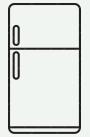


냉장고를 야금야금

AI기반 챗봇(GPT) 활용한 웹개발(Python)전문가 과정
Team. Yagumyagum
강수빈 이정임 허기범



01

프로젝트 소개

(개요 및 기획 배경)

[p.3](#)

02

프로젝트 기획

(목표 및 일정)

[p.6](#)

03

디자인 및 화면설명서

(전체적인 화면 흐름)

[p.14](#)

04

Front-end

(기능 및 코드 리뷰)

[p.25](#)

05

Back-end

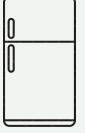
(기능 및 코드 리뷰)

[p.34](#)

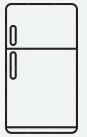
06

후기

[p.39](#)



01. 프로젝트 소개

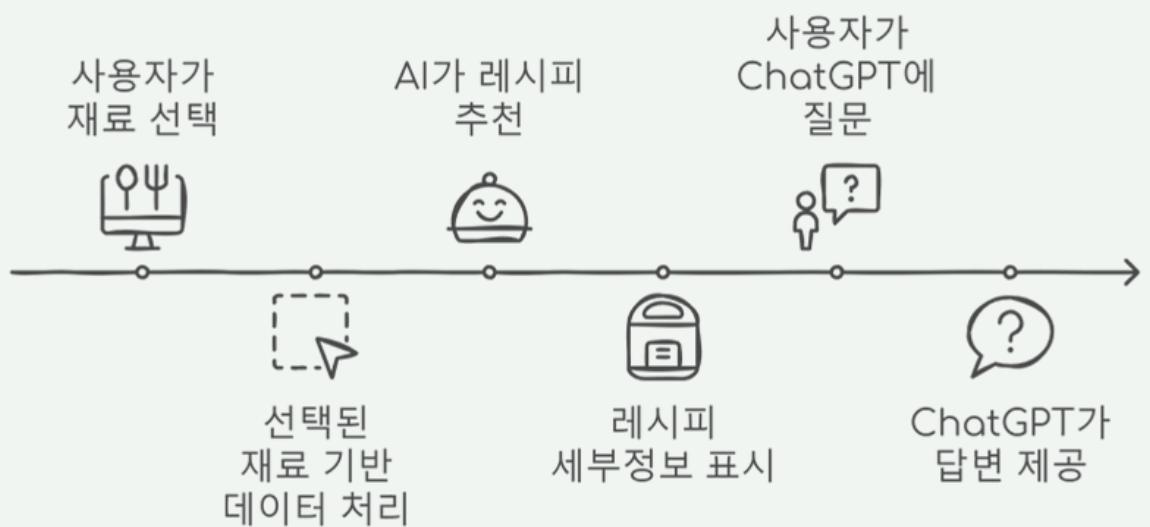


01. 프로젝트 소개

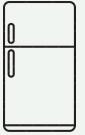
프로젝트 개요

사용자가 냉장고 속 재료를 선택하면,
해당 재료로 만들 수 있는 요리 레시피를 추천하고,
조리 과정을 시각적으로 보여주는 웹 애플리케이션입니다.

React와 Django를 기반으로 구축하였으며,
OpenAI API를 활용하여 레시피 추천과 조리 방법을 생성합니다.



추가로, 실시간 질문 기능을 활용하여
사용자가 요리 레시피를 확인하거나 조리 중에 생길 수 있는 궁금증을 즉각적으로 해결할 수 있도록
레시피 화면에 질문 입력란 및 음성 인식 기능을 설계했습니다.
사용자는 실시간으로 ChatGPT에 질문을 보내고 답변을 받을 수 있습니다.



프로젝트 기획 배경

AI 기술 활용

최신 기술을 프로젝트의 핵심 기능으로 선택하여, 사용자가 AI와 상호작용하는 경험을 자연스럽게 제공.

AI 기술의 다양한 활용 가능성에 대해 분석하고, 실질적인 효과를 경험.

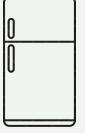
실생활 / 실용성

일상 속 실용적인 문제 해결.
요리 초보자가 조리 방법이나 대체재 정보를 쉽게 얻을 수 있도록, 요리 아이디어와 정보를 간단히 제공함으로써 일상 속 불편을 해소.
검색 시간을 줄이고 효율성을 높임.

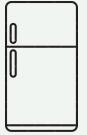
흥미로운 경험

냉장고 UI, 클릭 애니메이션, 단계별 조리 과정을 시각적으로 제공하여 사용자가 몰입감 있게 애플리케이션을 사용할 수 있도록 설계.

ChatGPT와의 실시간 대화를 통해 단순한 기능 제공을 넘어, 재미있고 유익한 경험을 제공.



02. 프로젝트 기획



페르소나



이름: 김지수 (Jisoo Kim)

나이: 29세

직업: 마케팅 기획자

취미: 요리, 플랜테리어(식물 인테리어), 전시회 관람

거주: 서울 강남구 원룸

성격:

- 새로운 트렌드에 민감하며, 창의적이고 도전적인 성격을 가짐.
- 일을 계획적으로 처리하지만, 일과 삶의 균형을 찾는데 종종 어려움을 겪음.
- 감각적인 취미 생활을 즐기며, 작은 디테일도 놓치지 않는 꼼꼼함이 특징.

최근 관심사

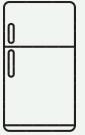
바쁜 일상 중에도 건강한 식사를 직접 준비하려고 노력하는 편이다. 평소 환경 보호와 음식물 쓰레기를 줄이는 방법에도 관심이 많기 때문에, 냉장고에 자주 남는 재료들을 최대한 효과적으로 활용하고 싶다.

고민

아무래도 바쁜 업무로 인해 직접 요리할 시간이 부족하다. 재료를 사용하지 못하고 낭비하는 일이 잦아질 땐 죄책감을 느끼지만, 요리 하기로 마음 먹고 레시피를 찾아 봐도 너무 많은 옵션 탓에 가끔은 메뉴를 선택하는 것도 스트레스다.

니즈

남은 재료로 빠르고 간단한 요리를 추천받아 시간을 절약하고 싶다. 요리가 능숙하지 않아, 요리 중에 생길 수 있는 궁금증(대체재, 조리 팁 등)을 즉각적으로 해결해줄 수 있는 기능이 있으면 좋겠다.



페르소나



이름: 이정훈 (Junghoon Lee)

나이: 35세

직업: 프리랜서 소프트웨어 엔지니어

취미: 보드게임, 다큐멘터리 시청, 캠핑

거주: 경기도 성남시 아파트

성격:

- 논리적이고 체계적인 사고를 선호하며, 기술적인 도전을 즐기는 편.
- 독립적인 성향이 강하지만, 효율성을 중요시하여 단순한 작업에는 시간을 낭비하지 않으려 함.
- 필요한 정보는 즉시 찾으려는 성격으로, 문제를 해결하는 과정에서 만족감을 느낀다.

최근 관심사

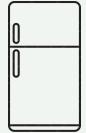
요즘 새로운 취미가 요리이다. 기존에 해보지 않은 요리를 하면서, 스킬을 조금씩 발전시키려고 한다. 요리에 소요되는 시간을 줄이고, 냉장고에 있는 재료를 최대한 활용하고 싶다.

고민

요리 경험이 부족해 냉장고에 있는 재료로 무엇을 만들어야 할지 쉽게 떠오르지 않는다. 자주 사용하는 요리 레시피는 아무래도 지루하고, 새로운 요리를 시도할 때는 실패할까 걱정이다.

니즈

간단히 재료를 선택하고, 효율적으로 요리를 계획할 수 있는 솔루션이 필요하다. 번거롭지 않게 따라할 수 있는 간단한 조리법도 많이 제공되었으면 좋겠다. 이런 기술이 있다면 평소에도 자주 사용할 것 같다.



페르소나



이름: 박소현 (Sohyun Park)

나이: 43세

직업: 두 아이의 주부

취미: 가족 요리, 가드닝, 홈카페 운영

거주: 부산 해운대구 단독주택

최근 관심사

아이들의 건강 관리가 가장 큰 관심사이다. 알레르기 정보에 맞춰, 영양소 균형이 잡힌 식단을 구성하려고 노력하고 있다. 반복적인 요리가 아니라 새로운 메뉴를 항상 고민하는 편이고, 특히 복잡한 조리 과정 없이도 맛있는 메뉴를 찾고 있다.

고민

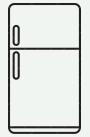
가족 모두 선호하는 음식이 달라서 한 가지 요리를 선택하는 게 가장 어렵다. 알레르기가 있는 아이를 위해서 특정 재료를 제외해야 할 때, 대체재를 찾는데 어려움을 겪는다. 요리를 매일 새롭게 구성하려니 시간이 많이 소요되어 지치기도 한다.

니즈

다양한 레시피를 추천받아 가족 모두의 입맛을 만족시키기 위해 선택할 수 있는 옵션이 많아졌으면 좋겠다. 특정 재료의 알레르기를 바로 확인할 수 있었으면 좋겠고, 보기 편하게 조리 과정이 사진으로 있으면 좋을 것 같다.

성격:

- 가족 구성원 모두의 건강과 만족을 우선시하며, 섬세하고 따뜻한 성격.
- 새로운 요리 레시피를 탐구하는 것을 즐기며, 창의적인 요리로 가족에게 즐거움을 주는 것을 중요하게 생각함.
- 관리형 성향이 강해, 주어진 자원을 최대한 활용하려는 노력을 아끼지 않음.



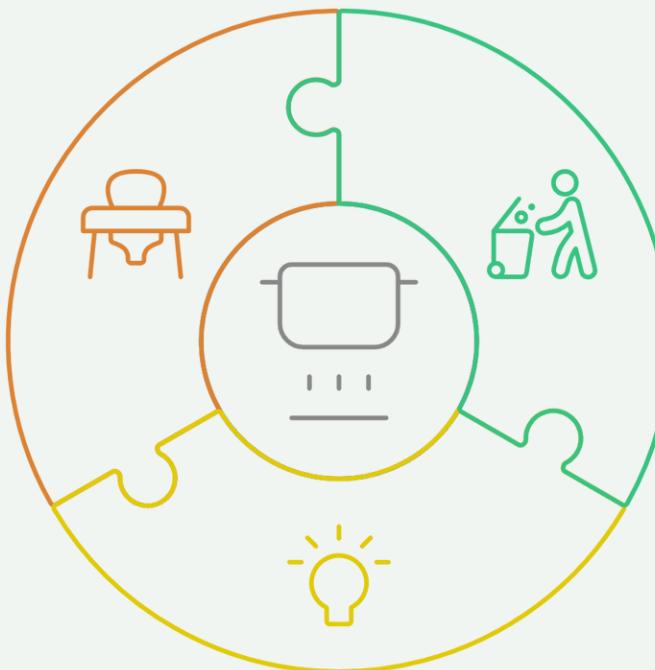
프로젝트 목표

(니즈에 따른 해결방안)

사용자의 다양한 요구와 시간 부족

바쁜 현대인들은 요리를 준비하는 과정에서
검색과 계획에 많은 시간을 소모하며,
간단하고 직관적인 솔루션을 원함.

또한 가족 구성원의 다양한 취향, 알레르기,
건강 고려 사항 때문에 한 가지 요리를
선택하기 어려움.



요리 아이디어 부족

특정 재료를 사용할 새로운 레시피를 찾는 것이 번거로움.

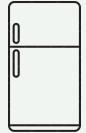
정보 부족

요리 초보자는 조리 과정에서 정확한 조리법이나 대체 재료 정보를 얻기 어려움.

재료 낭비와 음식물 쓰레기 증가

많은 사람들이 냉장고 속 남은 재료를 어떻게
활용할지 몰라 재료를 그대로 버리는 경우가 많음.

이는 음식물 쓰레기 증가와 그에 따른
환경 문제와 관련이 있기 때문에, 재료 활용을
극대화 할 수 있는 방법이 필요.



프로젝트 목표

(니즈에 따른 해결방안)

01

효율성

재료와 시간의 최대 활용

사용자가 선택한 재료를 기반으로
**AI가 즉각적이고 적합한 요리
레시피를 추천**하여 재료 낭비를 줄임.

복잡한 검색 과정을 생략하고,
데이터베이스와 OpenAI API를 활용해
개인화된 요리 추천을 제공.

조리 과정을 단계별로 시각적으로
안내하여 시간을 절약하고,
요리 준비를 더 효율적으로 만듦.

02

편리함

간단하고 직관적인 경험

냉장고 UI 기반의 직관적인

인터페이스를 통해 사용자가 재료를
쉽게 선택하고 요리를 추천받도록 설계.

ChatGPT 연동 기능으로 조리 중

생길 수 있는 궁금증을 즉각
해결할 수 있도록 지원.

레시피 추천에 **알레르기 정보**와

칼로리 정보를 포함하여

사용자 맞춤형 건강한 요리 경험을 제공.

03

재미와 자신감

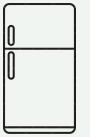
요리를 즐겁게 만드는 요소

몰입감 있는 시각적 조리 안내와

냉장고 UI, 클릭 애니메이션 등을 통해
사용자가 요리에 대한 흥미를
느끼도록 유도.

ChatGPT와의 실시간 상호작용을 통해
요리 과정에서 발생하는
두려움과 궁금증을 해소하고,
성공적인 요리 경험을 제공.

단순한 정보 전달을 넘어,
사용자가 요리 과정을 즐길 수 있는
재미있는 경험을 설계하여
요리에 대한 자신감을 높임.



02. 프로젝트 기획

프로젝트 일정

11.4



페르소나, 필요 기능
및 개선 기능 정리

11.5



화면 설계서,
기능 명세서 초안 작성

11.6 - 11.7



문서 체계화
(화면설계서, 기능명세서,
Wireframe)

11.8 - 11.10



세부 디자인 및
Concept 작업,
기본 코드 밑작업

11.11 - 11.12



Front-end 작업

11.13 - 11.14



Front-end 작업 및
Back-end 작업 진행

11.15 - 11.16



오류 수정 및
코드 Confirm

11.17

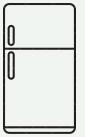


발표 문서 작업

11.18

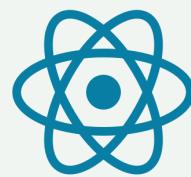


최종 점검,
프로젝트 종료



프로젝트에 사용된 기술 및 주요 라이브러리

Front-end



React



CSS

Back-end



Django

node.js

Database



MySQL

OpenAI API



chatGPT

Collaboration



github

Django

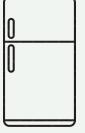
- json/os
- openAI
- mySQL

node.js

- express

React

- @fontawesome
- react-speech-kit
- react-router-dom
- react-dom

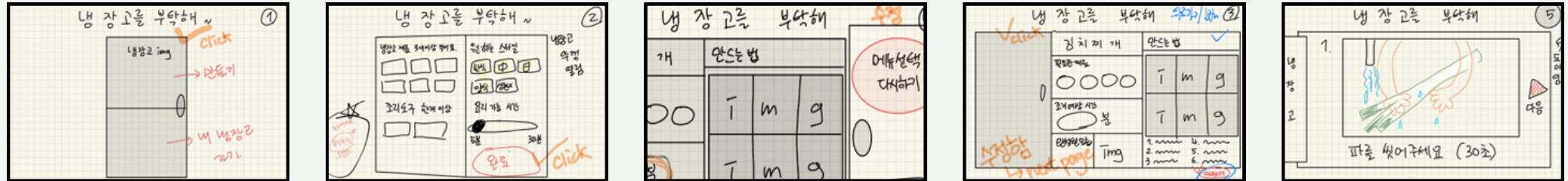


03. 디자인 및 화면 설명서

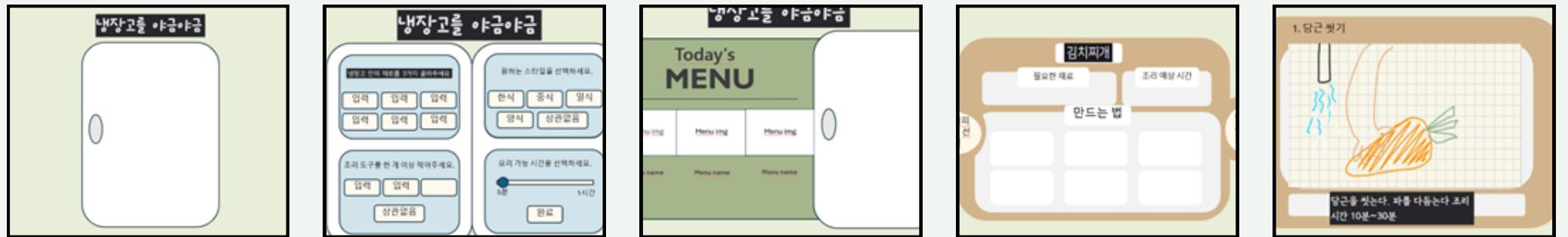


03. 디자인 및 화면 설명서

스토리 보드

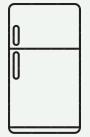


화면 설계



실제 화면





전체 프로세스

1. 냉장고 재료 선택

사용자가 냉장고에 있는 재료를 선택.
(최대 주재료 1개, 부재료 2개)

선택완료 버튼을 누르면 사용자가
선택한 데이터를 Django 서버로 전송.

2. 재료에 따른 메뉴 추천

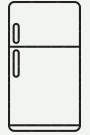
서버에서 사용자가 선택한 재료와 OpenAI의 응답에 따라
데이터베이스에 저장된 레시피 재료를 비교.
일치하는 재료들이 포함된 메뉴를 몇 가지 선정하고,
그 중 하나를 사용자에게 오늘의 메뉴로 추천.

3. 상세 레시피 설명

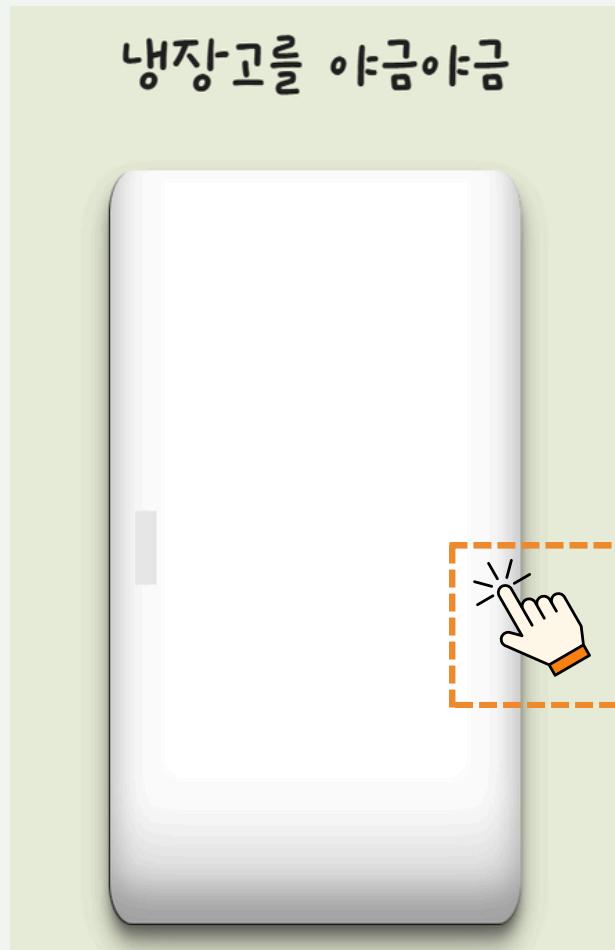
추천 레시피의 필요한 재료, 소요 시간, 상세 조리법,
알러지 및 칼로리 정보를 제공.
만드는 과정에 이미지를 포함하여 더욱 간결한
화면으로 표시.

4. 실시간 GPT 상호작용

상세 레시피 제공 화면에서 사용자가 만드는
과정 중에 궁금한 점을 실시간으로 해결할 수 있도록
동일한 페이지 내에서 GPT 상호작용 기능을 추가.
(텍스트 및 음성인식)



01. 냉장고 재료 선택



냉장고 클릭 시,
애니메이션 효과를
통해 문이 열리는
듯한 효과를 줌.

냉장고를 야금야금

주재료를 1가지 선택해주세요.

돼지고기 소고기

닭고기 생선

계란 햄

부재료를 2가지 선택해주세요.

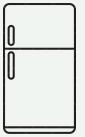
김치 감자

떡 대파

양파 마늘

선택완료

흥미롭고 직관적인 냉장고 모티브 디자인



01. 냉장고 재료 선택



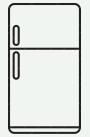
냉장고 안의 재료를 시각화

실제 냉장고와 비슷한 느낌을 내기 위해 음식 사진을 사용하여 냉장고에 재료가 들어있는 듯한 디자인을 선택함.

‘선택완료’ 버튼을 눌렀을 때의 로딩화면

GPT에게 응답을 기다리는 동안, 냉장고가 흔들리는 효과를 주어 사용자의 지루함을 해소시키고 흥미를 유발함.





02. 재료에 따른 메뉴 추천

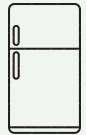


적절한 메뉴 추천

API로부터 받은 응답과 데이터베이스를 비교하여,
사용자가 선택한 재료에 따라
몇 가지 음식을 필터링하고, 그 중에 하나의 메뉴를
화면에 사진과 같이 나타내어 추천함.

‘냉장고로 돌아가기’ 버튼 클릭 시

사용자에게 적절한 메뉴가 아닐 경우,
재료 선택 화면으로 손쉽게 돌아갈 수 있게 하여,
새로운 메뉴를 추천받을 수 있게 함.



03. 디자인 및 화면 설명서

03. 상세 레시피 설명

알러지 & 칼로리

돼지고기 감자탕

냉장고로 돌아가기

필요한 재료

- 돼지고기 등뼈
- 대파
- 된장
- 소금
- 물 (필요량)
- 감자
- 다진 마늘
- 고춧가루
- 후추

조리 예상 시간

- 총 90분

만드는 법

- 돼지고기 등뼈를 찬물에 30분 정도 담가 핏물을 배줍니다.
- 핏물이 제거된 돼지고기는 끓는 물에 한번 데쳐 불순물을 제거한 후, 찬물에 행궈줍니다.
- 큰 냄비에 돼지고기 등뼈와 물을 넣고, 다진 마늘과 된장을 넣어 중불에서 1시간 정도 끓입니다.
- 끓이는 동안 감자는 깍둑썰기하고 대파는 송송 썰어 준비합니다.
- 1시간 후, 감자와 고춧가루, 소금, 후추를 넣고 다시 끓입니다. 감자가 익을 때까지 약 20분 더 끓입니다.
- 마지막에 대파를 넣고, 5분 정도 더 끓인 후 불을 꺼냅니다.

요리에 필요한 정보 전달

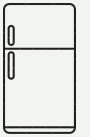
필요한 재료, 조리 예상 시간, 만드는 법을
직관적으로 디자인하여 조리 시작 전 사용자가 한눈에
정보를 확인할 수 있음.

‘음식 만들기’ 버튼 클릭 시

해당 레시피에 대한 상세 정보를 각 단계별로
사진과 함께 제공하는 페이지로 넘어감.

‘알러지 & 칼로리’ 버튼 클릭 시

해당 음식의 칼로리와 알러지에 대한
가장 기본적인 정보를 간략하게 제공함.



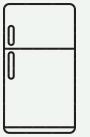
03. 상세 레시피 설명

The interface features a central image of a person cooking the dish on a stove. A callout box at the top left says '돼지고기 감자탕'. A green button at the top right says '냉장고로 돌아가기'. On the left, a circular button says '이전'. On the right, a circular button says '다음'. Below the image, a text box contains the instruction: '끓는 물에 뼈를 넣고 5분 정도 데쳐서 불순물을 제거한 후, 흐르는 물에 깨끗이 행합니다.' At the bottom, a button says 'ChatGPT에게 물어보기' with a red dashed oval around it. A red arrow points down to a footer bar. The footer bar has a text input field containing '감자 삶는 법', a microphone icon, and a green button labeled '전송'.

레시피 단계별 설명

DB에서 가져온 레시피 정보와 이미지를 가져와
동적으로 화면을 구성하여 사용자에게
실제 만드는 과정을 직관적으로 표시함.

'ChatGPT에게 물어보기' 버튼 클릭 시
사용자에게 검색창을 제공하여
실시간 응답 기능을 제공.



03. 디자인 및 화면 설명서

04. 실시간 GPT 상호작용

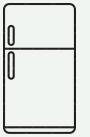


사용자가 전송 버튼을 클릭하면, 입력된 질문이 GPT 시스템으로 전송됨.

응답이 처리 되는 동안 메시지로 처리 중인 상황을 직관적으로 안내하며, 응답을 받아온 후에, 화면에 출력함.

음성 인식 기능 추가

사용자에게 음성 인식 기능을 제공하여, 더 쉬운 사용을 가능케 함.

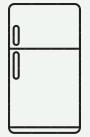


03. 디자인 및 화면 설명서

모바일 (반응형 디자인)

The diagram illustrates a responsive mobile application design for a recipe app, specifically for '냉장고를 야금야금' (Cooking with the Refrigerator). The screens are arranged horizontally, connected by orange arrows.

- Screen 1:** A large, blank white mobile phone icon on a light green background, representing the initial state of the app.
- Screen 2:** Two side-by-side mobile phone icons. The left phone shows a list of main ingredients: 돼지고기, 소고기, 닭고기, 생선, 계란, 햄. The right phone shows a list of side ingredients: 김치, 감자, 떡, 대파, 양파, 마늘. Both phones have the title '냉장고를 야금야금' at the top. Below the lists are buttons labeled '주재료를 1가지 선택해주세요.' and '부재료를 2가지 선택해주세요.' respectively.
- Screen 3:** A single mobile phone icon on a light green background. It features a large image of a dish (돼지고기 김치 감자조림) in the center. Above the image is a speech bubble containing the text '냉장고로 돌아가기'. Below the image is the dish's name: '돼지고기 김치 감자조림'. At the top of the screen is the title '냉장고를 야금야금' and below it is a section titled 'TODAY'S MENU'.



03. 디자인 및 화면 설명서

모바일 (반응형 디자인)

알려지! **돼지고기 김치 감자조림** **냉장고**

레시피 선택 **음식 만들기**

필요한 재료

| | |
|--------|------|
| - 돼지고기 | - 김치 |
| - 감자 | - 간장 |
| - 고춧가루 | - 대파 |
| - 마늘 | - 양파 |
| - 떡 | |

만드는 법

- 돼지고기를 먹기 좋은 크기로 썰고, 양파와 대파는 채 썰고 마늘은 다진다. 감자는 껍질을 벗기고 큼직하게 자른다.
- 팬에 돼지고기를 넣고 중불에서 겉면이 노릇해질 때까지 볶는다.
- 고기가 익으면 다진 마늘과 쪄간 양파, 김치를 추가하고 재료들이 잘 어우러지도록 볶는다.
- 간장과 고춧가루를 넣고 섞은 후, 감자와 떡을 추가한다.
- 재료가 삶길 정도로 물을 부은 뒤, 뚜껑을 덮고 20-25분간 끓인다.
- 감자가 부드럽게 익으면 대파를 넣고 5분 더 끓인 후, 불을 끄고 접시에 담아낸다.

돼지고기 김치 감자조림 **냉장고**

이전 **다음**

냄비나 깊은 팬에 기름을 두르고 중불에서 돼지고기를 볶아 기름을 살짝 배어나오게 합니다.

ChatGPT에게 물어보기

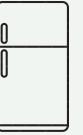
돼지고기 김치 감자조림 **냉장고**

이전 **다음**

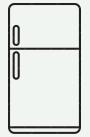
- 감자를 깨끗이 씻고 껍질을 벗깁니다.
- 큰 냄비에 물을 붓고 소금을 넣은 후 감자를 넣고 끓입니다.
- 감자가 부드러워질 때까지 15-20분 정도 삶은 후, 체에 받쳐 물기를 제거합니다.

닫기

감사합니다
음성인식 활성화 중
진행



04. Front-end



04. Front-end

“API를 통해 가지고 온 DATA를 화면에 출력”

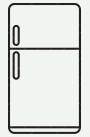
```
1  let num = 0;
2  const loadingPage = () => {
3      const text = [
4          <div className={`${styles.loading} ${styles.shake}`}>냉장고에서 재료 꺼내는 중... ● ● ○ </div>,
5          <div className={`${styles.loading} ${styles.shake}`}>chatGPT에게 메뉴 추천받는 중... ● ○ ○ </div>,
6          <div className={`${styles.loading} ${styles.shake}`}>메뉴판을 들고 오는 중... ○ ○ ● </div>,
7          <div className={`${styles.loading} ${styles.shake}`}>메뉴판 꾸미는 중... ○ ● ○ </div>,
8      ];
9
10     setLoading(text[num]);
11     num = (num + 1) % text.length;
12 };
13
14 let loadingText;
15 loadingText = setInterval(loadingPage, 1500);
16 setTimeout(() => {
17     clearInterval/loadingText);
18 }, 15000);
19
```

사용자가 선택한 재료를 받아와
GPT의 응답을 받기 전
동적으로 화면에 출력하기 위해
다양한 요소를 미리 설정함.

냉장고에서 재료 꺼내는 중... ● ● ○

chatGPT에게 메뉴 추천받는 중... ● ○ ○

1.5 초마다 인덱스 번호를 올려
setLoading을 text 인덱스 번호로
설정함.



04. Front-end

“useEffect로 jsonData를 받고 data가 있을 때 메뉴로 넘어가기”

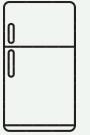
```
1  useEffect(() => {
2      console.log(data);
3      const nextPage = () => {
4          navigate('/menu')
5      }
6      if ((data != null)) {
7          let interval;
8          interval = setInterval(nextPage, 1000);
9          setTimeout(() => {
10              clearInterval(interval)
11          }, 1000);
12      }
13  }, [data])
```

```
1  const [data, setData] = useState(null);
```

```
1  const jsonData = await response.json();
2  setData(jsonData);
```

사용자가 선택한 재료를 보내고 값을 받으면 setData를 이용해서 jsonData를 저장함.

useEffect 함수를 이용하여 data 가 존재하면 다음 페이지로 넘어가게 하여 사용자가 메뉴를 받는 시간을 최소화 시킴.



04. Front-end

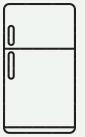
“React useEffect로 API 데이터 가져오기”

● ● ●

```
1 const [data, setData] = useState([]);
2 const [loading, setLoading] = useState(true);
3 const [error, setError] = useState(null);
4
5 useEffect(()=>{
6     const fetchData = async() =>{
7         try{
8             const response = await fetch('http://kkms4001.iptime.org:45211/api/data/');
9             if(!response.ok){
10                 throw new Error('Network response was not ok');
11             }
12             const jsonData = await response.json();
13             console.log('Fetched data:', jsonData); // 데이터 출력
14             setData(jsonData); //data에 jsonData저장
15
16         }catch(err){
17             console.error('Fetch error:', err);
18             setError(err.message);
19         }finally{
20             setLoading(false)
21         }
22     }
23     fetchData();
24 },[])
```

냉장고에서 고른 재료를 바탕으로 Django에서 전달된 데이터와 비교하여, 일치하는 메뉴 중 하나를 랜덤으로 선택받음.

```
random_choice:
allergy: "돼지고기, 대두(간장), 참깨(참기름)"
description: "돼지고기 등뼈, 감자, 대파, 다진 마늘, 원장, 고춧가루, 소금, 후추"
id: 2
image1: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/1.webp"
image2: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/2.webp"
image3: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/3.webp"
image4: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/4.webp"
image5: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/5.webp"
image6: "http://kkms4001.iptime.org:45211/static/2. 돼지고기 감자탕/6.webp"
kcal: "약 450-500 kcal"
name: "돼지고기 감자탕"
step1: "끓는 물에 뼈를 넣고 5분 정도 대쳐서 불순물을 제거한 후, 흐르는 물에 깨끗이 헹굽니다."
step2: "큰 날비에 뼈와 물을 넣고 대파, 마늘, 생강을 함께 넣어 1시간 이상 중불로 끓여 진한 육수를 만듭니다."
step3: "껍질을 벗겨 먹기 좋은 크기로 썰어 둔 감자를 날비에 같이 넣어 끓여줍니다."
step4: "고춧가루, 원장, 고추장, 간장, 다진 마늘, 다진 생강을 섞어 양념장을 만들어 넣고 감자가 익을 때까지 중불로 끓입니다."
step5: "배추나 무거지를 넣고, 모든 재료가 부드럽게 익을 때까지 더 끓여줍니다."
step6: "마지막으로 대파와 청양고추를 넣어 한 번 더 끓이고, 불을 끈 후 들깨가루와 참깨를 뿌려 고소한 맛을 더하면 완성입니다."
▶ [[Prototype]]: Object
```

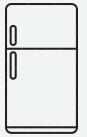


04. Front-end

“GPT 응답을 받고 화면에 출력”

```
1 // ===== 필요한 재료를 뿌리기
2 let need =[];
3 jsonData[0].cooking_details[1].forEach((v,i)=>{
4   need.push(<p className={styles.needFood}>{v}</p>)
5 })
6 // console.log(need)
7 setNeedFood(need)
8 // ===== 만드는 법 뿌리기
9 let tiny =[];
10 jsonData[0].cooking_details[0].forEach((v,i)=>{
11   tiny.push(
12     <li className={styles.makeFood}>{v}</li>
13   );
14 });
15 setTinyInFo(tiny)
16 // ===== 소요시간 뿌리기
17 setNeedTime(jsonData[0].cooking_details[2]);
18 setKcal(<p className={styles.foodDetail}>|| {jsonData[0].random_choice.kcal} || </p>);
19 setAllergy(<p className={styles.foodDetail}>▲ {jsonData[0].random_choice.allergy} ▲ </p>);
20
```

JSON 데이터를 기반으로
필요한 재료, 만드는 법, 소요 시간, 칼로리,
알레르기 정보 등을 useState를
사용하여 저장하고
이를 화면에 동적으로 렌더링함.



04. Front-end

“GPT 응답을 받고 화면에 출력”

```
● ○ ●  
1 const basicGptValue = <p className={styles.basicGptValue}>GPT 의 응답을 기다리는 중... 🐱</p>;  
2 const [gptStoredValue, setGptStoredValue] = useState(basicGptValue);  
3  
4 const chatGPT = () => {  
5   fetch('http://kkms4001.iptime.org:45211/chat_data/', {  
6     method: "POST",  
7     headers: {  
8       'Content-Type': 'application/json', // JSON 형식  
9     },  
10    body: JSON.stringify({ inputdata: gptInputValue })  
11  })  
12    .then((response) => {  
13      return response.json();  
14    })  
15    .then((data) => {  
16      console.log(data.answer);  
17      let temp = [];  
18      data.answer.forEach((v) => {  
19        temp.push(<p className={styles.gptValue}>{v}</p>);  
20      })  
21      console.log(temp);  
22      setGptStoredValue(temp);  
23    })  
24    .catch((err) => {  
25      console.error("ERROR:", err);  
26    });  
27}
```

GPT 응답을 받기 전, 초기 값 설정을 하여 사용자의 혼란을 최소화 함.

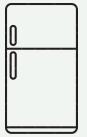
GPT 의 응답을 기다리는 중... 🐱

닫기

GPT 응답을 받으면 Data를 gptStorageValue에 담아서 화면에 나오도록 함.

- 살짝 볶는 것은 재료의 겉면이 살짝 익어 향이 나도록 하는 과정입니다.
- 일반적으로 중불에서 1-2분 정도 볶는 것이 적당합니다.
- 재료가 색이 변하거나 부드러워지기 시작할 때가 적절한 시점입니다.

닫기



04. Front-end

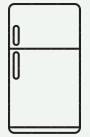
“React useEffect로 API 데이터 가져오기”

```
● ○ ■  
1  useEffect(()=>{  
2      const fetchData = async() =>{  
3          try{  
4              const response = await fetch('http://kkms4001.ptime.org:45211/api/data/')  
5              if(!response.ok){  
6                  throw new Error('Network response was not ok');  
7              }  
8              const jsonData = await response.json();  
9              console.log('Fetched data:', jsonData); // 데이터 출력  
10             setData(jsonData)  
11         }catch(err){  
12             console.error('Fetch error:', err); // 오류 출력  
13             setError(err.message); // 오류처리  
14         } finally{  
15             setLoading(false)  
16         }  
17     }  
18     fetchData();  
19 },[])
```

```
● ○ ■  
1  const images = data[0]?.random_choice ? [  
2      data[0].random_choice.image1,  
3      data[0].random_choice.image2,  
4      data[0].random_choice.image3,  
5      data[0].random_choice.image4,  
6      data[0].random_choice.image5,  
7      data[0].random_choice.image6  
8  ]: [];  
10  const steps = data[0]?.random_choice ? [  
11      data[0].random_choice.step1,  
12      data[0].random_choice.step2,  
13      data[0].random_choice.step3,  
14      data[0].random_choice.step4,  
15      data[0].random_choice.step5,  
16      data[0].random_choice.step6  
17  ]: [];  
19  
20
```

사용자가 고른 재료와 Django 서버에서 가져온 데이터를 비교하여,
랜덤으로 레시피를 추천함.

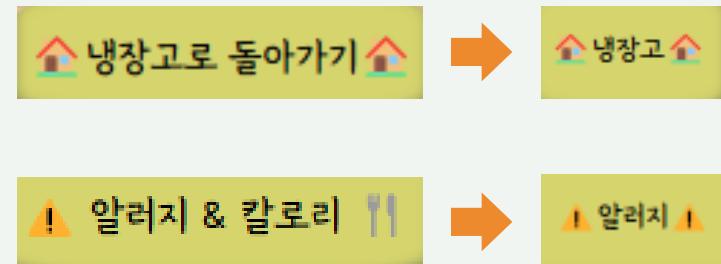
서버에서 데이터를 비동기적으로 받아와 화면에
표시하고, 조리법도 각 단계별로 동적으로 출력함.



04. Front-end

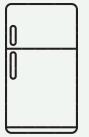
“반응형 UI 및 화면 크기 대응”

```
● ○ ●  
1 useEffect(() => {  
2     if (windowWidth > 750) {  
3         setHomeBtn('🏠 냉장고로 돌아가기 🏠');  
4         setLineHeightR('2.5');  
5     } else {  
6         setHomeBtn('🏠 냉장고 🏠');  
7     }  
8 }, [windowWidth]);  
9 useEffect(() => {  
10    if (windowWidth > 750 && next === '레시피') {  
11        setHomeBtn('🏠 냉장고로 돌아가기 🏠');  
12        setLineHeightR('1.8');  
13    } else {  
14        setLineHeightR('2.5');  
15    }  
16 }, [windowWidth, next]);
```



화면 크기에 맞추어 버튼 텍스트와 레이아웃을 동적으로
변경하여 데스크탑/모바일 화면에서 모두 적절히
동작하도록 구현함.

작은 화면에서는 버튼 텍스트가 간소화되고, 레이아웃이
재구성되어 사용자에게 최적화된 UI를 제공함.

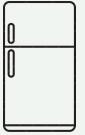


04. Front-end

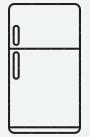
“음성 인식과 chatGPT 연동”

```
1 const { listen, listening, stop } = useSpeechRecognition({
2     onResult: (result) => {
3         setValue(result); //value에 음성 인식 결과 저장
4         setGptInputValue(result); //GPT에 전송할 값 설정
5     },
6 });
7
8 useEffect(() => {
9     if (listening) {
10         setMicState(true); // 마이크 상태 고기
11     } else {
12         setMicState(false); // 마이크 상태 켜기
13     }
14 }, [listening]);
15
16 // 음성 인식 버튼 클릭 시 동작하는 함수
17 const toggleMicState = () => {
18     if (micState) {
19         stop(); // 음성 인식 종료
20     } else {
21         listen(); // 음성 인식 시작
22     }
23};
```

```
1 //GPT에 질문 보내기
2 const chatGPT = ()=>{
3     fetch('http://kkms4001.iptime.org:45211/chat_data',{
4         method:"POST",
5         headers: {
6             'Content-Type': 'application/json',
7         },
8         body : JSON.stringify({inputdata : gpt inputValue})
9     })
10    .then((response)=>{
11        return response.json(); //GPT 응답 저장
12    })
13    .then((data)=>{
14        console.log(data.answer);
15        let temp = [];
16        data.answer.forEach((v)=>{
17            temp.push(<p className={styles.gptValue}>{v}</p>);
18        })
19        console.log(temp);
20        setGptStoredValue(temp);
21    })
22    .catch((err) => {
23        console.error("ERROR:", err);
24    });
25 }
```



05. Back-end



05. Back-end

receive_data 함수: 사용자 입력 처리 및 레시피 추천

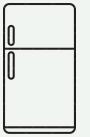
```
● ● ●  
1 @csrf_exempt  
2 def receive_data(request):  
3     try:  
4         recipes = Recipe.objects.all()  
5         total = [  
6             {  
7                 "id": recipe.id,  
8                 "name": recipe.name,  
9                 "description": recipe.description,  
10                "image1": recipe.image1,  
11                "image2": recipe.image2,  
12                "image3": recipe.image3,  
13                "image4": recipe.image4,  
14                "image5": recipe.image5,  
15                "image6": recipe.image6,  
16                "step1": recipe.step1,  
17                "step2": recipe.step2,  
18                "step3": recipe.step3,  
19                "step4": recipe.step4,  
20                "step5": recipe.step5,  
21                "step6": recipe.step6,  
22                "allergy": recipe.allergy,  
23                "kcal": recipe.kcal  
24            }  
25            for recipe in recipes  
26        ]  
27    except Exception as e:  
28        return JsonResponse({"error": "Database error"}, status=500)
```

데이터베이스에서 모든 레시피 데이터를 추출하고,
각 레시피의 주요 정보를 리스트 형태로 정리하여 저장함.

사용자로부터 전달받은 JSON 데이터를 읽고,
선택한 재료를 기준으로 데이터베이스에서 레시피를 필터링함.

필터링된 레시피 중 하나를 랜덤으로 선택함.

```
● ● ●  
1 data = []  
2 if request.method == 'POST':  
3     try:  
4         data = json.loads(request.body)  
5     except json.JSONDecodeError:  
6         return JsonResponse({"error": "Invalid JSON format"}, status=400)  
7     else:  
8         return JsonResponse({"error": "Only POST requests are allowed"}, status=405)  
9  
10  
11 filtered_total = [item for item in total if all(ingredient in item['description']  
12                                         for ingredient in data.get("main_food", []) + data.get("sub_foods", [])])]  
13 random_choice = random.choice(filtered_total) if filtered_total else {}
```



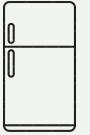
05. Back-end

receive_data 함수:
OpenAI API 상호작용

OpenAI API를 호출해
추천된 레시피에 대한
조리 과정을 생성함.

응답은 ChatGPT의
출력 내용을 적절한 형태의
데이터로 파싱하여
조리 과정, 재료 목록,
소요 시간으로 정리함.

```
1 question = ( f"""다음 재료로 '{random_choice['name']}'  
2         요리를 만드는 방법을 6단계로 간결하게 설명해 주세요:  
3             {random_choice['description']}.\n\n"""  
4         f"요리 과정 설명은 다음 형식을 따라 주세요:\n\n"  
5         f"1. '조리과정:'으로 시작하고, 각 단계를 순서대로 작성해 주세요.\n"  
6         f"2. '재료 목록:'으로 시작하고 요리에 필요한 다른 모든 재료까지 목록으로 자세히 제공해 주세요.\n"  
7         f"3. '예상 소요 시간:'으로 시작하고, 요리에 필요한 전체 시간을 분 단위로 알려주세요.\n\n"  
8         f"예시 형식:\n"  
9         f"조리과정:\n1. ...\\n2. ...\\n"\n10        f"재료 목록:\\n- 재료1\\n- 재료2\\n\\n"  
11        f"예상 소요 시간:\\n- 총 XX분" )  
12 messages = [{"role": "user", "content": question}]  
13 completion = openai.ChatCompletion.create(model="gpt-4o-mini", messages=messages)  
14 openai_response = completion.choices[0].message['content'].strip()  
15  
16 response_array = [[], [], []]  
17 steps_section, ingredients_section, time_section = False, False, False  
18  
19 for line in openai_response.split("\n"):  
20     line = line.strip()  
21     if line.startswith("조리과정:") or line.startswith("Steps:"):br/>22         steps_section, ingredients_section, time_section = True, False, False  
23         continue  
24     elif line.startswith("재료 목록:") or line.startswith("Ingredients:"):br/>25         steps_section, ingredients_section, time_section = False, True, False  
26         continue  
27     elif line.startswith("예상 소요 시간:") or line.startswith("Time:"):br/>28         steps_section, ingredients_section, time_section = False, False, True  
29         continue  
30     if steps_section and line:  
31         response_array[0].append(line)  
32     elif ingredients_section and line:  
33         response_array[1].append(line)  
34     elif time_section and line:  
35         response_array[2].append(line)
```



05. Back-end

receive_data 함수: 사용자 입력 처리 및 레시피 추천

```
● ● ●  
1 response_data = [{  
2     "random_choice": random_choice,  
3     "cooking_details": response_array  
4 }]  
5 file_path = os.path.join(settings.BASE_DIR, 'test/data/response_data.json')  
6 with open(file_path, "w", encoding='utf-8') as json_file:  
7     json.dump(response_data, json_file, ensure_ascii=False, indent = 4)  
8 return JsonResponse(response_data, safe=False)
```

데이터를 구조화하고 저장한 뒤, 이를 클라이언트에 반환함.

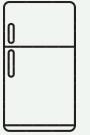
해당 결과 데이터를 다른 요청에도 재사용할 수 있도록, JSON 형태의 파일로 저장함.

get_json_data 함수: 정적 JSON 파일 제공

```
● ● ●  
1 @require_GET  
2 def get_content_json_data(request):  
3     file_path = os.path.join(settings.BASE_DIR, 'test', 'data', 'response_data.json')  
4     #Json 파일 읽기  
5     with open(file_path, 'r', encoding= 'utf-8') as file:  
6         data = json.load(file) #데이터 파싱  
7         #print(data)  
8     return JsonResponse(data, safe=False)  
9
```

로컬 파일 시스템에 저장된 JSON 데이터를 읽어와 API 형식으로 반환함.

데이터베이스를 사용하지 않는 정적 데이터 제공 시 사용함.



05. Back-end

chat_data 함수: 실시간 질문 처리

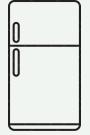
```
1 @csrf_exempt
2 def chat_data(request):
3     if request.method == 'POST':
4         data = json.loads(request.body)
5         chat_ask = f"'{data['inputdata']}'.
6         3줄로 간결하게 설명해 주세요. 예시형식: \n- ...
7         messages = [{"role": "user", "content": chat_ask}]
8
9         response = openai.ChatCompletion.create(
10             model="gpt-4o-mini",
11             messages = messages
12         )
13
14         answer = response.choices[0].message.content.strip()
15         answer_array = []
16         for line in answer.split("\n"):
17             line = line.strip()
18             if line:
19                 answer_array.append(line)
20         print(answer_array)
21         return JsonResponse({'answer': answer_array})
22
23     return JsonResponse({'error': 'Invalid request method'}, status=405)
```

사용자가 입력한 질문 데이터를 JSON 형식으로
수신하고, 이를 OpenAI API에 전달함.

OpenAI의 응답을 리스트로 파싱하여 사용자에게 반환.
요리에 관한 정보를 간결하게 제공함.



06. 후기



06. 후기 ★★★★★

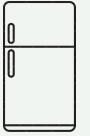
이 프로젝트는 저의 첫 팀 프로젝트이자, React와 Django를 활용한 첫 프로젝트였기 때문에 저에게는 큰 도전이었고, 매우 값진 경험이었습니다. 처음에는 팀 내에서 분업이 명확하지 않아 모두가 같은 작업을 하게 되어 일의 진행이 원활하지 않았지만, 각자의 역할을 명확히 분담하고 작업 범위를 구체적으로 나누면서 프로젝트가 점차 체계적으로 진행될 수 있었습니다.

저는 주로 Django에서 데이터를 조회 및 저장하는 작업을 담당하고, React에서 데이터를 동적으로 렌더링하는 부분을 맡았습니다. Django에서 데이터베이스에 저장된 값을 JSON 배열 형태로 받아와 React에서 동적으로 렌더링하는 방법을 배우며, React의 컴포넌트를 활용하여 UI 요소를 재사용 가능하면서 독립적으로 관리할 수 있음을 실감했습니다. 이를 통해 코드의 효율성을 크게 향상시킬 수 있었습니다. 또한, 데이터 파일 처리와 API 연결의 중요성도 깨닫고, 이를 통해 데이터 처리 능력을 한층 더 향상시킬 수 있었습니다.

OpenAI 같은 외부 API를 사용한 경험도 매우 기억에 남습니다. 이미 개발된 기능을 사용하는 것이 처음이라 신기했으며, 그 기능을 손쉽게 프로젝트에 적용할 수 있어 외부 API의 효율성과 편리함을 실감할 수 있었습니다.

이 프로젝트를 통해 실무에서 바로 활용할 수 있는 다양한 기술을 습득할 수 있었고, 프로젝트 진행 중 마주친 문제들을 함께 해결하면서 팀워크의 중요성을 다시 한번 깨달았습니다. 이번 경험은 앞으로의 개발자로서 성장하는 데 큰 밑거름이 될 것이라고 생각합니다.

마지막으로, 함께 프로젝트를 잘 마친 팀원들에게 고마운 마음을 전하고 싶습니다. 수고 많으셨습니다. 😊



06. 후기 ★★★★★

처음 기획할 당시,

냉장고에 있는 재료를 가져와 메뉴를 추천받고 만드는 법까지 알려주는 앱이 어쩌면 누군가에게 정말 도움이 될 수 있겠다는 생각에 설렘을 느꼈습니다.
이 프로젝트가 실제로 구현되었다는 점이 너무 감사하고, 뿌듯합니다.

프로그램을 처음 기획하는 과정에서 팀원들의 역할이 분명하게 나누어져 있지 않아 어렵게 느껴졌지만,

모든 코드에는 각 팀원의 흔적이 남아 있어 팀 프로젝트의 진정한 “맛”을 느낄 수 있었습니다.

또한 프로그램을 짜면서 어려운 부분들을 팀원들과 함께 고민하고 소통하면서 점차 완성해 가는 과정이 즐거웠습니다.

이번 프로젝트에서 저는

전반적인 “front-end” 작업과 반응형웹, 화면 디자인과 구조를 설계하였습니다.

“React”를 활용하여 웹 컴포넌트 기반 개발과 외부 API를 호출하고, 클라이언트와 서버 간 데이터 흐름을 구현했습니다.

“Django”를 활용하여 OpenAI의 답변을 화면에 출력하는 기능을 구현했으며, front-end와 back-end 간의 값 전달 구조를 설계했습니다.

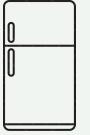
이번 프로젝트를 통해 front-end 와 back-end 의 통합적인 부분과 외부 API 활용하는 것, 클라이언트와 서버 간의 상호작용을 배울 수 있었습니다.

또한 제일 중요한 팀원들 간의 “협력”을 배웠습니다.

저 혼자의 힘으로는 만들기 어려웠을 프로젝트를 팀원들과 함께 함으로써 완성할 수 있었다고 생각합니다.

이런 값진 경험을 할 수 있게 만들어준

모든 팀원들께 감사합니다. 😊



06. 후기 ★★★★★

이 프로젝트는 제가 코딩을 배우고 처음 참여한 팀 프로젝트였습니다.

코딩이라는 업무가 팀 내에서 어떻게 분업되고 진행되는지 막연한 상태였기 때문에, 이 부분에서 가장 큰 걱정을 가지고 있었습니다.

하지만 프로젝트를 끝까지 마무리하고 결과물을 만들어내면서, 프론트엔드와 백엔드의 업무가 어떻게 분담되고,
협업이 어떻게 이루어지는지를 조금이나마 배울 수 있었습니다. 저에게는 이 경험이 이번 프로젝트를 통해 얻은 가장 큰 성과였습니다.

기능 구현의 범위나 효율적인 분업 방식에 대해 저와 팀원 모두 명확하게 알지 못했던 탓에

초반 기획 단계에서 많은 어려움을 겪었고, 그로 인해 수정 사항이 계속 발생하기도 했습니다.

하지만 서로 의견을 조율하며 방향을 잡아가려는 노력 덕분에 끝까지 잘 마무리할 수 있었습니다.

팀원들에게 너무 고맙고, 함께 노력하며 배운 과정 자체가 더욱 의미 있었던 시간이었습니다.

업무적으로는 백엔드를 주로 담당했습니다. 처음으로 백엔드 개발을 다루면서 시행착오도 많았지만,

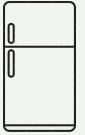
문제를 해결해 나가며 큰 성취감을 느낄 수 있었고, 실무적인 이해도를 높이는 데도 많은 도움이 되었습니다.

Django를 활용한 서버 구축, 적절한 데이터베이스 설계, 특히 OpenAI와 같은 외부 API를 처음으로 연동해 본 경험까지 모두 값지게 남았습니다.

이번 프로젝트는 저에게 코딩 기술뿐만 아니라 협업, 문제 해결 능력, 태도 등에 대해서도

더욱 깊이 생각해 볼 수 있는 기회였습니다. 이 경험을 통해 앞으로 협업이나 프로젝트를 진행할 때 더 수월하게 적응하고,

자신감을 가지고 임할 수 있을 것이라 느꼈습니다. 여러모로 소중한 경험이었습니다. 😊



감사합니다.



AI기반 챗봇(GPT) 활용한 웹개발(Python)전문가 과정

Team. Yagumyagum

강수빈 이정임 허기범