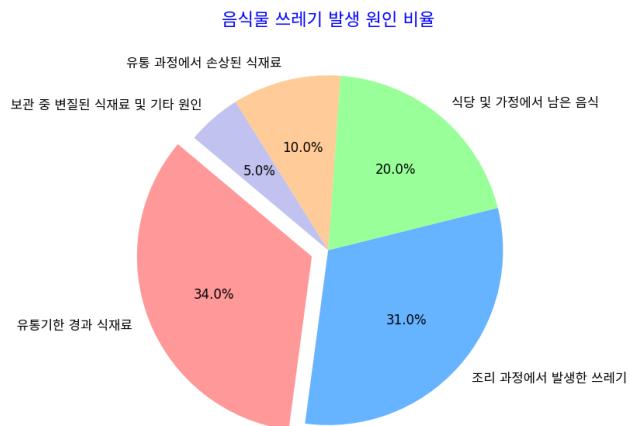


Figure 1: 음식물 쓰레기 발생 원인 비율, 서울특별시

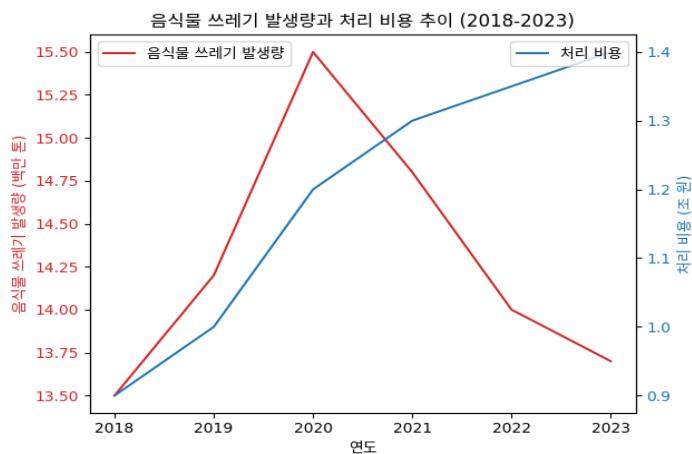


Figure 2: 음식물 쓰레기 발생량과 처리 비용 추이(2018-2023), 환경부-통계청 자료

기존의 재고 관리 어플리케이션은 대부분 물품을 수동으로 입력해야 하며, 이 과정에서 입력해야 할 물품 정보(이름, 수량, 가격, 구매일자, 유통기한 등)가 많아 번거롭고 반복적이라는 이유로 장기 사용이 어렵다는 피드백이 많았다. 따라서 본 프로젝트는 이러한 문제를 해결하기 위해 OCR(Optical Character Recognition) 기술을 활용한 반자동 재고 등록 기능을 구현하고자 한다.

사용자가 영수증 이미지를 업로드하면, 구매한 물품의 명칭과 수량은 자동으로 추출된다. 추출된 내용은 사용자가 직접 확인하고 수정할 수 있으며, 필요한 항목만 선택하여 인벤토리에 등록할 수 있도록 UI를 제공한다. 또한 단순히 물품 정보를 기록하는 것에 그치지 않고, 보유 중인 물품을 기반으로 GPT API를 통해 레시피를 추천 받을 수 있는 기능도 함께 제공한다. 이는 사용자가 재료를 보다 적극적으로 소비하도록 유도하여, 식재료 낭비를 줄이고 실행활에서의 활용도를 높이는 데 기여하기 위함이다.

2.2 KEY FEATURES

본 프로젝트는 사용자의 냉장/냉동고 물품 관리를 편리하게 하기 위해 다음과 같은 주요 기능을 제공한다.

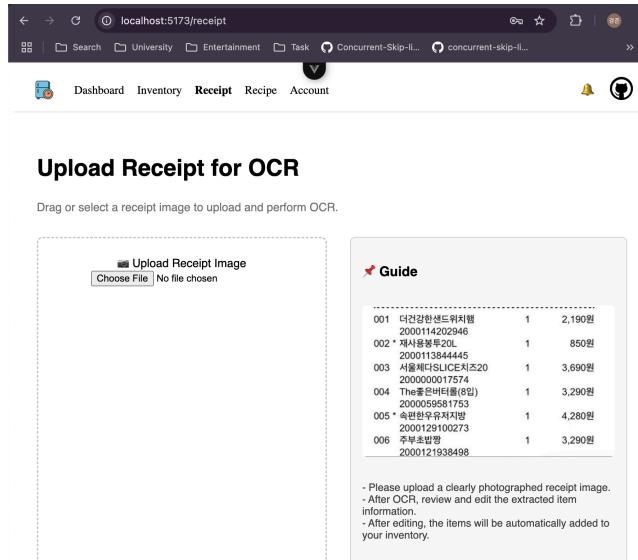


Figure 3: 영수증 이미지 업로드 페이지

사용자는 종이 또는 디지털 영수증을 이미지 형태로 업로드할 수 있다. 해당 기능은 OCR 처리를 위한 사전 단계로, 업로드된 이미지는 BE의 비지니스 로직에 의해 Firebase DB에 저장된 후 외부 OCR API로 전달된다.

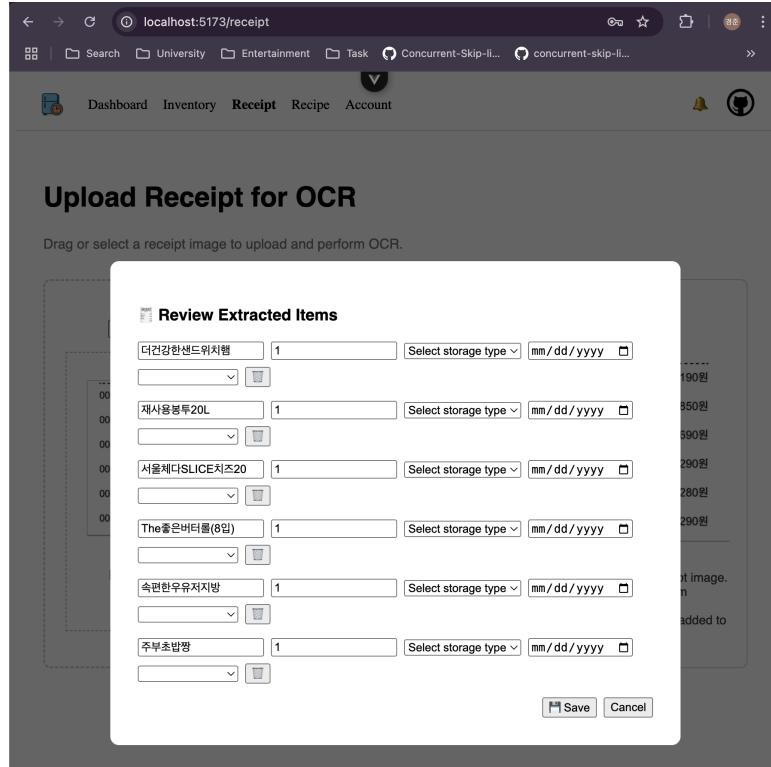


Figure 4: OCR 추출 결과 화면

OCR 처리를 마친 결과는 화면에 리스트 형태로 출력된다. 사용자는 추출된 항목(ParsedItem) 중 필요한 품목만 선택하여 등록할 수 있으며, 물품명과 수량은 자동완성된다. 필요시 물품명과 수량은 사용자가 직접 수정할 수 있으며, 저장 위치(냉장/냉동)와 유통기한은 아직 미입력 상태이므로 반드시 FE에서 사용자로부터 추가로 입력 받아야 한다. 최종 저장 시, 입력받은 물품 정보를 기반으로 BE는 해당 데이터를 실제 인벤토리(Inventory)에 인벤토리 물품(InventoryItem)으로 저장한다.

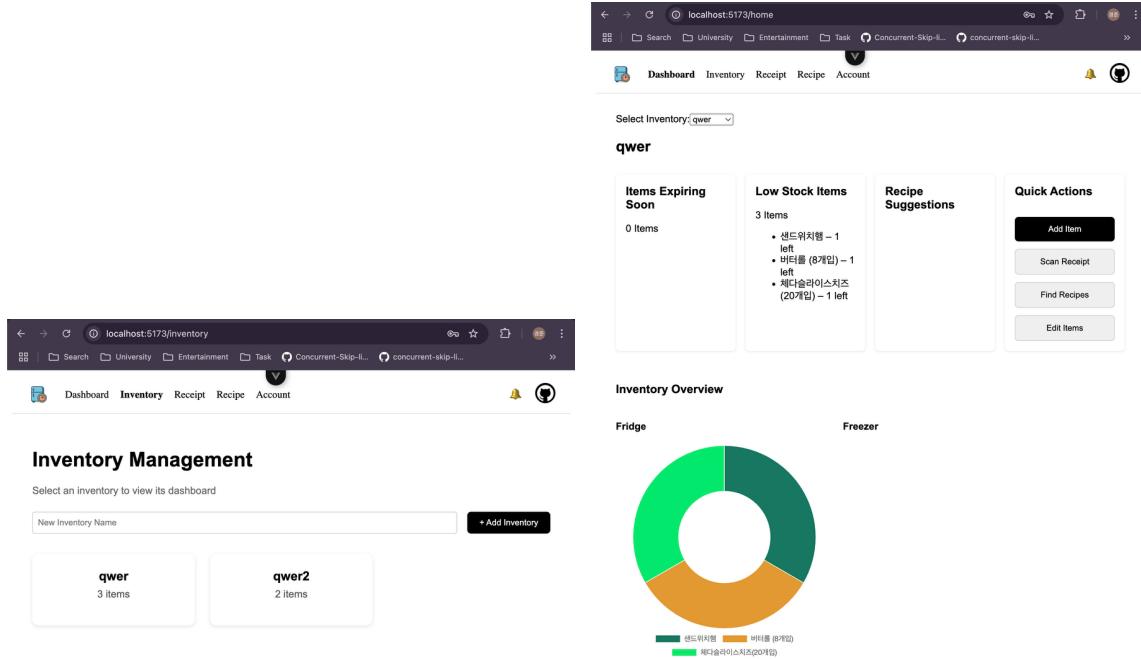


Figure 5: 물품 등록 및 재고 관리 페이지

사용자는 인벤토리(냉장/냉동고)를 자유롭게 생성하고, 각 인벤토리별로 재고를 등록, 확인, 수정, 삭제할 수 있다.

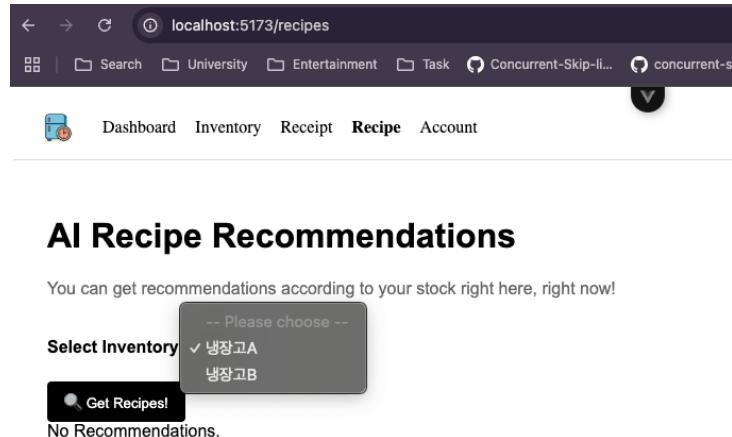


Figure 6: AI 레시피 생성 페이지

사용자는 재고 목록에서 일부 항목을 선택하여 레시피 생성을 요청할 수 있다. 선택된 물품은 GPT API로 전달되며, 최종적으로 FE는 레시피 결과를 전달받아 사용자에게 출력한다. 생성된 레시피는 사용자 계정에 저장되어 추후 재사용이 가능하다.

2.3. SYSTEM ARCHITECTURE AND USER FLOW

본 장에서는 전체 시스템의 구조와 사용자가 실제로 서비스를 이용하는 흐름을 도식화하여 설명한다.

2.3.1 System Architecture

본 프로젝트는 사용자 웹 인터페이스, 비지니스 로직 처리, 데이터 저장, 외부 API 연동 등 각 계층을 독립적으로 구성하였으며, 전체 시스템 아키텍처는 다음의 그림과 같다.

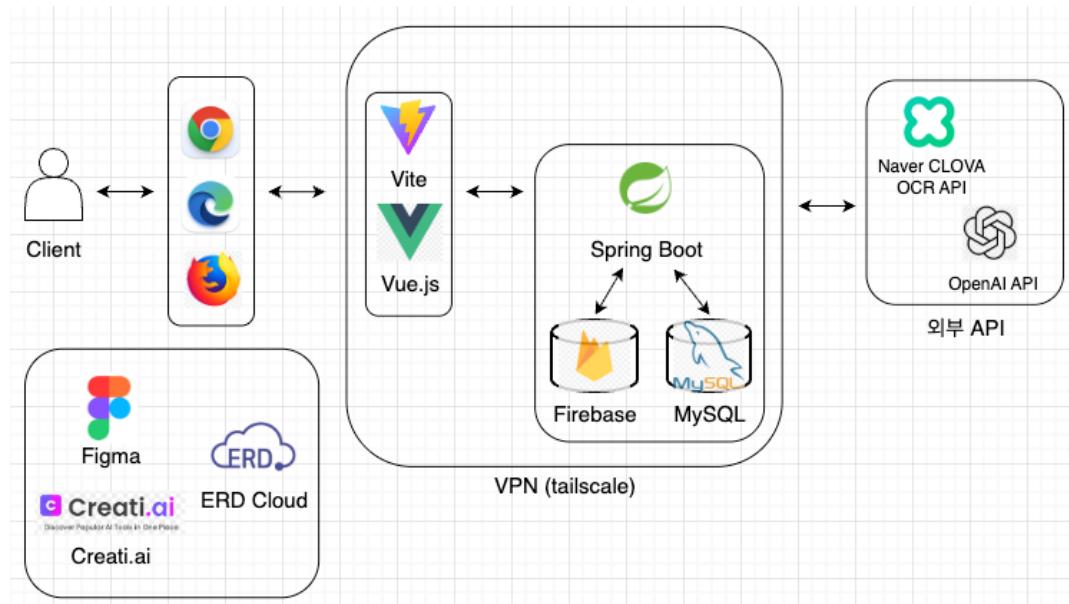


Figure 7: 시스템 아키텍처 개요

사용자는 웹 브라우저(Chrome, Edge 등)를 통해 서비스에 접근하여 상호작용한다. 사용자 인터페이스 및 FE는 Vue.js를 기반으로 구현되었으며, 사용자 입력, OCR 처리 결과, 재고 및 데이터 처리 UI 등을 제공한다. 빌드 도구로는 Vite를 함께 활용하였으며, FE와 BE는 RESTful API를 통해 서로 통신한다.

BE는 대표적인 프레임워크인 Spring Boot를 기반으로 개발되었으며 API 서버 역할을 수행한다. 데이터베이스로는 RDBMS인 MySQL과 영수증 이미지 파일을 저장하기 위한 Firebase를 함께 사용하였다.

OCR 처리는 Naver CLOVA OCR API 를 통해 수행되며, 사용자가 업로드한 영수증 이미지로부터 텍스트 형식으로 물품명과 수량을 추출한다.

2.3.2 User Flow

본 서비스에서 제공하는 핵심 서비스에 대한 사용자 흐름은 다음과 같이 구성된다.

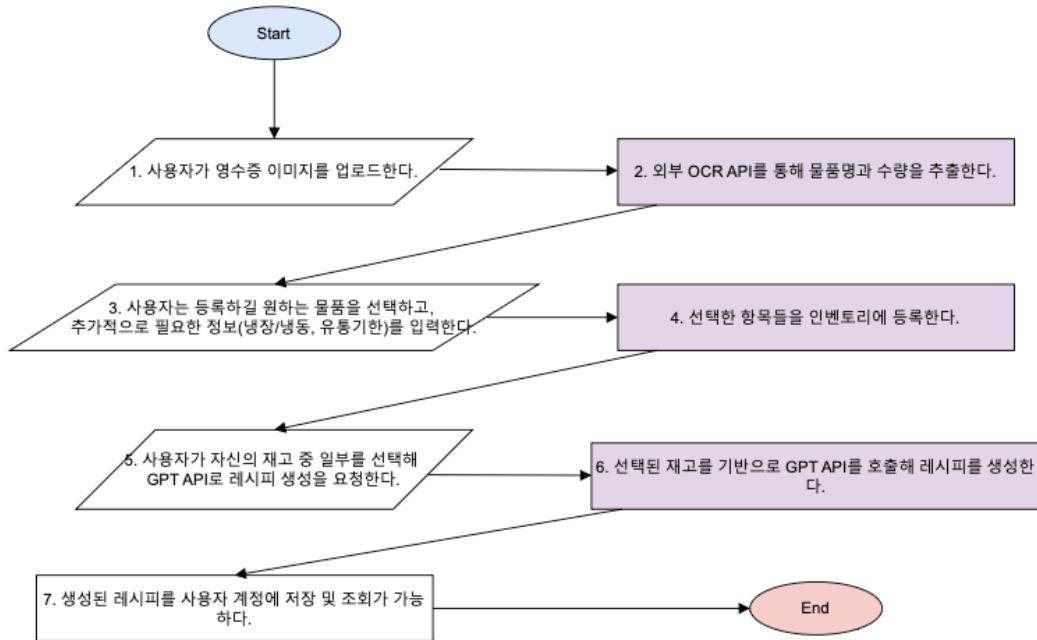


Figure 8: OCR 기반 물품 등록 및 GPT 레시피 생성 사용자 흐름도

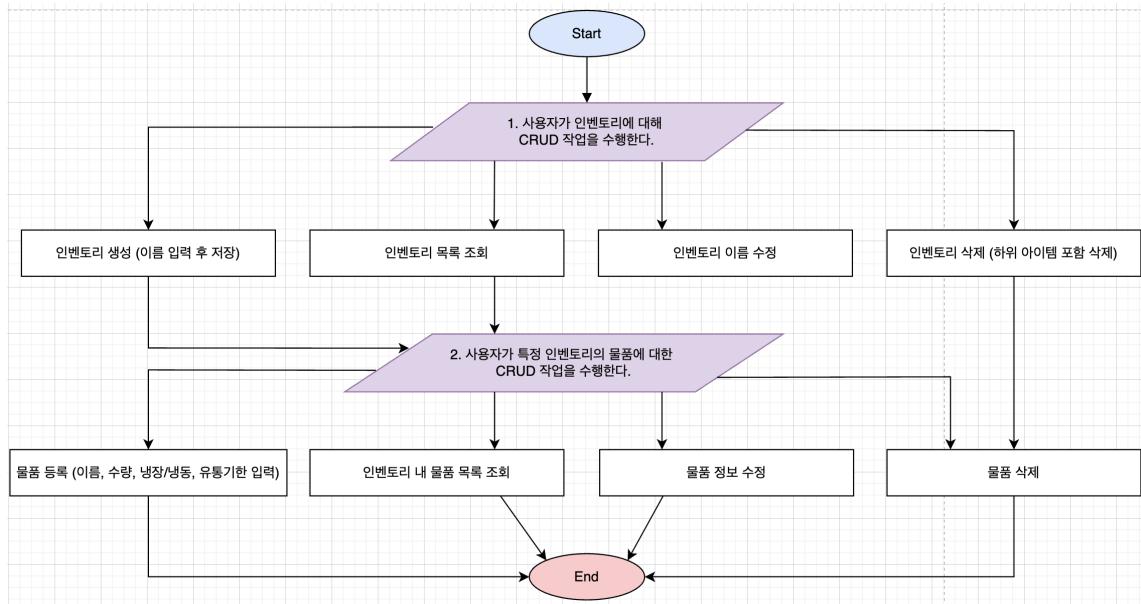


Figure 9: 인벤토리 및 물품 관리 기능에 대한 사용자 흐름도

먼저 사용자가 영수증을 업로드하면 (POST), 해당 이미지를 FE에서 BE로 전송하고 DB에 이미지를 저장한 뒤 성공 또는 실패 여부를 응답한다. 이미지 저장이 정상적으로 완료된 경우, BE는 해당 영수증에 부여된 receipt_id와 업로드 시각을 JSON 형식으로 클라이언트에 반환한다.

저장된 영수증에 대한 OCR 결과를 얻기 위해 FE는 receipt_id를 포함한 ParsedItem 조회를 BE에 요청한다 (GET). BE는 해당 receipt_id에 연결된 ParsedItem이 존재할 경우 추출된 모든 물품 데이터를 반환한다. 각 ParsedItem 레코드는 물품명과 수량 정보를 필드로 갖는다.

FE는 사용자에게 OCR 결과를 출력하고, 그 중 실제 저장할 물품을 선택할 수 있도록 UI를 구성한다. 이때 각 항목에 대해 물품명과 수량은 이미 자동완성 되어 있으며, 수정이 가능하다. 동시에 InventoryItem에 저장하기 위해 필요한 저장 방식(냉장/냉동) 및 유통기한 정보를 사용자로부터 추가로 입력 받는다.

모든 입력이 완료되면 FE는 최종적으로 사용자가 선택한 항목을 BE에 저장 요청 (POST) 및 해당 ParsedItem에 대한 삭제 요청 (POST)도 함께 전송한다. BE는 InventoryItem 테이블에 물품 정보를 저장하고, 등록이 완료된 ParsedItem들을 삭제함으로써 영수증 기반 물품 등록 프로세스를 종료한다.

한편, 사용자는 등록된 재고 중 일부 항목을 선택하여 레시피 생성을 요청 (POST) 할 수 있다. 먼저 FE는 사용자가 선택한 InventoryItem들의 정보 중 물품명과 수량을 BE에 전달한다. 이를 바탕으로 BE는 레시피 생성을 요청하는 프롬프트(prompt)를 생성하여 GPT API에 요청한다. 반환된 레시피에는 레시피 제목(title), 레시피 내용(content)이 포함되며 사용자 계정에 저장된다.

3. SYSTEM DESIGN AND IMPLEMENTATION

본 장에서는 프로젝트의 핵심 기능을 구현하기 위한 기술적 구성 요소와 데이터베이스 설계 구조를 설명한다. 전체 시스템은 프론트엔드와 백엔드, 외부 API(OCR 및 GPT)로 구성된다.

3.1 TECHNOLOGY STACK

본 프로젝트는 웹 기반 재고 관리 및 레시피 추천 시스템으로 다음과 같은 기술 스택을 기반으로 개발되었다.

Position	Person in charge	Tech stack	Description
Frontend	곽태훈 이예권	- Vue.js - Vite	- UX/UI 설계 및 시각화 - 빠른 개발 속도 및 유지보수에 용이
Design		- Figma - Creati.ai	- 기본 워드프레임 설계 및 화면 흐름 구성 - 시각적 요소 개발 보조 목적
Backend	하경준 한승진	- Springboot	- 데이터베이스 연동 간소화 - 소셜 로그인 및 보안 기능 적용 용이 - 대규모 커뮤니티 지원
Database		- MySQL - Firebase	- 사용자, 재고, 레시피, 영수증 등 데이터 저장 및 관계 관리
ERD Design		- ERD Cloud	- 간단한 ERD 공동 작업에 용이
OCR API		- Naver CLOVA OCR	- 영수증 이미지로부터 물품명 및 수량 자동 추출 - 타 OCR에 비해 한글 정확도 우수)
Recipe AI		- OpenAI GPT API	- 보유 재료 기반 간단한 레시피 텍스트 자동 생성

3.2 DATABASE DESIGN AND ERD

3.2.1 Entity Relation Diagram

사용자 식재료 등록과 레시피 추천을 핵심 기능으로 하여 최종 ERD 를 다음과 같이 설계하였다.

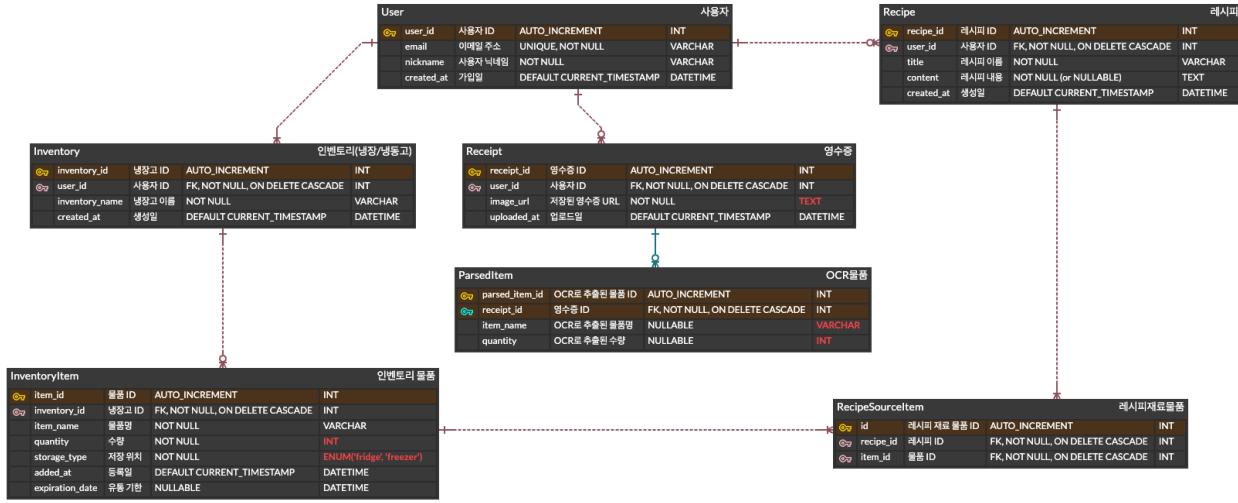


Figure 10: Entity-Relation Diagram (ERD)

3.2.2 Table Details

Table	Description
User	사용자 정보 및 인증 관리
Inventory	사용자가 소유한 냉장/냉동고 단위 저장소
InventoryItem	인벤토리에 포함된 실제 물품 정보
Receipt	OCR 처리를 위한 영수증 이미지 업로드 기록
ParsedItem	OCR로 추출된 물품명 및 수량(임시 저장용)
Recipe	GPT API로 생성된 레시피 및 텍스트 본문 관리
RecipeSourceItem	레시피에 사용된 물품의 원천을 추적하기 위한 테이블

3.2.2 RELATION DETAILS

Table	Table	Relation type	Description
User	Inventory	1 : N (Non-identifying)	한 명의 사용자는 여러 인벤토리를 소유 가능

User	Receipt	1 : N (Non-identifying)	한 명의 사용자는 여러 영수증을 저장 가능
User	Recipe	1 : N (Non-identifying)	한 명의 사용자는 여러 레시피를 생성 가능
Inventory	InventoryItem	1 : N (Non-identifying)	각 인벤토리는 여러 물품을 저장 가능
Receipt	ParsedItem	1 : N (Identifying)	OCR로 추출된 물품은 영수증에 완전 종속
InventoryItem	RecipeSourceItem	1 : N (Non-identifying)	하나의 물품은 여러 레시피에서 사용 가능
Recipe	RecipeSourceItem	1 : N (Non-identifying)	하나의 레시피는 여러 재료 물품을 포함 가능

테이블 간 관계는 대부분 비식별(Non-identifying) 관계로 설정하여 각 테이블이 독립적인 기본키를 갖고 자율적으로 관리될 수 있도록 하였다. 식별(Identifying) 관계는 단 하나의 경우에 적용되었는데, ParsedItem 테이블은 OCR 결과로 생성된 임시 데이터로 해당 항목은 반드시 특정 영수증에 종속되어 존재하기 때문이다.

한편, InventoryItem은 여러 레시피에서 참조될 수 있고, 하나의 레시피는 여러 InventoryItem을 사용할 수 있으므로 이러한 다대다(Many-to-Many) 관계를 해결하기 위한 목적으로 RecipeSourceItem을 추가하였다. 이는 InventoryItem과 레시피 간 중복 없이 양방향 참조가 자연스럽게 가능하도록 한다.

마지막으로, 사용자 삭제 시 관련된 인벤토리, 영수증, 레시피, 재고 아이템 등 모든 연관 데이터를 자동으로 제거할 수 있도록 외래 키 제약 조건에 ON DELETE CASCADE 옵션을 설정하였다. 이외에도, 데이터 간 종속성이 강하거나 생명주기를 함께하는 관계에 대해서도 동일한 제약을 설정하여 불필요한 고아 레코드 발생을 방지하도록 설계하였다.

3.3 CORE FUNCTIONAL MODULES

본 절에서는 프로젝트의 핵심 기능을 구성하는 주요 모듈과 각 모듈의 동작 흐름 및 역할에 대해 설명한다.

3.3.1 Receipt Upload and OCR Processing

본 모듈은 사용자가 업로드한 영수증 이미지를 처리하고, 이미지 내 물품 정보(물품명, 구매 수량)를 자동으로 추출하는 역할을 담당한다. 동작 화면은 다음과 같다:

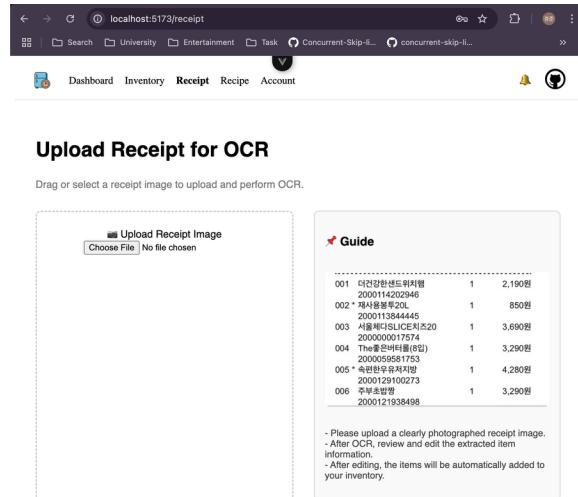


Figure 11: 영수증 업로드 화면

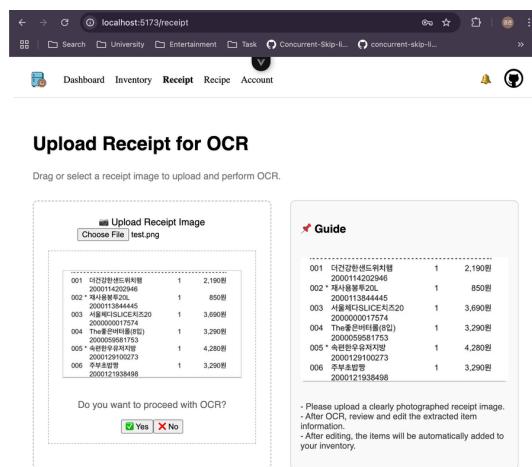


Figure 12: 업로드할 영수증 선택 후 화면

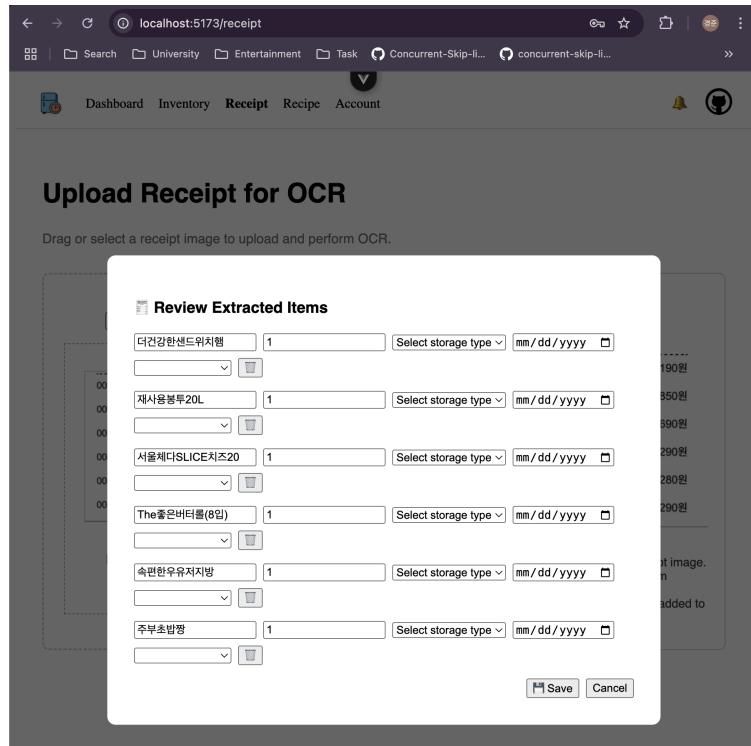


Figure 13: OCR 처리 후 결과화면

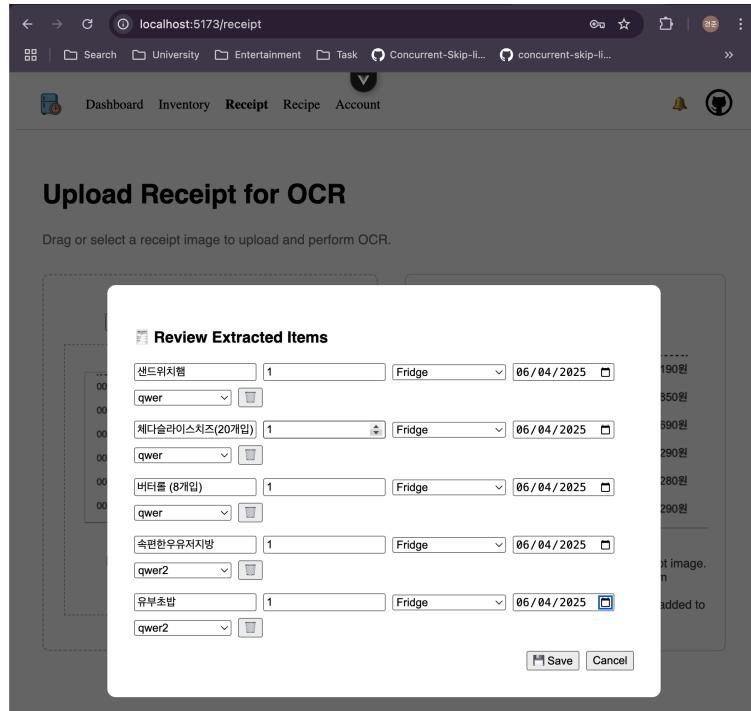


Figure 14: 불필요한 물품 제거 및 물품 등록을 위한 추가정보 입력

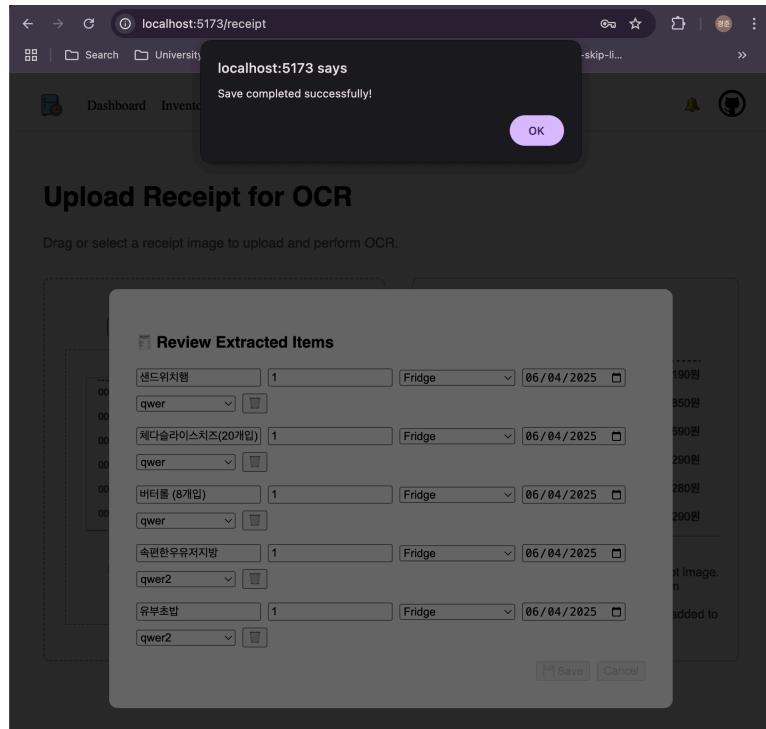


Figure 15: 최종 저장 후 결과화면

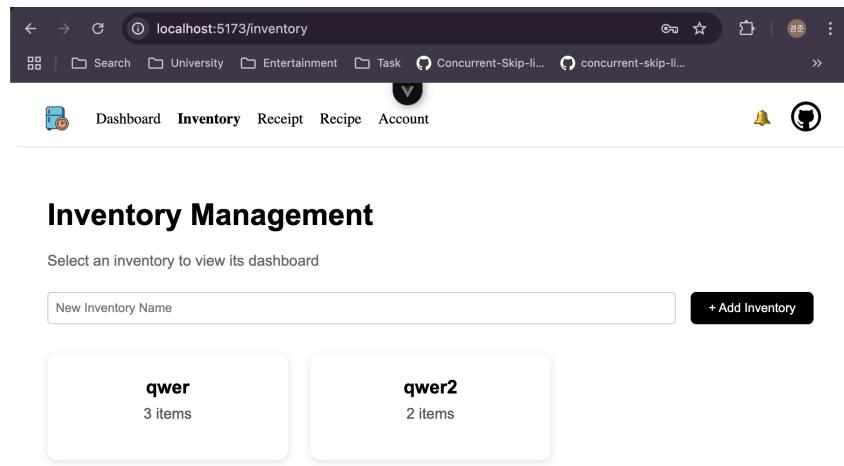


Figure 16: 전체 인벤토리 조회 (qwer: 3 개, qwer2: 2 개)

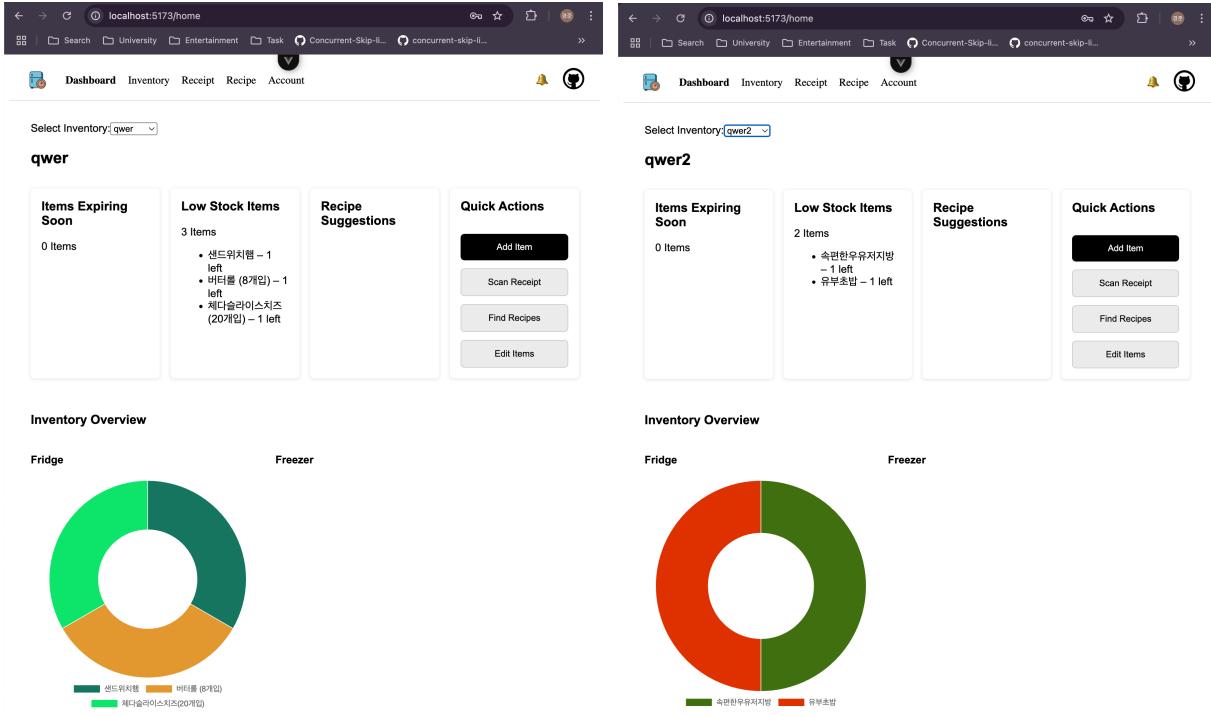


Figure 17: 각 인벤토리에 물품이 저장되었는지 확인(qwer, qwer2)

사용자가 영수증 이미지를 업로드하면, BE는 해당 이미지를 Firebase에 저장한 후 외부 OCR API로 전달한다. OCR 처리 결과에는 물품명과 수량이 포함되며, 이를 기반으로 ParsedItem 레코드가 생성된다. FE는 이 결과를 사용자에게 UI로 제공하며, 사용자는 원하는 항목을 선택하거나 물품 정보를 수정 및 추가할 수 있다. OCR 결과로서 저장된 ParsedItem 레코드는 사용자에 의해 InventoryItem 레코드로서 최종 등록이 완료되면 모두 삭제된다.

3.3.2 Inventory and Stock Management

본 모듈은 사용자가 인벤토리 및 해당 인벤토리에 등록된 물품 정보를 직접 생성, 조회, 수정, 삭제할 수 있도록 하는 CRUD 기능을 제공한다.

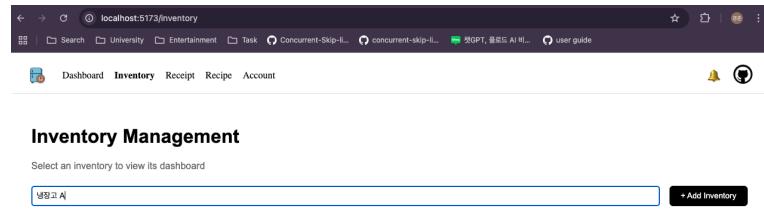


Figure 18: 새로운 인벤토리 생성 ("냉장고 A")

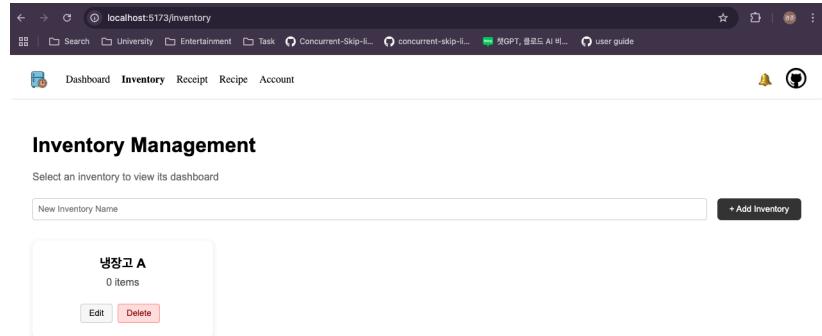


Figure 19: 인벤토리 생성 결과

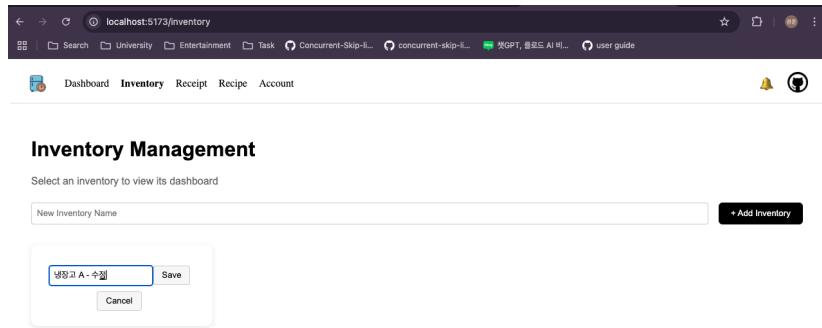


Figure 20: 인벤토리 이름 수정 ("냉장고 A" → "냉장고 A - 수정")

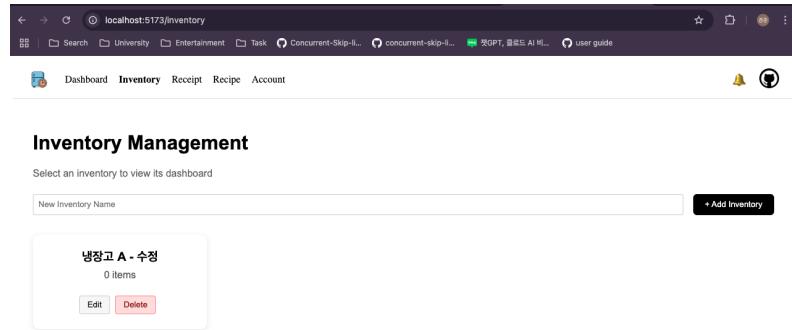


Figure 21: 인벤토리 이름 수정 후 결과

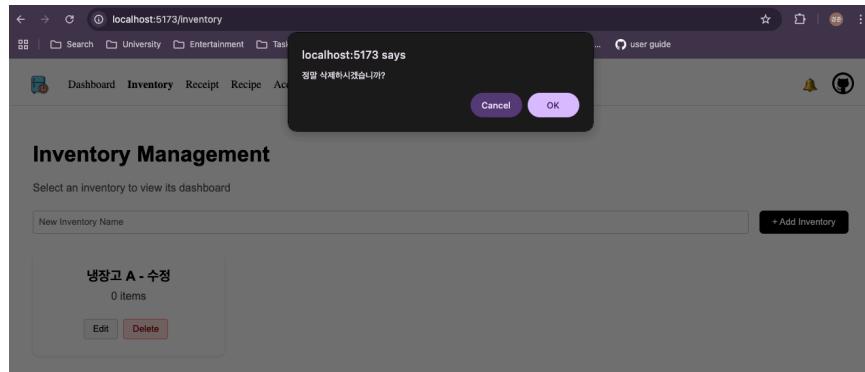


Figure 22: 인벤토리 삭제 과정 (“냉장고 A – 수정”)

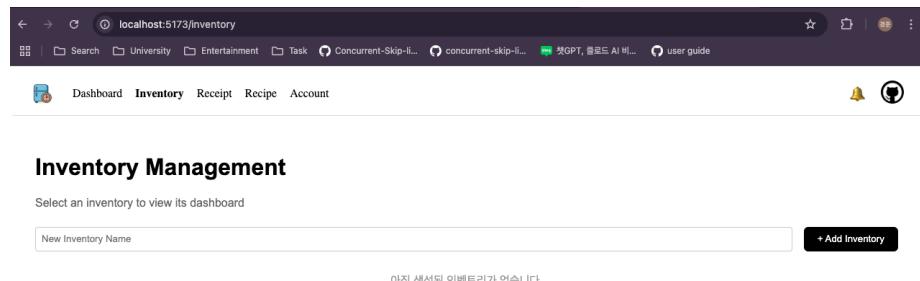


Figure 23: 인벤토리 삭제 후 결과

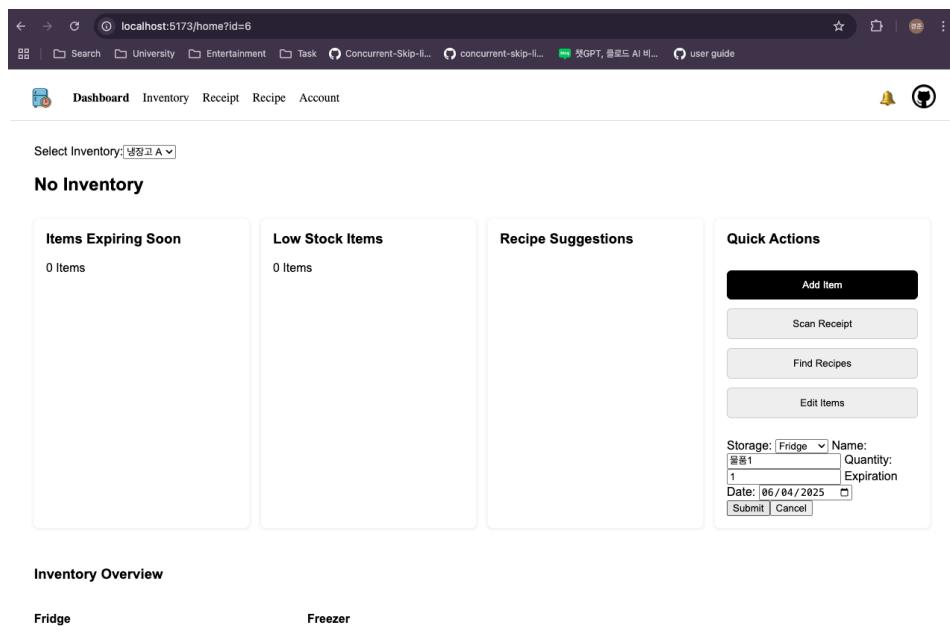


Figure 24: 새로운 물품 등록 (“물품 1”)

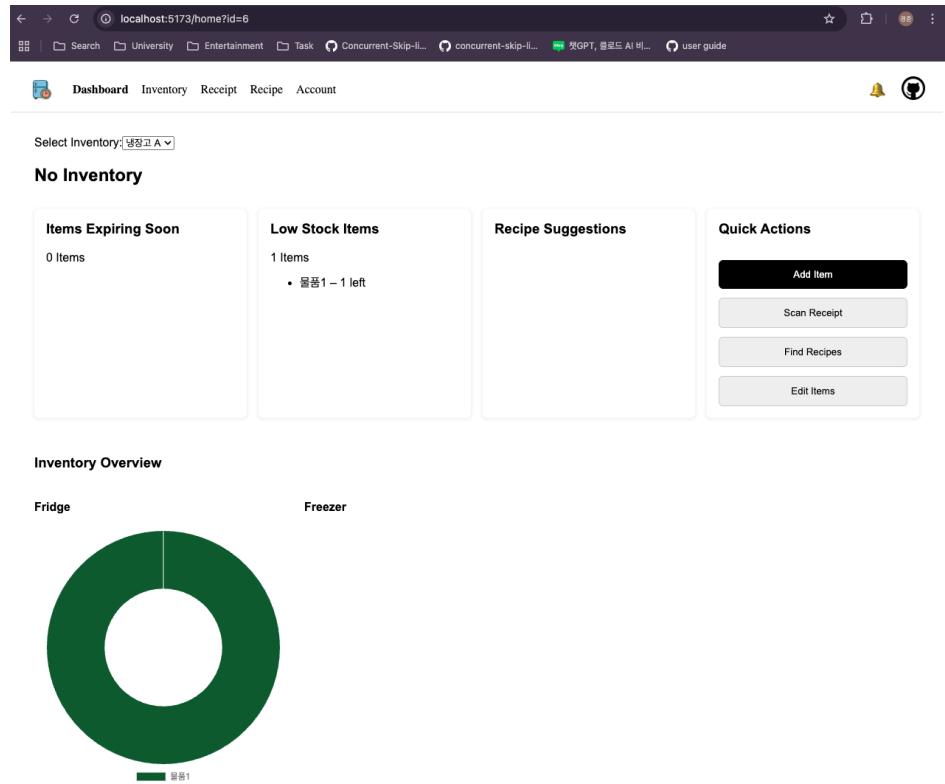


Figure 25: 인벤토리에 등록된 물품 조회

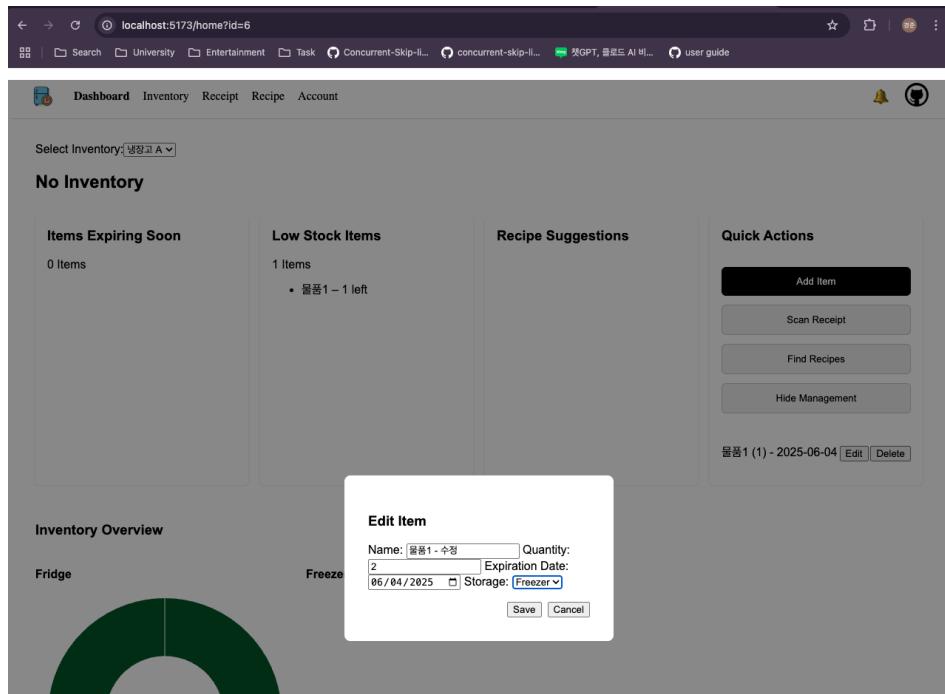


Figure 26: 등록된 물품 중 "물품 1"에 대한 정보 수정 화면

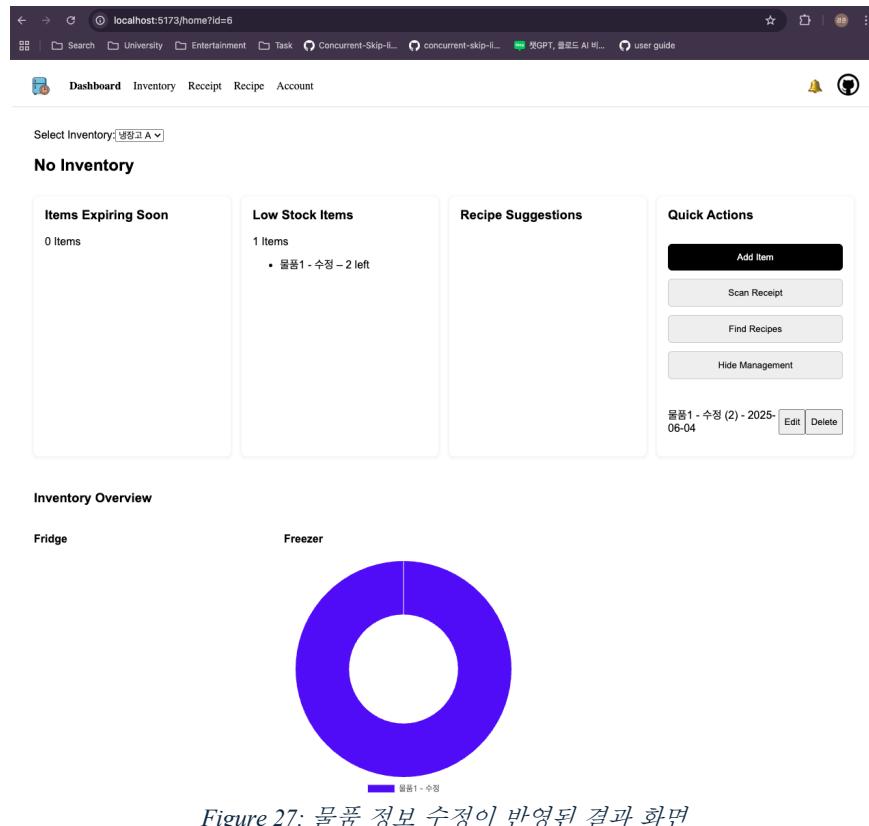


Figure 27: 물품 정보 수정이 반영된 결과 화면

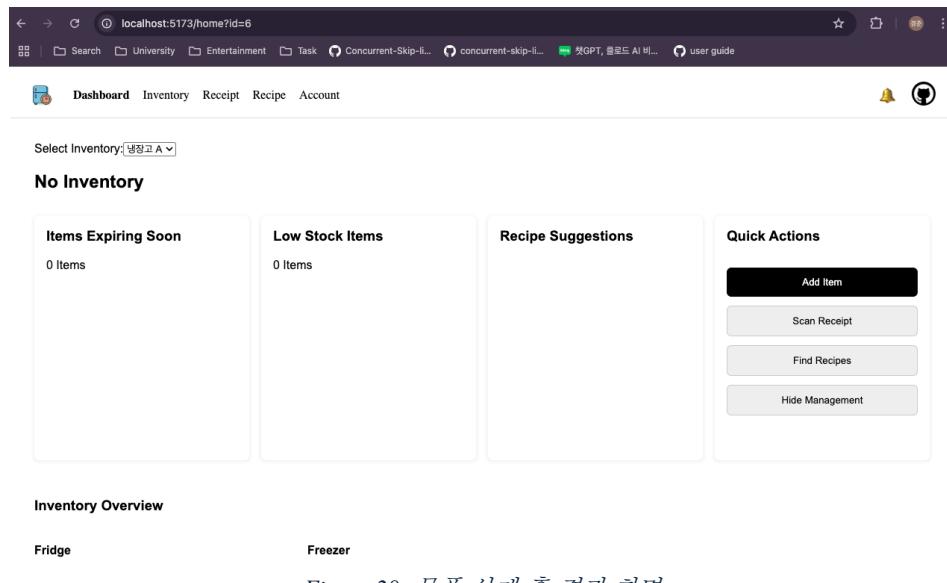


Figure 28: 물품 삭제 후 결과 화면

사용자는 하나 이상의 인벤토리를 생성할 수 있으며, 각 인벤토리는 물품들이 등록될 수 있는 저장소 단위로서 동작한다. 인벤토리 내 물품(InventoryItem)은 수동 입력 또는 OCR

기반 반자동 등록을 통해 추가할 수 있으며, 각 항목은 물품명, 수량, 냉장/냉동 여부, 유통기한 등의 필드를 포함한다. 모든 재고는 특정 인벤토리에 반드시 소속되며, 사용자는 재고 목록을 조회하거나 항목별 수정, 삭제 작업을 수행할 수 있다. 인벤토리가 삭제되면 그 안의 모든 재고 항목 역시 함께 삭제된다(ON DELETE CASCADE).

3.3.3 Recipe Recommendation using GPT API

본 모듈은 사용자가 선택한 재고 항목을 바탕으로 요리 레시피를 자동 생성하는 기능을 수행한다.

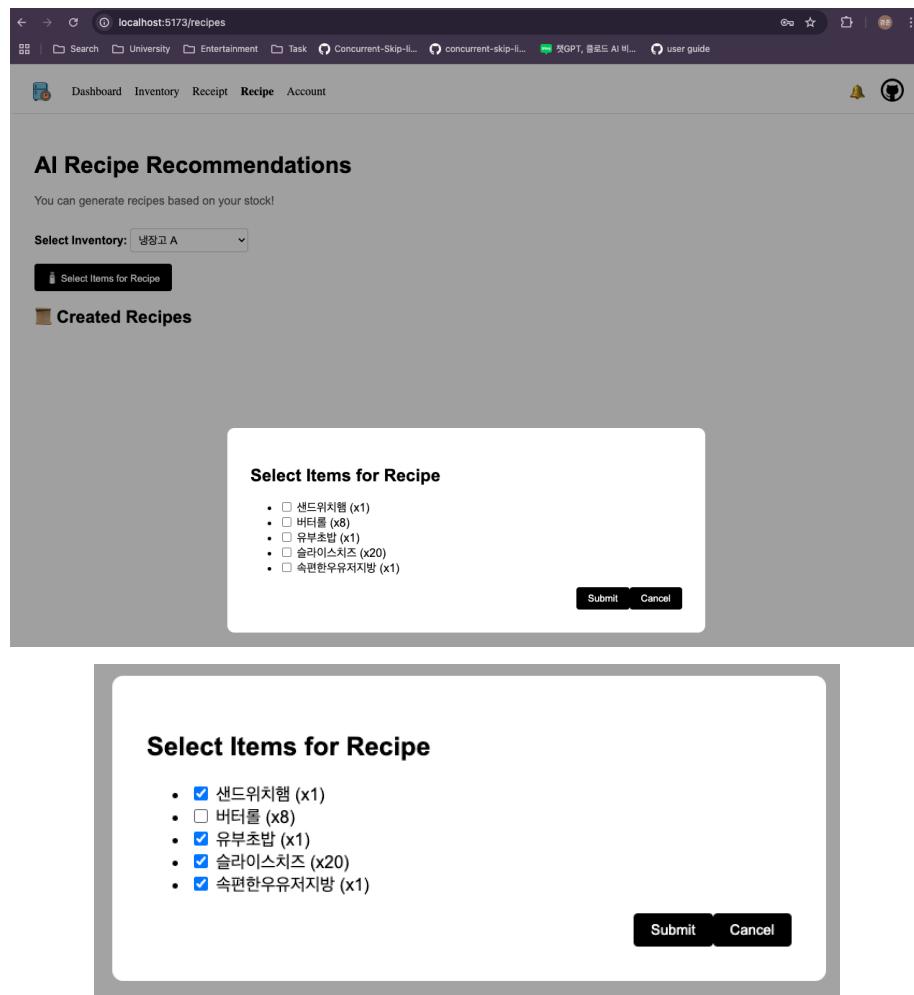


Figure 29: 냉장고 A 의 물품 중 레시피 재료로 사용할 품목 선택

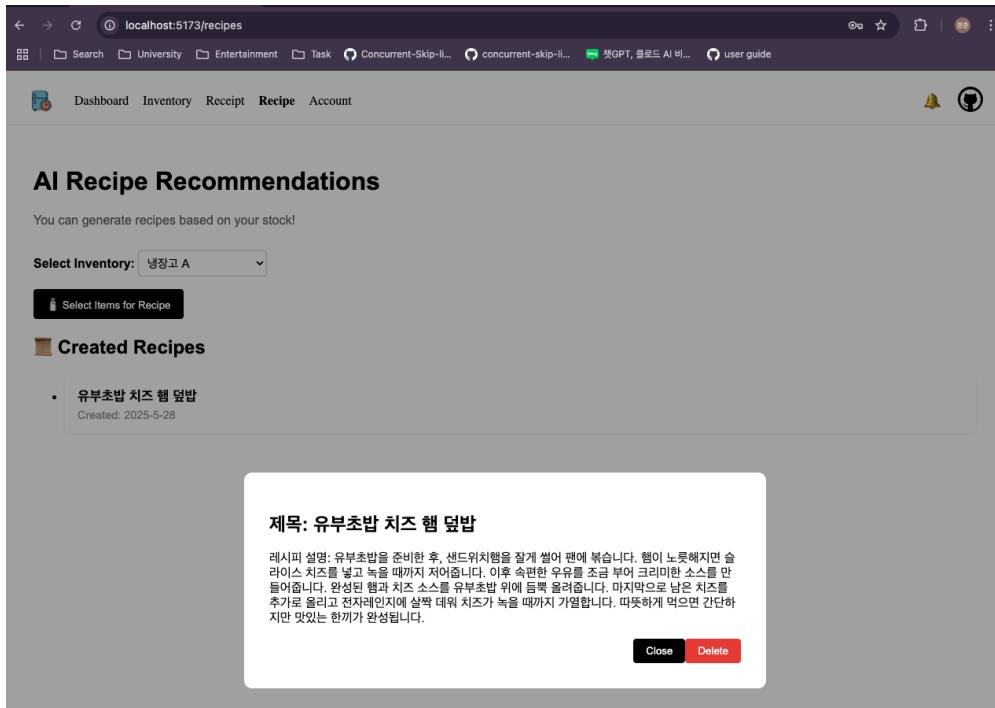


Figure 30: 레시피 생성 요청 후 결과 화면

사용자는 인벤토리 내 여러 재고 항목 중 일부를 선택하여 레시피 생성을 요청할 수 있다. BE는 선택된 항목들의 이름과 수량을 조합하여 GPT API에 전달할 프롬프트를 구성하고, 이를 바탕으로 레시피 텍스트를 생성한다. 생성된 레시피는 사용자 계정 단위로 저장되며, 관련된 재고 항목은 RecipeSourceItem 테이블을 통해 레시피와 연결된다. 사용자는 저장된 레시피를 추후 언제든 조회할 수 있다.

3.4 RESTful API Specification

본 프로젝트는 클라이언트-서버 간의 명확한 역할 분리를 위해 RESTful API 설계를 활용한다. 모든 주요 기능은 HTTP 메서드 기반의 엔드포인트를 통해 접근 가능하며, JSON 형식으로 데이터를 송수신한다.

API 명세서: 냉장고 재고 관리 및 레시피 추천 서비스

- **Base URL** : /api
- **Request Format** : application/json (단, 파일 업로드는 multipart/form-data)
- **Response Format** : application/json
- **Authentication**: 필요 시 Authorization: Bearer <token> 사용 가능
- **HTTP Status Code** 기준:
 - 200 OK — 요청 성공
 - 201 Created — 새 리소스 생성
 - 204 No Content — 성공적 삭제, 반환 없음
 - 400 Bad Request — 잘못된 입력
 - 404 Not Found — 존재하지 않는 리소스
 - 500 Internal Server Error — 서버 에러

1. 영수증 이미지 업로드

POST /receipts

- 설명: 영수증 이미지를 업로드하고 OCR 추출을 요청
- 요청 타입: multipart/form-data

Request

```
POST /receipts HTTP/1.1
Content-Type: multipart/form-data
Authorization: Bearer <token>
```

```
Form Data:
- image: [파일 업로드]
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
```

```
{
  "receipt_id": 1,
  "uploaded_at": "2025-05-22T10:00:00"
}
```

2. OCR 결과 조회

GET /receipts/{receipt_id}/parsed-items

- 설명: OCR로 추출된 품목들을 반환

Request

```
GET /receipts/1/parsed-items HTTP/1.1
Authorization: Bearer <token>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
[
  {
    "parsed_item_id": 101,
```

```
[{"item_name": "계란", "quantity": 12}, {"parsed_item_id": 102, "item_name": "우유", "quantity": 1}]
```

3. 인벤토리 CRUD

GET /inventories

- 설명: 현재 로그인한 사용자의 인벤토리 목록 조회

Request

```
GET /inventories HTTP/1.1  
Authorization: Bearer <token>
```

Response

```
[{"inventory_id": 1, "inventory_name": "냉장고 A"}]
```

POST /inventories

- 설명: 새로운 인벤토리를 생성

Request

```
POST /inventories HTTP/1.1  
Content-Type: application/json  
Authorization: Bearer <token>
```

```
{"inventory_name": "냉장고 A"}
```

Response

```
HTTP/1.1 201 Created
```

```
{"inventory_id": 1}
```

PUT /inventories/{inventory_id}

- 설명: 인벤토리 이름을 수정

Request

```
{"inventory_name": "냉장고 B"}
```

Response

HTTP/1.1 200 OK

```
{  
  "message": "Inventory updated"  
}
```

DELETE /inventories/{inventory_id}

- 설명: 인벤토리를 삭제 (소속된 재고도 함께 삭제).

Request

```
DELETE /inventories/1 HTTP/1.1  
Authorization: Bearer <token>
```

Response

HTTP/1.1 200 OK

```
{  
  "message": "Inventory and items deleted"  
}
```

4. 물품 CRUD (InventoryItem)

GET /inventories/{inventory_id}/items

| 설명 | 인벤토리에 등록된 모든 물품 조회 |

Response

```
[  
 {  
   "item_id": 1,  
   "item_name": "계란",  
   "quantity": 12,  
   "storage_type": "fridge",  
   "expiration_date": "2025-06-01"  
 }  
]
```

POST /inventory-items

- 설명: 물품 등록 (수동 입력 또는 ParsedItem 기반 입력)

Request

```
{  
  "inventory_id": 1,  
  "item_name": "계란",  
  "quantity": 10,  
  "storage_type": "fridge",  
  "expiration_date": "2025-06-01"  
}
```

Response

HTTP/1.1 201 Created

```
{  
  "item_id": 3  
}
```

PUT /inventory-items/{item_id}

- 설명: 물품 정보 수정

Request

```
{  
    "quantity": 8,  
    "storage_type": "freezer",  
    "expiration_date": "2025-07-01"  
}
```

Response

```
{  
    "message": "Item updated"  
}
```

DELETE /inventory-items/{item_id}

- 설명: 물품 삭제

Response

```
HTTP/1.1 200 OK
```

```
{  
    "message": "Item deleted"  
}
```

5. GPT 기반 레시피 추천 저장

POST /api/recipes

- AI 레시피 생성 (GPT 기반)

항목	설명
Method	POST
Auth	필요 (Authorization: Bearer <token>)
Content-Type	application/json
Description	선택된 물품명과 수량 정보를 기반으로 GPT 레시피를 하나 생성하고 DB에 저장한 뒤, 생성된 레시피를 응답

Request Body

```
[  
    { "item_name": "계란", "quantity": 2 },  
    { "item_name": "대파", "quantity": 1 },  
    ...  
]
```

Response Body

```
[  
    {  
        "recipieId": 1,  
        "title": "계란볶음밥",  
        "content": "계란과 대파를 활용한 간단한 볶음밥 레시피입니다."  
    }  
]
```

GET /api/recipes

- 사용자 레시피 목록 조회

항목	설명
Method	GET
Auth	필요 (전과 동일)

항목	설명
Description	현재 사용자 계정에 저장된 모든 "레시피 목록"을 반환

Response Body

```
[
  [
    {
      "recipieId": 1,
      "title": "계란볶음밥",
      "createdAt": "2025-05-27T12:30:00"
    },
    {
      "recipieId": 2,
      "title": "계란국",
      "createdAt": "2025-05-27T12:31:00"
    }
  ]
]
```

GET /api/recipes/{recipieId}

- 단일 레시피 상세 조회

항목	설명
Method	GET
Auth	필요
Path Param	recipieId: 레시피 ID (Long)
Description	특정 레시피의 제목, 내용, 생성일 등 상세 정보를 반환

Response Body

```
{
  "recipieId": 1,
  "title": "계란볶음밥",
  "content": "계란과 대파를 활용한 볶음밥 레시피입니다.",
  "createdAt": "2025-05-27T12:30:00"
}
```

DELETE /api/recipes/{recipieId}

- 레시피 삭제

항목	설명
Method	DELETE
Auth	필요
Path Param	recipieId: 레시피 ID (Long)
Description	특정 레시피와 관련된 정보를 삭제

Response

```
{
  "message": "레시피가 성공적으로 삭제되었습니다."
}
```

4. TESTING AND ANALYSIS

본 장에서는 FE 와 BE 연동 이후, 개발된 기능이 요구사항에 따라 정상적으로 동작하는지를 확인하기 위한 테스트 절차 및 그 결과를 설명한다.

4.1 TEST SCENARIOS AND CASES

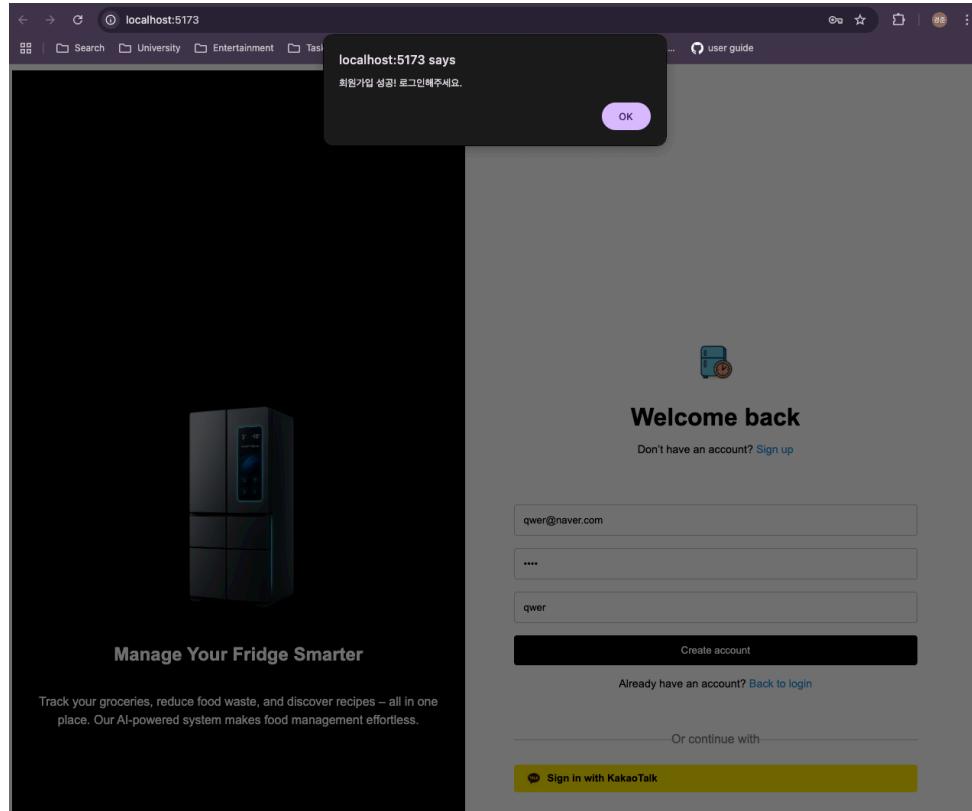
다음은 각 주요 기능에 대해 설계한 테스트 시나리오 및 테스트 케이스이다.

Function	Test scenario	Test case
회원가입/로그인	유효한 정보로 회원가입 및 로그인 시도	올바른 이메일과 비밀번호 입력 후 회원가입 레코드 및 로그인(Access token) 성공 확인
인벤토리 생성/조회	새로운 인벤토리 이름 입력 후 생성 버튼 클릭	인벤토리 테이블에 새 레코드가 추가되는지 DB 확인
인벤토리 수정	기존 인벤토리 이름을 수정 후 저장 버튼 클릭	인벤토리 테이블 레코드의 필드가 변경되는지 DB 확인
인벤토리 삭제	삭제 버튼 클릭 시 해당 인벤토리와 관련 물품 전부 삭제	인벤토리 테이블 레코드가 삭제되는지 DB 확인
인벤토리 물품 추가/조회	유효한 항목명, 수량, 저장 위치, 유통기한 입력 후 등록 클릭	인벤토리 물품 테이블에 새 레코드가 추가되는지 DB 확인
인벤토리 물품 수정	기존 물품 정보를 수정하고 저장 클릭	인벤토리 물품 테이블 레코드의 필드가 변경되는지 DB 확인
인벤토리 물품 삭제	기존 물품 삭제 버튼 클릭	인벤토리 물품 테이블 레코드가 삭제되는지 DB 확인
영수증 업로드	사용자는 영수증 이미지를 업로드	영수증 테이블에 새 레코드가 추가되는지 DB 확인
OCR 추출	업로드가 확정되면 업로드된 영수증에서 품목 정보 추출	OCR로 추출된 물품들이 ParsedItem 테이블의 레코드로서 정상적으로 추가되는지 DB 확인
레시피 생성/조회	선택된 물품들을 기반으로 하나의 레시피 생성 요청	생성된 레시피가 레시피 테이블의 새 레코드로 추가되는지 DB 확인
레시피 삭제	레시피 삭제 버튼 클릭	삭제 요청된 레시피에 대한 레코드가 레시피 테이블에서 정상적으로 삭제되는지 DB 확인

4.2 TEST RESULTS AND ANALYSIS

각 테스트 케이스는 개발 및 연동 완료 후 기능별로 실행되었으며, 모든 기능이 정상적으로 동작하고 있음을 확인하였다. 다음은 기능별 테스트 결과 및 분석이다.

- 회원가입 / 로그인



```
mysql> select * from users;
+-----+-----+-----+-----+
| created_at | user_id | email | nickname | password |
+-----+-----+-----+-----+
| 2025-05-28 21:00:26,736295 | 1 | qwer@naver.com | qwer | $2a$10$yusqj4dL9qRgCUAvQZkRG.6Y.aeanXcD./H/T7vxbamdE3H4TK77W |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 31: 회원가입 후 DB 확인

The figure consists of three vertically stacked screenshots of a web application interface.

Top Screenshot: A login page for a refrigerator management system. It features a large image of a modern four-door refrigerator on the left. In the center, there's a blue circular icon with a lock and a keyhole, followed by the text "Welcome back". Below it is a "Sign in" button. At the bottom left, there's a section titled "Manage Your Fridge Smarter" with the subtext "Track your groceries, reduce food waste, and discover recipes – all in one place. Our AI-powered system makes food management effortless." There are also "Forgot password?" and "Don't have an account? Sign up" links.

Middle Screenshot: A dashboard titled "localhost:5173/home". It includes a navigation bar with "Dashboard", "Inventory", "Receipt", "Recipe", and "Account". Below the navigation is a dropdown menu labeled "Select Inventory". The main area is titled "No Inventory". It contains four cards: "Items Expiring Soon" (0 Items), "Low Stock Items" (0 Items), "Recipe Suggestions" (empty), and "Quick Actions" (with buttons for "Add Item", "Scan Receipt", "Find Recipes", and "Edit Items").

Bottom Screenshot: A detailed view of a network request header in a browser developer tools interface. The table has columns for Name, Headers, Preview, Response, Initiator, and Timing. The Headers column shows various HTTP headers like Keep-Alive, Pragma, Transfer-Encoding, Vary, X-Content-Type-Options, X-Frame-Options, and X-Xss-Protection. The Preview column shows the raw request body: "application/json, text/plain, */*". The Response column shows the raw response body: "gzip, deflate en-US,en;q=0.9,ko;q=0.8". The Initiator column shows the URL: "http://localhost:5173". The Timing column shows the duration: "100.76.109.125:8080". The User-Agent column shows the browser information: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36".

Figure 32: 로그인 이후 Request header의 Access token이 정상적으로 삽입된 모습

회원가입/로그인 기능이 정상적으로 동작함을 확인하였으며, 정상적인 로그인 이후에는 BE에서 JWT 토큰을 발급하고 클라이언트는 이를 저장하여 인증 세션을 유지한다.

- 인벤토리 생성

The screenshot shows a web browser window for 'localhost:5173/inventory'. The navigation bar includes links for Dashboard, Inventory (which is active), Receipt, Recipe, and Account. Below the navigation is a search bar and a 'New Inventory Name' input field with a placeholder 'New Inventory Name'. A 'Add Inventory' button is located to the right of the input field. A notification icon and a user profile icon are also present.

Inventory Management

Select an inventory to view its dashboard

New Inventory Name

+ Add Inventory

냉장고 A
0 items

Edit Delete

mysql> select * from inventory;

created_at	inventory_id	user_user_id	inventory_name
2025-05-28 21:05:22.941208	1	1	냉장고 A

1 row in set (0.00 sec)

mysql>

Figure 33: 새로운 인벤토리 생성 후 DB 확인

사용자가 인벤토리 이름을 입력하고 “Add Inventory” 버튼을 누르면 새로운 인벤토리가 생성된다. DB에 정상적으로 저장됨을 확인하였다.

- 인벤토리 수정

The screenshot shows a web browser window for 'localhost:5173/inventory'. The navigation bar includes links for Dashboard, Inventory (which is active), Receipt, Recipe, and Account. Below the navigation is a search bar and a 'New Inventory Name' input field with a placeholder 'New Inventory Name'. A 'Save' button is located to the right of the input field. A 'Cancel' button is below the save button. A notification icon and a user profile icon are also present.

Inventory Management

Select an inventory to view its dashboard

New Inventory Name

Nestle - 수정

Save

Cancel

```

mysql> select * from inventory;
+-----+-----+-----+-----+
| created_at | inventory_id | user_user_id | inventory_name |
+-----+-----+-----+-----+
| 2025-05-28 21:05:22.941208 | 1 | 1 | 냉장고 A - 수정 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>

```

Figure 34: 기존 인벤토리 이름 수정 후 DB 확인

기존 인벤토리 이름을 수정한 결과가 정상적으로 반영됨을 웹 화면과 DB를 통해 확인하였다.

- 인벤토리 삭제

The screenshot shows the 'Inventory Management' page with a confirmation dialog box overlaid. The dialog box contains the text 'localhost:5173 says' and '정말 삭제하시겠습니까?' (Are you sure you want to delete?). There are 'Cancel' and 'OK' buttons at the bottom of the dialog.

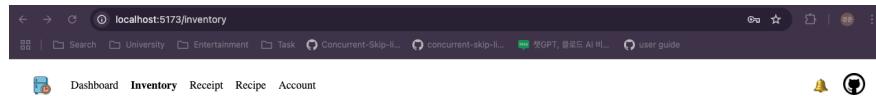


Figure 35: 인벤토리 삭제 후 결과화면

삭제 요청이 정상적으로 수행되어 해당 항목이 목록에서 제거되고, DB에서도 레코드가 삭제됨을 확인하였다.

- 인벤토리 물품 추가

Inventory Overview

Fridge

Freezer

The screenshot shows a web-based inventory management system. At the top, there's a header with a search bar, navigation links for Dashboard, Inventory, Receipt, Recipe, Account, and user notifications. Below the header, a section titled "Select Inventory" shows "냉장고 A". The main area is titled "냉장고 A" and contains four cards: "Items Expiring Soon" (0 Items), "Low Stock Items" (1 Item: 달걀 - 1 left), "Recipe Suggestions" (empty), and "Quick Actions" (Add Item, Scan Receipt, Find Recipes, Edit Items). Below these cards is a "Inventory Overview" section with a donut chart labeled "Fridge" and "Freezer", both of which are entirely blue. To the right of the chart is a legend with a blue square labeled "闺房".

```

mysql> select * from inventory;
+-----+-----+-----+-----+
| created_at | inventory_id | user_user_id | inventory_name |
+-----+-----+-----+-----+
| 2025-05-28 21:11:14.035582 | 2 | 1 | 냉장고 A |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from inventory_item;
+-----+-----+-----+-----+-----+-----+
| expiration_date | quantity | added_at | inventory_id | item_id | item_name | storage_type |
+-----+-----+-----+-----+-----+-----+
| 2025-06-04 | 1 | 2025-05-28 21:12:03.304623 | 2 | 2 | 달걀 | fridge |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Figure 36: 물품 수동 입력 및 저장 후 결과 화면

물품의 이름과 수량, 저장 위치(냉장/냉동), 유통기한을 수동으로 입력하고 “Submit 버튼”을 클릭한 후의 결과 화면이다. DB 에도 정상적으로 잘 저장됨을 확인하였다.

- 인벤토리 물품 수정

The screenshot shows an "Edit Item" dialog box. It contains fields for Name (달걀), Quantity (3), Expiration Date (06/04/2025), and Storage (Fridge). There are "Save" and "Cancel" buttons at the bottom. The background of the dialog is dark grey, and the text is white.

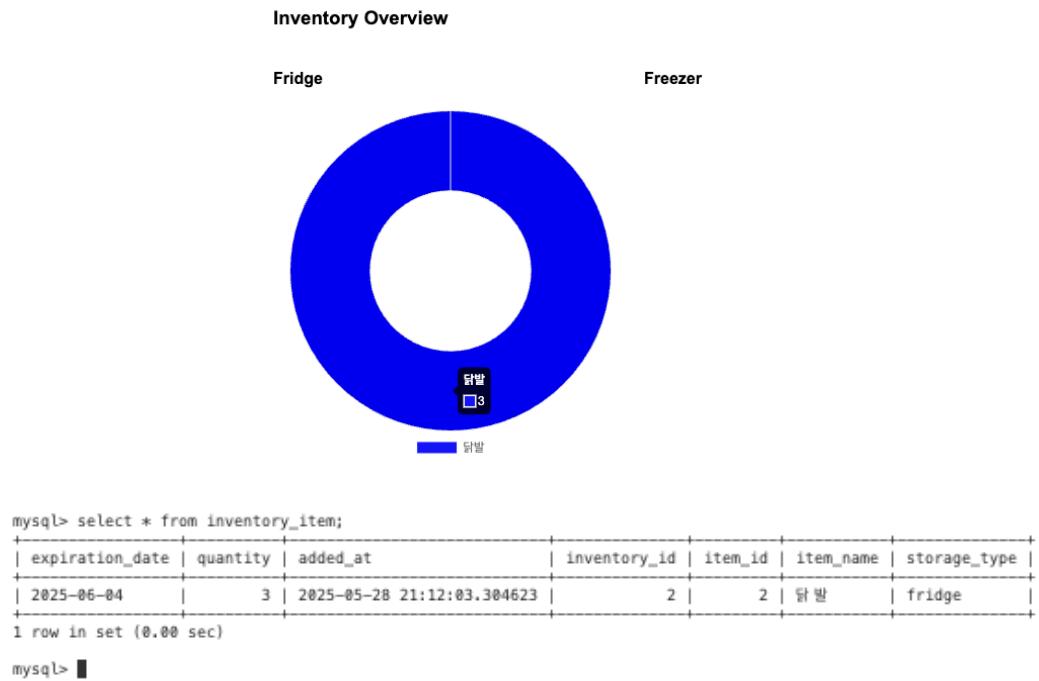
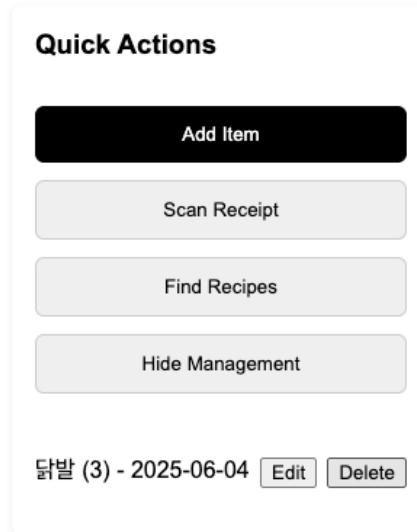


Figure 37: 기존에 등록된 물품의 수량을 1에서 3으로 수정한 후 결과 화면

사용자가 기존 물품 정보를 직접 수정하고 “Save 버튼”을 누른 후의 결과 화면이다. 수정된 물품 정보가 정상적으로 DB에 반영됨을 확인하였다.

- 인벤토리 물품 삭제



localhost:5173/home

Select Inventory: 냉장고 A

냉장고 A

Items Expiring Soon	Low Stock Items	Recipe Suggestions	Quick Actions
0 Items	0 Items		Add Item Scan Receipt Find Recipes Hide Management

Inventory Overview

Fridge Freezer

```
mysql> select * from inventory_item;
Empty set (0.00 sec)

mysql> █
```

Figure 38: 선택한 물품을 삭제 요청한 후 결과 화면

삭제 요청이 정상 수행되어 해당 항목이 목록에서 제거되고, DB에서도 삭제됨을 확인하였다.

- 영수증 업로드

localhost:5173/receipt

Dashboard Inventory Receipt Recipe Account

Upload Receipt for OCR

Drag or select a receipt image to upload and perform OCR.

Upload Receipt Image Choose File test.png

Do you want to proceed with OCR?

Yes No

Guide
001 디건강한샌드위치랩 2000114202946 1 2,190원
002 * 재사용봉투20L 200011384445 1 850원
003 서울체다SLICE치즈20 2000000017574 1 3,690원
004 The좋온나비티풀(8입) 2000059581753 1 3,290원
005 * 솔편한우유저지방 2000129100273 1 4,280원
006 주부조밥빵 2000121938498 1 3,290원

Please upload a clearly photographed receipt image.
After OCR, review and edit the extracted item information.
After editing, the items will be automatically added to your inventory.

```

mysql> select * from receipt;
+-----+-----+-----+
| receipt_id | uploaded_at | user_id | image_url
+-----+-----+-----+
| 1 | 2025-05-28 21:16:06.728147 | 1 | https://storage.googleapis.com/opensource-ssg.firebaseiostorage.app/receipts/61a66e76-8d29-4c6d-9be4-9b8c69395ce0-test.png |
+-----+-----+-----+
1 row in set (0.00 sec)

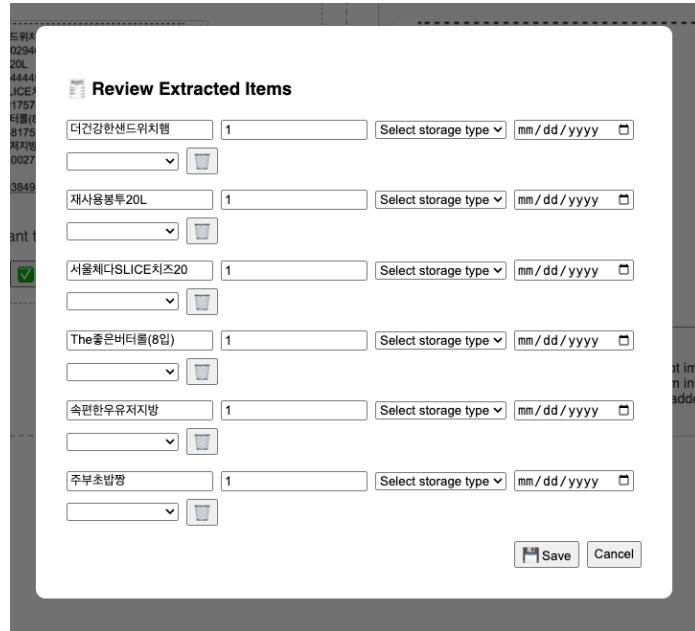
mysql>

```

Figure 39: 이미지 업로드 후 결과 화면

사용자가 이미지를 업로드하면 BE에서 이를 수신하여 Firebase DB에 저장하고 URL을 반환한다. 해당 URL은 OCR 추출 요청에서 활용하며, DB에 정상적으로 잘 기록됨을 확인하였다.

- OCR 추출



```

mysql> select * from parsed_item;
+-----+-----+-----+-----+
| quantity | parsed_item_id | receipt_id | item_name
+-----+-----+-----+-----+
| 1 | 1 | 1 | 더 건강 한 샌드 위 치 햄
| 1 | 2 | 1 | 재 사용 봉 투 20L
| 1 | 3 | 1 | 세 을 체 다 SLICE치 즈 20
| 1 | 4 | 1 | The좋 은 버 터 풀 (8입)
| 1 | 5 | 1 | 속 편 한 우 유 저 지 방
| 1 | 6 | 1 | 주 부 초 밥 빵
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

Figure 40: 업로드한 영수증 OCR 추출 후 결과 화면

사용자가 영수증 이미지를 업로드하면, BE는 이를 Firebase DB에 저장한 후 생성된 URL을 바탕으로 Naver CLOVA OCR API를 호출한다. 이 API를 통해 물품명과 수량을 추출한다. OCR 추출이 성공적으로 수행될 경우, 물품명과 수량이 정리된 JSON 파일을 클라이언트에

전달한다. 사용자는 이 정보를 바탕으로 인벤토리에 추가될 물품을 선택하거나 불필요한 물품은 제외할 수 있도록 화면을 구성한다. 선택된 항목에 대해 추가 정보를 입력함으로써 최종적으로 인벤토리에 저장된다.

Review Extracted Items

샌드위치햄	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
슬라이스치즈	20	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
버터풀	8	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
속편한우유저지방	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
유부초밥	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			

Save **Cancel**

localhost:5173 says
Save completed successfully!

Upload Receipt for OCR

Drag or select a receipt image to upload and perform OCR.

Guide

001 떠건강한샌드위치햄	1	2,190원
002 * 쇠고기등심	1	850원
003 * 슬라이스치즈	1	3,690원
004 * 헤이즐넛초콜릿	1	3,290원
005 * 치킨翅膀	1	4,280원
006 * 치킨翅膀	1	3,290원

Review Extracted Items

샌드위치햄	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
슬라이스치즈	20	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
버터풀	8	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
속편한우유저지방	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			
유부초밥	1	Fridge	06/04/2025	<input type="checkbox"/>
냉장고 A	<input type="button" value="Delete"/>			

Save **Cancel**

```

mysql> select * from inventory_item
-> ;
+-----+-----+-----+-----+-----+-----+-----+
| expiration_date | quantity | added_at           | inventory_id | item_id | item_name          | storage_type |
+-----+-----+-----+-----+-----+-----+-----+
| 2025-06-04      | 1        | 2025-05-28 21:18:56.284671 | 2             | 3       | 샌드위치 햄          | fridge       |
| 2025-06-04      | 8        | 2025-05-28 21:18:56.284671 | 2             | 4       | 버터 블레인          | fridge       |
| 2025-06-04      | 1        | 2025-05-28 21:18:56.284671 | 2             | 5       | 유부초밥            | fridge       |
| 2025-06-04      | 20       | 2025-05-28 21:18:56.284671 | 2             | 6       | 슬라이스 치즈          | fridge       |
| 2025-06-04      | 1        | 2025-05-28 21:18:56.284671 | 2             | 7       | 속편한우유저지방          | fridge       |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> 

```

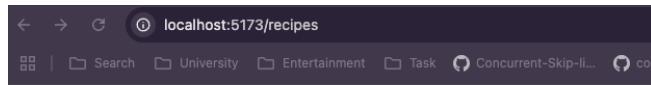
Figure 41: OCR 추출 및 사용자가 직접 입력한 물품 정보들을 바탕으로 인벤토리 물품 최종 저장 결과

- 레시피 생성/조회

The screenshot shows a web application running at localhost:5173/recipes. The main page has a header with navigation links: Dashboard, Inventory, Receipt, Recipe (which is currently selected), and Account. Below the header, there's a title "AI Recipe Recommendations" and a sub-instruction "You can generate recipes based on your stock!". A dropdown menu labeled "Select Inventory:" is set to "냉장고 A". A button labeled "Select Items for Recipe" is present. A section titled "Created Recipes" is empty. A modal window titled "Select Items for Recipe" is open in the foreground. This modal contains a list of items with checkboxes:

- 샌드위치 햄 (x1)
- 버터 블레인 (x8)
- 유부초밥 (x1)
- 슬라이스 치즈 (x20)
- 속편한우유저지방 (x1)

At the bottom of the modal are "Submit" and "Cancel" buttons.



AI Recipe Recommendations

You can generate recipes based on your stock!

Select Inventory: 냉장고 A

Select Items for Recipe

Created Recipes

- 유부초밥 치즈 햄 덮밥

Created: 2025-5-28

제목: 유부초밥 치즈 햄 덮밥

레시피 설명: 유부초밥을 준비한 후, 샌드위치행을 잘게 썰어 펜에 볶습니다. 햄이 노릇해지면 슬라이스 치즈를 넣고 녹을 때까지 저어줍니다. 이후 숨편한 우유를 조금 부어 크리미한 소스를 만들어줍니다. 완성된 햄과 치즈 소스를 유부초밥 위에 들판 올려줍니다. 마지막으로 남은 치즈를 추가로 올리고 전자레인지에 살짝 데워 치즈가 녹을 때까지 기다립니다. 따뜻하게 먹으면 간단하지만 맛있는 한끼가 완성됩니다.

Close **Delete**

```

mysql> select * from recipe;
+-----+-----+-----+
| created_at | recipe_id | user_id | content |
+-----+-----+-----+
| 2025-05-28 21:22:54,384812 | 1 | 1 | 레시피 설명: 유부초밥을 준비한 후, 샌드위치행을 잘게 썰어 펜에 볶습니다. 햄이 노릇해지면 슬라이스 치즈를 넣고 녹을 때까지 저어줍니다. 이후 숨편한 우유를 조금 부어 크리미한 소스를 만들어줍니다. 완성된 햄과 치즈 소스를 유부초밥 위에 들판 올려줍니다. 마지막으로 남은 치즈를 추가로 올리고 전자레인지에 살짝 데워 치즈가 녹을 때까지 기다립니다. 따뜻하게 먹으면 간단하지만 맛있는 한끼가 완성됩니다. |
+-----+-----+-----+
1 row in set (0.00 sec)

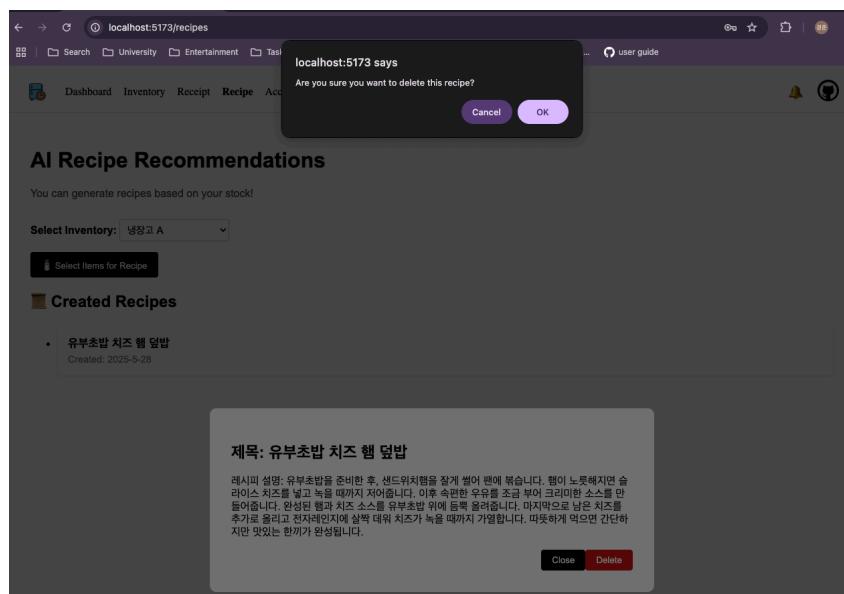
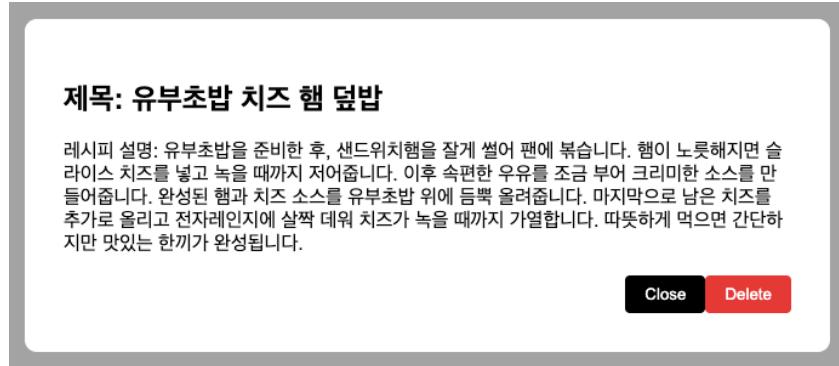
mysql>

```

Figure 42: 레시피 생성 요청 후 결과 및 조회 화면

사용자가 선택한 물품 목록을 바탕으로 레시피 생성을 요청하면, 일정 시간이 지난 후에 생성된 레시피 목록에 반영된다. DB 에도 정상적으로 저장됨을 확인하였다.

- 레시피 삭제



```
mysql> select * from recipe;
Empty set (0.00 sec)

mysql> ■
```

Figure 43: 레시피 삭제 후 결과 화면

사용자가 레시피 상세보기 창에서 삭제 버튼 클릭 시, 해당 레시피가 DB 및 UI에서 모두 제거됨을 확인하였다.

5. TERMINOLOGY AND REFERENCES

본 장에서는 프로젝트 전반에 사용된 주요 용어의 정의를 정리하고, 참고한 외부 자료와 문헌을 제시한다.

5.1 TERMS

Term	Description
OCR	Optical Character Recognition 의 약어로, 이미지 속 문자 영역을 인식하여 디지털 텍스트로 변환하는 기술이다.
Identifying / Non-identifying	테이블 간 관계를 의미한다. 비식별 관계는 자식 테이블이 독립적인 기본키를 갖고 부모 테이블과 느슨하게 연결되는 방식으로, 연결된 다른 테이블에 의해 기본키가 종속되지 않고 독립적으로 관리된다. 반면, 식별 관계는 자식 테이블의 존재가 부모 테이블에 종속되는 경우이다.
RESTful API	리소스를 URI로 표현하고 HTTP 메서드(GET, POST, PUT, DELETE 등)를 통해 CRUD 작업을 수행하는 웹 아키텍처 스타일이다.
Firebase	Google에서 제공하는 클라우드 기반 플랫폼으로, 업로드된 영수증 이미지를 URL 형식으로 저장한다.
CLOVA OCR API	네이버에서 제공하는 OCR API 서비스로, 이미지 내 한글 텍스트 인식에 최적화되어 있다.
JWT 토큰 (JSON Web Token)	사용자 인증을 위해 사용되는 토큰 기반 인증 방식으로, 로그인 시 클라이언트에 JWT를 발급하며 이후 모든 요청에 대해 해당 토큰을 사용하여 인증을 수행한다.

5.2 REFERENCES

[1] 서울특별시. (n.d.). 서울시 음식물류 폐기물 발생량 및 처리현황 통계. 서울 열린데이터광장.

[2] 환경부, & 한국환경공단. (2023). 전국 폐기물 발생 및 처리 현황.

[3] 오순도순. (2023, November 30). [OCR/AI] 2023년 최신판 OCR 8 가지 API 비교평가 테스트.

DevOcean. <https://devocean.sk.com/blog/techBoardDetail.do?ID=165524&boardType=techBlog>

[4] OpenAI. (n.d.). *Text generation and prompting*. OpenAI Platform.

<https://platform.openai.com/docs/guides/text?api-mode=chat>

[5] 유통기한 언제지. <http://ourneeds.co.kr/>

[6] seyeon-shijuan. (n.d.). *냉장고를 부탁해*. Github. <https://github.com/MultiNDjango/ndjango-django>