

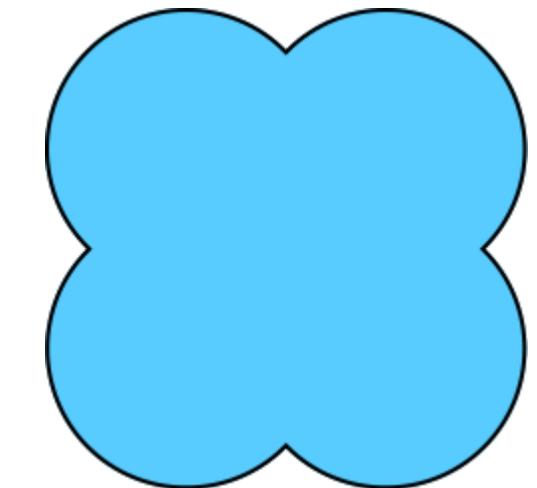
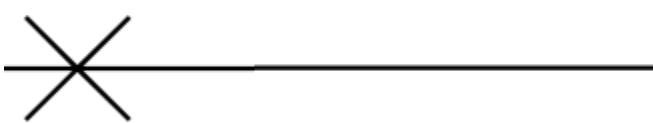


재료 맞춤형

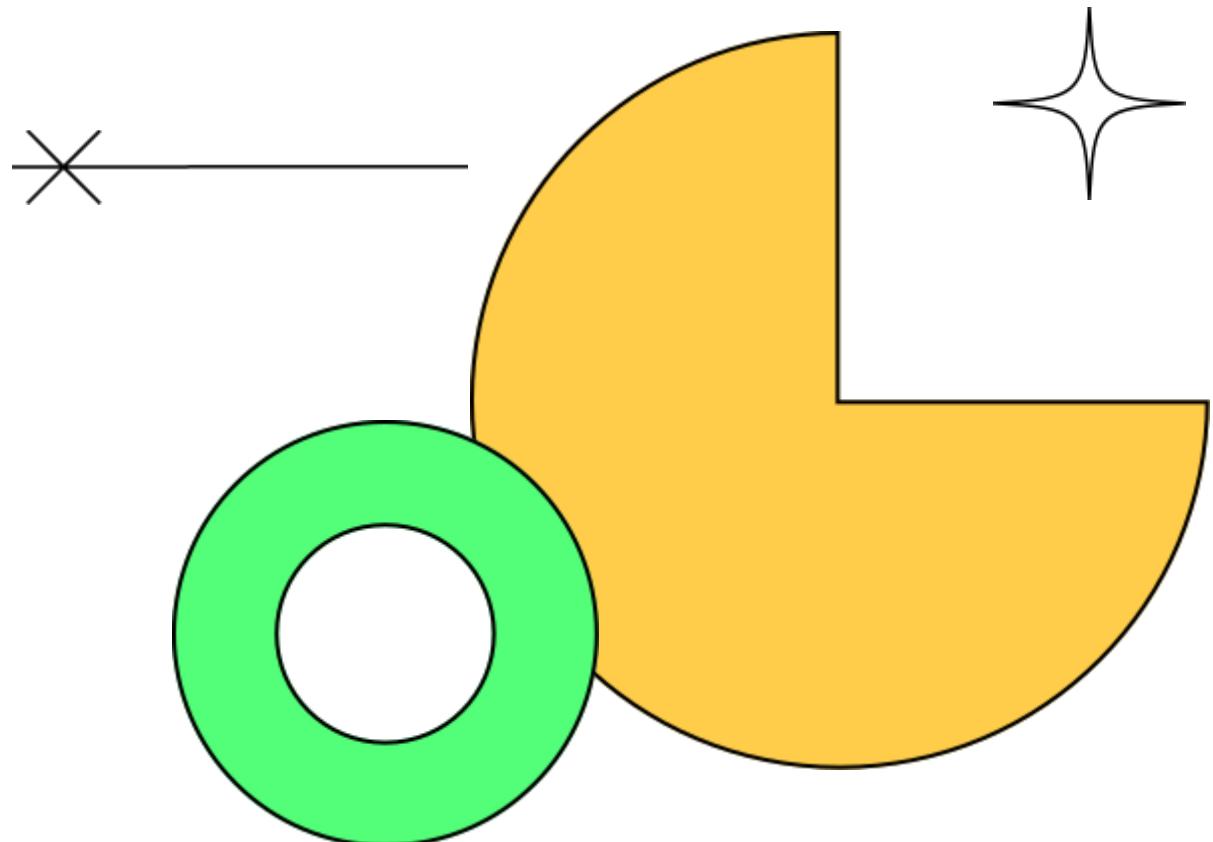


레시피 추천 서비스

2조(밥조)



# CONTENTS



## 01. 프로젝트 소개

- 문제 정의
- 프로젝트 소개

## 02. 프로젝트 개요

- 프로젝트 목표
- flow chart
- system architecture

## 03. 코드 설명

- html code
- Python code

## 04. 프로젝트 결과

- 프로젝트 결과
- 구현 영상

## 05. 트러블 슈팅

- 팀원 소개
- 소감

## 06. 소감

# 1. 프로젝트 소개

## 문제정의

재료 맞춤형 레시피 추천 서비스

항상 같은 메뉴 지겨워요



냉장고에 있는 재료로 무슨 요리를 할지 고민입니다

돈을 아껴야해요

“오늘 뭐 먹지?”  
고민 해결을 위한  
요리 추천 서비스

오늘은 그냥 외식하고 싶어요



맛집이 궁금해요



# 프로젝트 소개

오늘 뭐먹지?

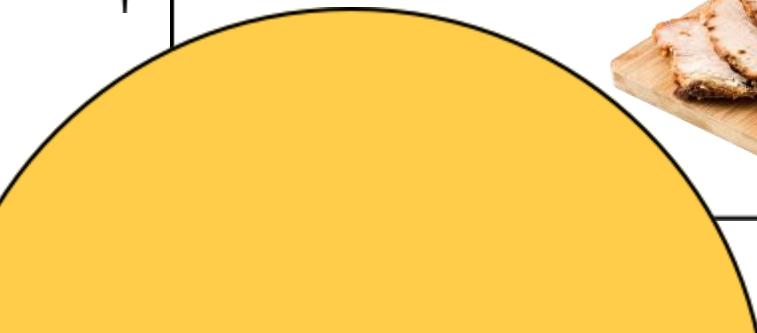
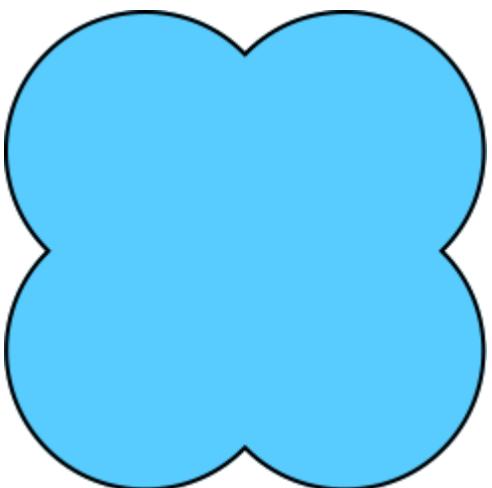
집밥부터 외식까지 한눈에 해결하는  
종합 식사 추천 서비스

물가가 오르며 외식보다 집에서 끼니를 해결하는 일이 많아졌습니다.  
하지만 냉장고 앞에서 '**오늘 뭐 먹지?**' 고민하는 일은 여전합니다.

비슷한 메뉴에 지치고,  
있는 재료로 뭘 만들 수 있을지 몰라 막막할 때..  
저희는 이런 불편함을 해결하고자 **맞춤형 식사 추천 웹페이지**를 기획했습니다.

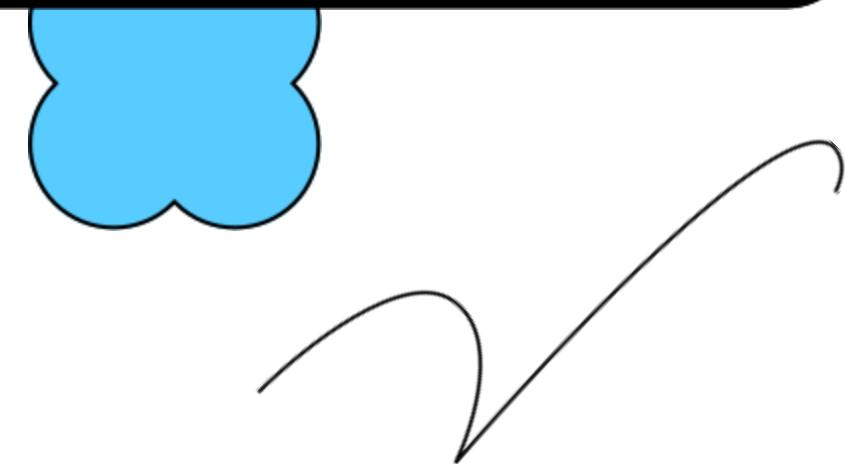
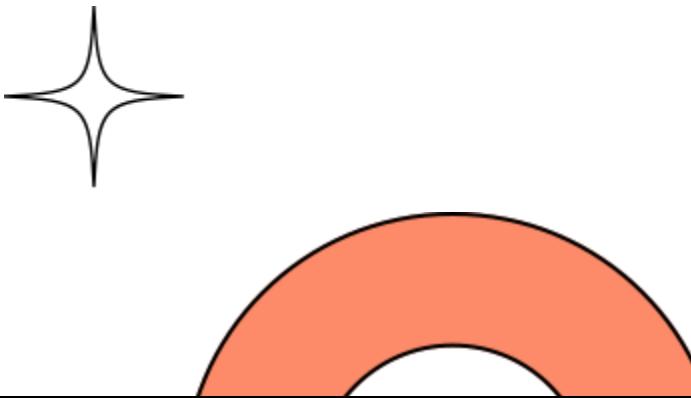
사용자가 가진 재료로 만들 수 있는 요리를 추천하고,  
부족한 재료는 실시간 구매 정보와 함께,  
레시피와 유튜브 영상도 한눈에 확인할 수 있도록 구성했습니다.

또한 집밥이 지겨운 날엔  
맛집 정보를 실시간 추천하는 기능도 제공합니다.



## 2. 프로젝트 개요

# 프로젝트 목표



## 요리 레시피 크롤링 및 DB 통합

- 1) 요리 레시피 사이트(메뉴판 닷컴)의 정보들을 크롤링을 통해 수집  
(요리 이름, 레시피, 재료, 양념, 칼로리, 이미지)
- 2) 수집한 데이터를 CSV -> DB 파일로 변환하여 데이터 통합

## Flask 서버 구축 및 웹페이지 구현

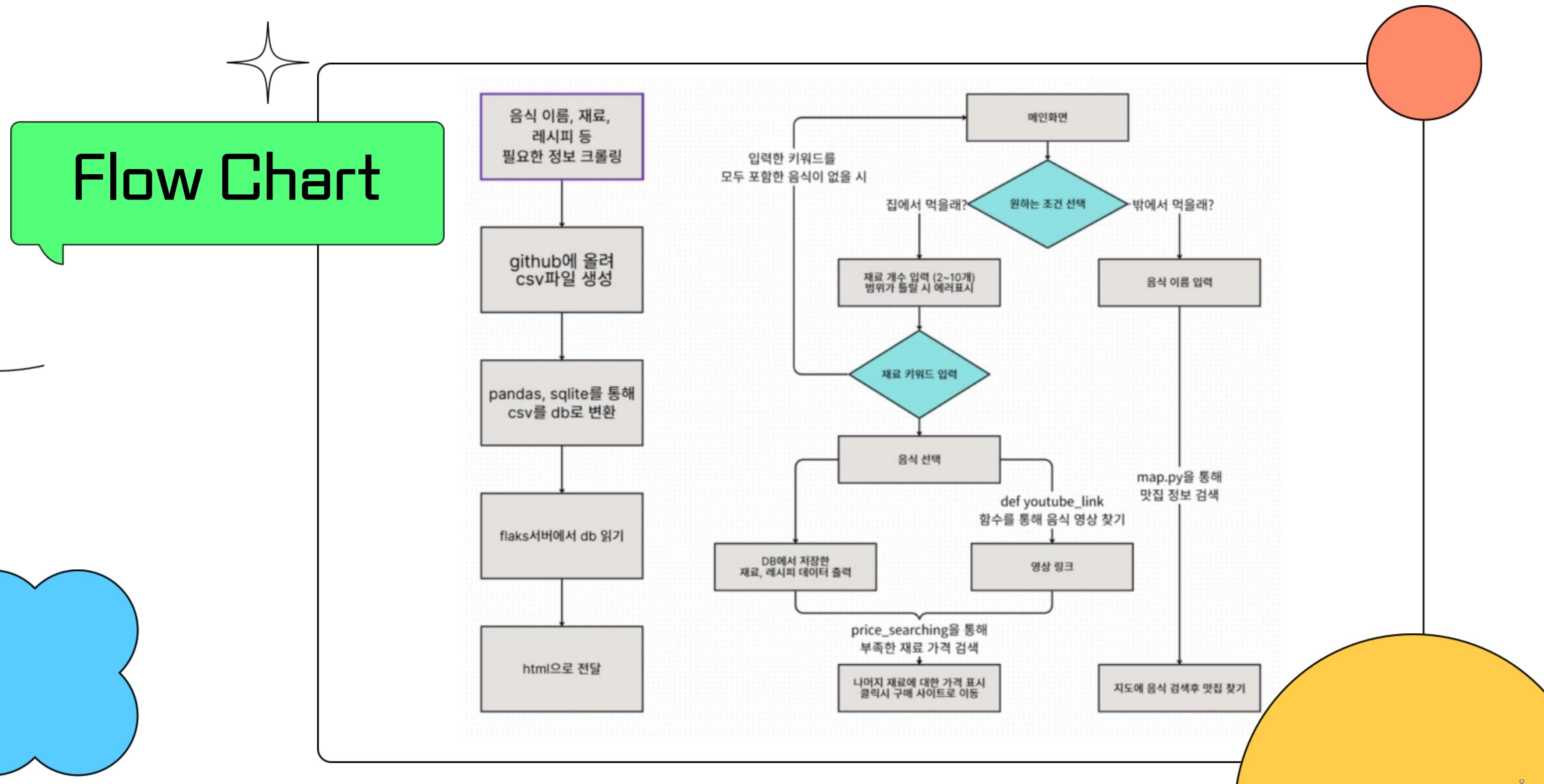
- 1) DB 로 저장된 데이터들을 웹페이지에 올리기 위해서 flask server 구축 및 경로 지정
- 2) html 문서를 사용하여 데이터 나타내기

## 실시간 정보 제공 기능 구현

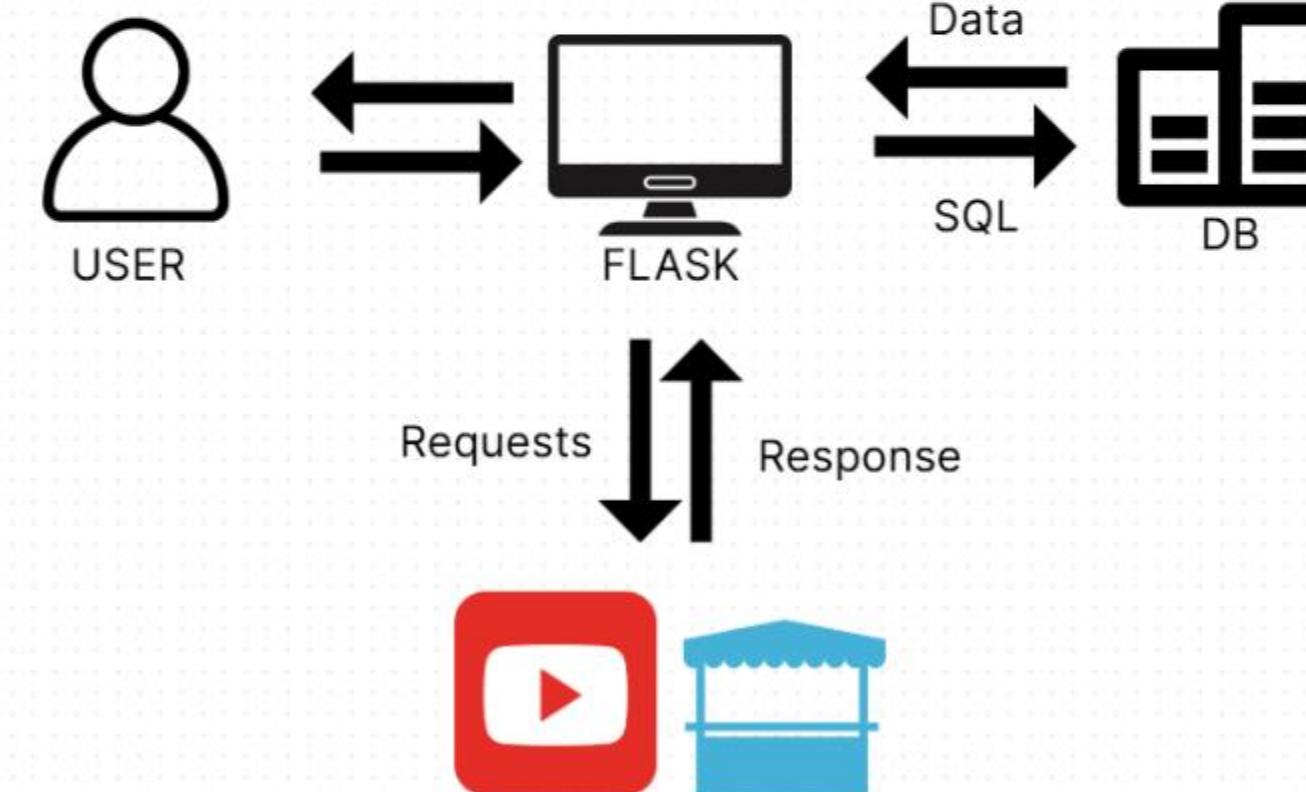
- 1) 네이버 지도에서 맛집 정보를 실시간 크롤링하여 링크 제공
- 2) 부족한 재료의 실시간 가격 및 구매처 링크 제공
- 3) 관련 유튜브 영상 크롤링을 통해 요리 시청 자료 제공

## 사용자 맞춤형 요리 추천 기능 구현

- 1) 사용자가 보유한 재료를 선택하면 만들 수 있는 음식 추천
- 2) 선택한 음식에 대해 아래 정보 제공  
전체 재료 정보 / 조리 레시피 / 유튜브 영상 링크 / 부족한 재료의 구매 링크



# System Architecture



## 3-1. 코드 설명

### [ html code ]

## HTML 구성 / base\_layout.html

```

└── templates/
    ├── base_layout.html
    ├── main_food_choice.html
    ├── home_ingredient_count_input.html
    ├── home_ingredient_details_input.html
    ├── home_dish_selection.html
    ├── home_dish_recipe_display.html
    └── missing_ingredients.html

    └── out_restaurant_recommendation.html

```

모든 웹페이지는  
Flask app.route 이용하여  
연결되도록 구성

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>{% block page_title %}음식 추천{% endblock %}</title>

    <style>
        /* 전체 바디 스타일 */
        body {
            font-family: 'Inter', 'Arial', sans-serif; /* Inter 폰트 우선, Arial 폴백 */
            background-color: #fffaf0; /* 부드러운 베이지 배경 */
            margin: 0;
            padding: 20px; /* 전체 페이지 패딩 */
            text-align: center; /* 기본 텍스트 정렬 */
            line-height: 1.6; /* 줄 간격 */
        }
    </style>

```

```

<body>
    <div class="main_container">
        {% block content %}
    {% endblock %}

    {% block footer_section %}
        <div class="page_footer">
            <a href="{{ url_for('main_page') }}" class="footer_home_link">🏠 첫 페이지로 돌아가기 </a>
        </div>
    {% endblock %}
</div>
</body>
</html>

```

모든 페이지 제일 하단에  
'첫 페이지로 돌아가기' 버튼 생성

 첫 페이지로 돌아가기

모든 페이지를  
같은 디자인으로  
구성하기 위해  
base\_layout.html  
생성

## main\_food\_choice.html

```
@app.route('/')
def main_page():
    return render_template('main_food_choice.html')
```

base\_layout.html 상속

어디에서 밥 먹을 지 결정하기

```
{% extends 'base_layout.html' %} # base_layout.html 상속 #

{% block page_title %}오늘 뭐 먹지?{% endblock %} # 페이지 제목 설정 #

{% block content %} # base_layout.html의 'content' 블록에 내용 삽입 #
<div class="content_section main_choice_section">
    <h1>오늘 뭐 먹지? 🍲</h1>
    <p class="intro_text">집에서 직접 요리할까요, 아니면 밖에서 맛있는 음식을 사 먹을까요?</p>

    <form action="{{ url_for('handle_food_choice') }}" method="post">
        <div class="button_container" style="display:flex; justify-content:center; gap:30px;">
            <button class="option_button" type="submit" name="where" value="home">
                <span class="emoji">🏠</span>
                집에서 먹을래
            </button>
            <button class="option_button" type="submit" name="where" value="out">
                <span class="emoji">🍜</span>
                밖에서 먹을래
            </button>
        </div>
    </form>
</div>
```

## 오늘 뭐 먹지? 🍲

집에서 직접 요리할까요, 아니면 밖에서 맛있는 음식을 사 먹을까요?



집에서 먹을래



밖에서 먹을래

```
{% block footer_section %}
    첫 페이지의 경우
    '첫 페이지로 돌아가기' 버튼 삭제
{% endblock %}
```

'첫 페이지로 돌아가기' 버튼 삭제

## home\_ingredient\_count\_input.html

```
@app.route('/choose', methods=['POST'])
def handle_food_choice():
    place = request.form.get('where')
    if place == 'home':
        return render_template('home_ingredient_count_input.html')
    else:
        return render_template('out_restaurant_recommendation.html', recommended_restaurants_list=None)
```

```
{% extends 'base_layout.html' %}
{% block page_title %}재료 개수 입력{% endblock %}
{% block content %}

<div class="content_section">
    <h2>☞ 사용할 재료 개수는?</h2>
    <p class="guide_text">집에 있는 재료가 몇 가지인지 숫자로 입력해주세요. (2개에서 10개 사이)</p>
    <form method="POST" action="{{ url_for('home_dish_ingredient_count_input') }}">
        <label for="ingredient_count_input" class="sr_only">재료 개수:</label>
        <input
            type="number"
            id="ingredient_count_input"
            name="ingredient_count"
            min="2"
            max="10"
            placeholder="예: 3"
            required
            aria-label="재료 개수 입력" # 접근성을 위한 레이블 #
        >
        <input type="submit" value="다음 단계로">
    </form>
    {% if error_message %}
        <p class="error_message">{{ error_message }}</p>
    {% endif %}
</div>
```

☞ 사용할 재료 개수?

집에 있는 재료가 몇 가지인지 숫자로 입력해주세요. (2개에서 10개 사이)

1

값은 2 이상이어야 합니다.

다음 단계로

첫 페이지로 돌아가기

냉장고에 있는 재료를 다양하게 이용하고  
레시피 개수를 한정시키기 위해  
재료 개수는 2-10개로 설정

## home\_ingredient\_details\_input.html

```
@app.route('/home_dish_ingredient_count_input', methods=['GET', 'POST'])
def home_dish_ingredient_count_input():
    if request.method == 'POST':
        count = int(request.form.get('ingredient_count', 0))
        if 2 <= count <= 10:
            return render_template('home_ingredient_details_input.html', ingredient_count=count)
        else:
            return render_template('home_ingredient_count_input.html', error_message="재료 개수는 2개에서 10개 사이로 입력해주세요.")
    return render_template('home_ingredient_count_input.html')
```

```
{% extends 'base_layout.html' %}
{% block page_title %}재료 입력 및 요리 선택{% endblock %}
{% block content %}

<div class="content_section">
    <h2>❶ 재료를 입력해주세요</h2>
    <p class="guide_text">집에 있는 재료들을 하나씩 입력해주세요. 입력하신 재료를 바탕으로 요리를 추천해 드립니다.</p>
    <form method="POST" action="{{ url_for('dish_selection_page') }}">
        {% for i in range(1, ingredient_count + 1) %} #{ Flask에서 전달된 'ingredient_count' 만큼 입력칸 생성 #}
            <div class="ingredient_input_group">
                <label for="ingredient_input_{{ i }}" class="sr_only">재료 {{ i }}:</label>
                <input
                    type="text"
                    id="ingredient_input_{{ i }}"
                    name="ingredient_{{ i }}"
                    placeholder="재료 {{ i }} 입력"
                    required
                    aria-label="재료 {{ i }} 입력" #{ 접근성을 위한 레이블 #
                >
            </div>
        {% endfor %}
        <input type="submit" value="요리 찾기">
    </form>
</div>
```

<집에서 먹기>선택  
- 사용할 재료명 입력하기

❶ 재료를 입력해주세요

집에 있는 재료들을 하나씩 입력해주세요. 입력하신 재료를 바탕으로 요리를 추천해 드립니다.

김치

계란

양파

❶ 재료를 입력해주세요

집에 있는 재료들을 하나씩 입력해주세요. 입력하신 재료를 바탕으로 요리를 추천해 드립니다.

백미

대추

재료 3 입력

! 이 입력란을 작성하세요.  
요리 찾기

첫 페이지로 돌아가기

반복문 이용하여  
재료 개수만큼 재료 입력칸 만들기

# home\_dish\_selection.html

```

@app.route('/dish_selection_page', methods=['POST'])
def dish_selection_page():
    user_ings = [v for k, v in request.form.items() if k.startswith('ingredient_') and v]
    normalized_user_ings_for_display = []
    for ing in user_ings:
        normalized_name = normalize_ingredient_name(ing)
        if normalized_name:
            normalized_user_ings_for_display.append(normalized_name[0])

    # 여기서 find_recipes_by_ingredients 호출
    dishes = find_recipes_by_ingredients(user_ings)
    return render_template('home_dish_selection.html',
                           input_ingredients_list=normalized_user_ings_for_display,
                           recommended_dish_list=dishes)

```

```

{% extends 'base_layout.html' %}
{% block page_title %}요리 선택{% endblock %}
{% block content %}



<h2>내가 입력한 재료:</h2>
{# 입력된 재료 목록을 쉼표로 구분 #}
<p class="input_ingredients_display">{{ input_ingredients_list | join(', ') }}</p>

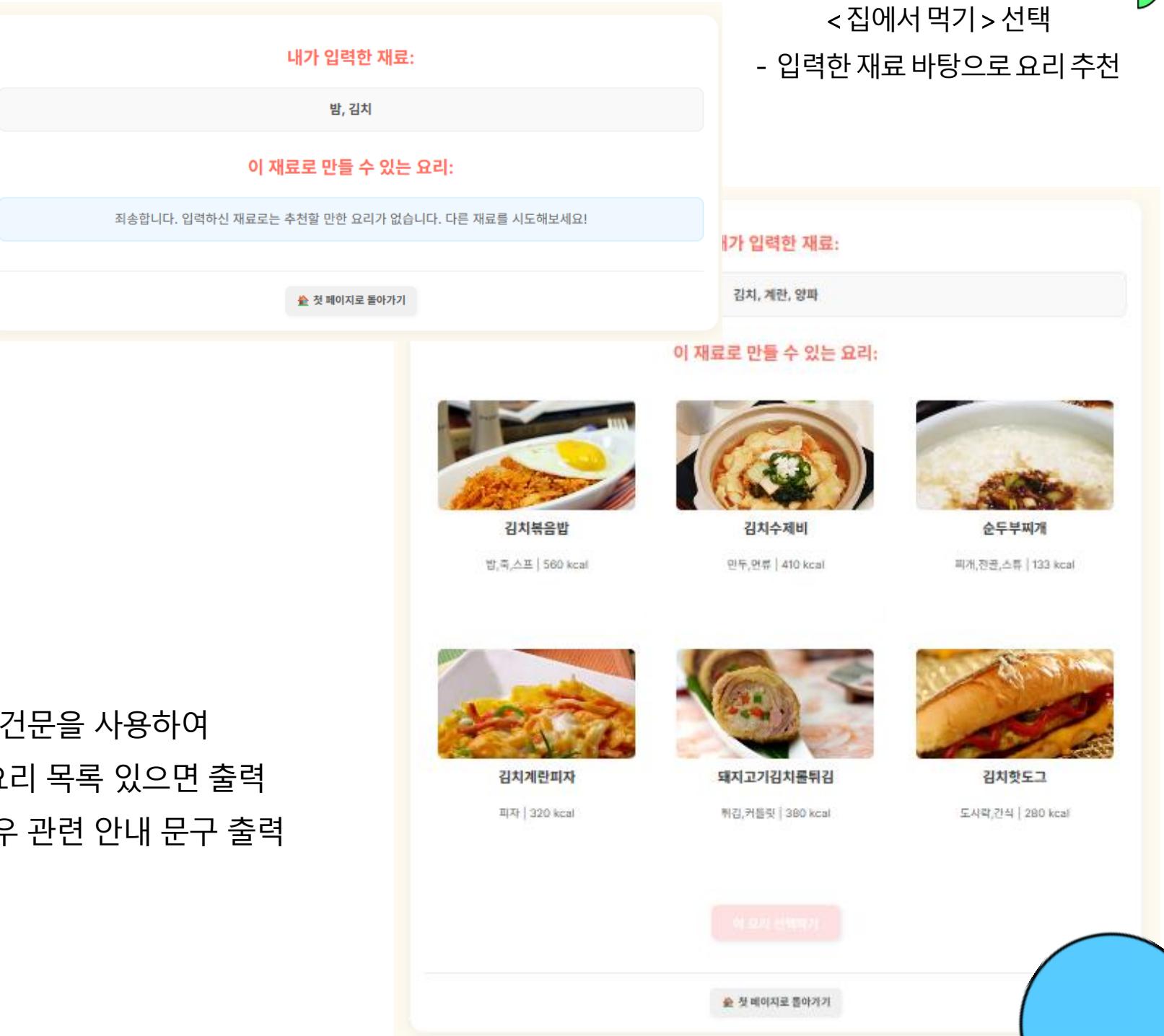
<h2>이 재료로 만들 수 있는 요리:</h2>

{# if recommended_dish_list %} {# 추천 요리 목록 o #}
<form method="POST" action="{{ url_for('view_recipe') }}">
{# 입력 재료를 hidden input으로 전달하여 다음 페이지에서 활용 #}
<input type="hidden" name="user_ingredients_str" value="{{ input_ingredients_list | join(',') }}>

<div class="dish_selection_grid">
{# for dish in recommended_dish_list %}
    <label class="dish_card_label">
        {# 라디오 버튼: name은 모두 같고, value는 요리 번호 (단일 선택) #}
        <input type="radio" name="selected_dish_number" value="{{ dish.food_number }}" required class="dish_radio_button">
        <div class="card_element dish_card"> {# base_layout.html의 카드 스타일 상속 #}
            
            <h4>{{ dish.food_name }}</h4>
            <p class="dish_type_kcal">{{ dish.types_cooking }} | {{ dish.kcal }} kcal</p> {# 음식 종류 및 칼로리 표시 #}
            <p class="dish_description">{{ dish.description }}</p>
        </div>
    </label>
{# endfor %}
</div>
<input type="submit" value="이 요리 선택하기" class="dish_select_button">
</form>
{# else %} {# 추천 요리 목록 x #}
    <p class="info_message">죄송합니다. 입력하신 재료로는 추천할 만한 요리가 없습니다. 다른 재료를 시도해보세요!</p>
{# endif %}
</div>


```

크롤링을 통해 제작한  
DB 바탕으로  
음식 이름, 설명, 사진 출력



조건문을 사용하여  
추천 요리 목록 있으면 출력  
없을 경우 관련 안내 문구 출력

## home\_dish\_recipe\_display.html

```

@app.route('/view_recipe', methods=['POST'])
def view_recipe():
    # 선택된 요리 번호 가져오기
    sel_no = int(request.form.get('selected_dish_number'))

    # 요리 정보 조회 (COOK_INGREDIENT_DATA에서 일반 정보, COOK_RECIPE_DATA에서 상세 레시피)
    general = next((d for d in COOK_INGREDIENT_DATA if d['food_number'] == sel_no), None)
    detail = next((r for r in COOK_RECIPE_DATA if r['food_number'] == sel_no), None)

    # 요리 정보 유효성 검사
    # 만약 general 또는 detail 정보가 없으면 (요리 번호가 잘못되었거나 데이터에 없는 경우)
    # 재료 개수 입력 페이지로 다시 가서 사용자에게 다시 시도하도록 함
    if not general or not detail:
        return redirect(url_for('home_dish_ingredient_count_input'))

    # 레시피 정보 통합 및 추가 데이터 준비
    # general과 detail 딕셔너리를 합쳐 하나의 recipe 딕셔너리 만들기
    recipe = {**general, **detail}
    recipe['recipe_text'] = detail['recipe']
    recipe['video_links'] = get_youtube_links_from_api(general['food_name'])
    recipe['buy_link'] = get_buy_link_from_api(general['food_name'])

    # 사용자 입력 재료 정보 준비 (세션 저장을 위해)
    user_ingredients_str = request.form.get('user_ingredients_str', '')
    user_ingredients_list_raw = [s.strip() for s in user_ingredients_str.split(',') if s.strip()]

    # 세션에 필요한 데이터 저장
    session['selected_recipe_ingredients'] = general['ingredients_list']
    session['user_provided_ingredients'] = user_ingredients_list_raw
    session['recipe_food_name'] = general['food_name']

    return render_template('home_dish_recipe_display.html', recipe=recipe)

```

- 크롤링을 통해 제작한 DB 바탕으로 음식 재료, 조리 순서 출력
- 크롤링으로 관련 유튜브 영상 불러오기 (iframe embed)

```

{% extends 'base_layout.html' %}
{% block page_title %}{{ recipe.name }} 레시피{% endblock %}
{% block content %}



## ● {{ recipe.name }} 레시피



인분: {{ recipe.serving_number }}



난이도: {{ recipe.level }}



### 재료




{% if recipe.original_ingredients_list %}
    {% for ingredient in recipe.original_ingredients_list %}
        - {{ ingredient }}

    {% endfor %}
    {% else %}
        <li>재료 목록 준비 중... </li>
    {% endif %}



### 조리 순서



{{ recipe.recipe_text | replace('\n', '  
') | safe }} # 레시피 표시 #



### 영상 보기



영상 섬네일이 안나오면 클릭을 해주세요!



{% for vid in recipe.video_links %}
    

{{ vid.title }}


{% endfor %}



모자란 재료 구입하러 가기


```

&lt;집에서 먹기&gt;선택

- 레시피 안내(재료, 조리 순서, 관련 영상)

# Flask / HTML

```
@app.route('/view_recipe', methods=['POST'])
def view_recipe():
    # 선택된 요리 번호 가져오기
    sel_no = int(request.form.get('selected_dish_number'))

    # 요리 정보 조회 (COOK_INGREDIENT_DATA에서 일반 정보, COOK_RECIPE_DATA에서 상세 레시피)
    general = next((d for d in COOK_INGREDIENT_DATA if d['food_number'] == sel_no), None)
    detail = next((r for r in COOK_RECIPE_DATA if r['food_number'] == sel_no), None)

    # 요리 정보 유효성 검사
    # 만약 general 또는 detail 정보가 없으면 (요리 번호가 잘못되었거나 데이터에 없는 경우)
    # 재료 개수 입력 페이지로 다시 가서 사용자에게 다시 시도하도록 함
    if not general or not detail:
        return redirect(url_for('home_dish_ingredient_count_input'))

    # 레시피 정보 통합 및 추가 데이터 준비
    # general과 detail 딕셔너리를 합쳐 하나의 recipe 딕셔너리 만들기
    recipe = {**general, **detail}
    recipe['recipe_text'] = detail['recipe']
    recipe['video_links'] = get_youtube_links_from_api(general['food_name'])
    recipe['buy_link'] = get_buy_link_from_api(general['food_name'])

    # 사용자 입력 재료 정보 준비 (세션 저장을 위해)
    user_ingredients_str = request.form.get('user_ingredients_str', '')
    user_ingredients_list_raw = [s.strip() for s in user_ingredients_str.split(',') if s]

    # 세션에 필요한 데이터 저장
    session['selected_recipe_ingredients'] = general['ingredients_list']
    session['user_provided_ingredients'] = user_ingredients_list_raw
    session['recipe_food_name'] = general['food_name']

    return render_template('home_dish_recipe_display.html', recipe=recipe)
```

- 크롤링을 통해 제작한 DB 바탕으로 음식 재료, 조리 순서 출력
  - 크롤링으로 관련 유튜브 영상 불러오기 (iframe embed)

# 🔍 레시피

인분: 2인분 기준

난이도: 초보환영

## 재료

✓ 김치(잘게썬것) 1/3포기

✓ 밤 2공기

✓ 감자 1/4개

✓ 양파 1/2개

✓ 햄 60g

✓ 식용유 약간

✓ 계란 2개

✓ 소금 약간

✓ 후춧가루 약간

## 조리 순서

- 김치는 1cm 크기로 썰고 햄, 양파, 감자도 1cm 정도의 크기로 잘게 썬다.
- 프라이팬에 기름을 두르고 김치와 감자를 볶다가 햄과 양파를 넣고 볶아준다.
- 양파가 어느 정도 익으면 밤을 넣고 펴듯이 볶아준다.
- 소금과 후춧가루로 간을 맞추고 계란프라이를 얹어 낸다.

## 영상 보기

영상 설명원미 만나으면 클릭해주세요!

절대 실패없는 김치볶음밥 활용레시피

돼지김치구이 레시피 #돼지김치구이 #...

김치볶음밥 그냥 놓지 말고 이과정을 거...

제일 쉬운 김치볶음밥

[모자란 재료 구입하러 가기](#)

◀ 첫 페이지로 돌아가기

# \_dish\_recipe\_display.html

## <집에서 먹기> 선택

- 레시피 안내 (재료, 조리 순서, 관련 영상)

```
place('\n', '<br>') | safe } }</p> {# 레시피 표시 #}
```

1</n>

te; encrypted-media; gyroscope; picture-in-picture"

[top link](#)

## missing\_ingredient.html

```

@app.route('/missing_ingredients')
def missing_ingredients():
    # 세션에서 필요한 데이터 가져오기
    selected_recipe_ingredients = session.pop('selected_recipe_ingredients', None)
    user_provided_ingredients = session.pop('user_provided_ingredients', None)
    recipe_food_name = session.pop('recipe_food_name', '선택된 요리')

    # 결과를 저장할 변수 초기화
    missing_ingredients_data = []
    info_message = None

    # 필수 데이터가 존재 확인
    # 만약 레시피 재료 정보와 사용자 재료 정보가 모두 세션에 존재한다면 (정상적인 접근)
    if selected_recipe_ingredients and user_provided_ingredients:
        # 사용자 보유 재료 정규화 및 소문자 집합 생성
        user_have_lower = {normalize_ingredient_name(ing)[0].lower() if normalize_ingredient_name(ing) else None for ing in user_provided_ingredients}
        # 실제 부족한 재료 목록 식별 및 부족한 재료 목록에 추가
        actual_missing_names = []
        for ingredient_in_recipe in selected_recipe_ingredients:
            if ingredient_in_recipe.lower() not in user_have_lower:
                actual_missing_names.append(ingredient_in_recipe)
        # 부족한 재료에 대한 처리 (부족한 재료 있으면 크롤링 진행)
        if actual_missing_names:
            missing_ingredients_data = price_searching(actual_missing_names)
        else:
            info_message = "모든 재료가 충분합니다! 별도로 구매할 재료가 없어요."
    else:
        # 세션 데이터가 없는 경우
        info_message = "재료 정보를 다시 불러올 수 없습니다. 레시피 페이지에서 다시 시도해주세요."
        return redirect(url_for('main_page')) # 메인 페이지로 이동

    return render_template('missing_ingredients.html',
                          missing_ingredients=missing_ingredients_data,
                          recipe_food_name=recipe_food_name,
                          info_message=info_message)

```

```

{% extends 'base_layout.html' %}
{% block page_title %}모자란 재료 구매 - {{ recipe_food_name }}{% endblock %}
{% block content %}



## "{{ recipe_food_name }} " 를(를) 만들기 위한 모자란 재료 !



이 요리를 만들려면 아래 재료들이 더 필요해요 !



{% if missing_ingredients %}
    {% for ingredient_info in missing_ingredients %}
        <div class="missing_ingredient_item">
            <div class="ingredient_name_header">
                <h4>{{ ingredient_info.name }}</h4>
            </div>
            <div class="seller_grid">
                {% for seller in ingredient_info.sellers %}
                    <a href="{{ seller.link }}" target="_blank" rel="noopener noreferrer" class="seller_card">
                        <div class="seller_image_wrapper">
                            
                        </div>
                        <div class="seller_text_content">
                            <p class="seller_type">{{ seller.type }}</p>
                            <p class="seller_price">{{ seller.price }}</p>
                            <span class="buy_link_button">구입링크</span>
                        </div>
                    </a>
                {% endfor %}
            {% else %}
                <p class="info_message">모든 재료가 충분합니다!</p>
            {% endif %}
        </div>
    {% endfor %}
{% else %}
    <p class="info_message">모든 재료가 충분합니다!</p>
{% endif %}


```

조건문 통해서

부족한 재료 있을 경우 -> 반복문으로 없는 재료 다 출력되도록 + 링크 연결

부족한 재료 없을 경우 -> 안내 문구 출력

<집에서 먹기>선택

- 모자란 재료 구매 사이트 연결

"김치볶음밥"을(를) 만들기 위한 모자란 재료!

이 요리를 만들려면 아래 재료들이 더 필요해요!

```

@app.route('/missing_ingredients')
def missing_ingredients():
    # 세션에서 필요한 데이터 가져오기
    selected_recipe_ingredients = user_provided_ingredients = session.get('selected_recipe_ingredients', None)
    recipe_food_name = session.pop('recipe_food_name', None)

    # 결과를 저장할 변수 초기화
    missing_ingredients_data = []
    info_message = None

    # 필수 데이터가 존재 확인
    # 만약 레시피 재료 정보와 사용자 제공 재료가 같은 경우
    if selected_recipe_ingredients:
        # 사용자 보유 재료 정규화 및 비교
        user_have_lower = {normalize(item['name']) for item in user_provided_ingredients}
        # 실제 부족한 재료 목록 설정
        actual_missing_names = []
        for ingredient_in_recipe in selected_recipe_ingredients:
            if ingredient_in_recipe['name'].lower() not in user_have_lower:
                actual_missing_names.append(ingredient_in_recipe['name'])

        # 부족한 재료에 대한 처리
        if actual_missing_names:
            missing_ingredients_data.append({'category': '밥', 'item': '멥쌀가루', 'price': 5250, 'link': '#'})
            missing_ingredients_data.append({'category': '감자', 'item': '깻잎', 'price': 6700, 'link': '#'})
            missing_ingredients_data.append({'category': '햄', 'item': '깻잎', 'price': 5600, 'link': '#'})
            missing_ingredients_data.append({'category': '식용유', 'item': '깻잎', 'price': 32000, 'link': '#'})
            missing_ingredients_data.append({'category': '소금', 'item': '깻잎', 'price': 13800, 'link': '#'})
            missing_ingredients_data.append({'category': '후춧가루', 'item': '깻잎', 'price': 3500, 'link': '#'})

        # 모자란 재료에 대한 처리
        if info_message is None:
            info_message = "모든 재료를 찾았습니다."
        else:
            info_message += "재료 정보를 찾았습니다."
    else:
        # 세션 데이터가 없는 경우
        info_message = "재료 정보를 찾았습니다."
    return render_template('missing_ingredients.html', missing_ingredients_data=missing_ingredients_data, info_message=info_message)

```

[첫 페이지로 돌아가기](#)

조건문 통해서

부족한 재료 있을 경우 -> 반복문으로 없는 재료 다 출력되도록 + 링크 연결

부족한 재료 없을 경우 -> 안내 문구 출력

## missing\_ingredient.html

<집에서 먹기>선택

- 모자란 재료 구매 사이트 연결

```

ends 'base_layout.html' %}
block page_title %}모자란 재료 구매 - {{ recipe_food_name }}{% endblock %}
block content %}

```

```
lass="content_section">
```

유기농 새벽을 엽니다. '첫만남을 기억하겠습니다.' [첫구매이벤트 >](#)

OASIS | OASIS ROUTE > 바른먹거리를 찾으세요? [\[목록보기\]](#) [고객센터](#) [이벤트/기획전 >](#)

로그인 회원가입 자주구매 주문/배송조회

인기 베이커리 맛집그대로 농축산물 할인지원 소상공인 찬들마루 전통식품 성남장터 정기구독

△ > 인기 > 채소

오아시스배송 후기 BEST

20% [20% 쿠폰] 국내산 두백 햅감자 (특~대, 1.7kg내외)  
포슬포슬한 식감에 고소하고 담백한 맛  
★ 4.9 리뷰 23,599  
7,500원 10% 6,700원  
28% 5,360원 (쿠폰 적용가)

농축산물 할인지원 29차 - 20% 할인 (최대 20,000원 할인) 쿠폰 적용가 5,360원

추가할인 정기배송시 3% 할인 (할인적용가 6,499원)  
진행중인 이벤트 쿠폰 모두 보기 >

결제혜택 오아시스 신현카드 결제 시 최대 13만원 캐시백 >  
카카오페이 미니 결제 시 최대 2,000원 할인 >  
토스계좌&머니 결제 시 최대 1% 즉시 할인(최대 1,000원) >  
카드사 무이자혜택 >

배송정보 배송구분 새벽배송 / 택배배송 / 당일배송  
배송비정책 5,000원 (30,000원 이상 무료)

상품정보 용량 (100g당 394원)  
보관방법 신선  
상품특징 [국내산]  
구매포인트 맛있는 감자로 반찬, 간식 등 다양한 요리로

더보기 >

오!감동 특가

4,980원 4,120원 18,900원

4.980원 4,120원 18,900원

# Flask / HTML

```
@app.route('/out_restaurant_recommendation', methods=['POST'])
def out_restaurant_recommendation():
    menu = request.form.get('menu', '')
    search_url = get_restaurants_from_api(menu) if menu else ""
    recommended_restaurants_list = []
    if search_url:
        recommended_restaurants_list.append({
            'name': f'{menu} 맛집 검색 결과',
            'address': "자세한 정보는 아래 링크를 클릭하세요.",
            'image_url': "https://placehold.co/250x150/ff6f61/ffffff?text=Click+Link", # 링크를 위한 대체 이미지
            'link': search_url # 실제 검색 결과 URL
        })
    return render_template('out_restaurant_recommendation.html', recommended_restaurants_list=recommended_restaurants_list)

if __name__ == '__main__':
    app.run(debug=True)
```

out\_restaurant\_recommendation.html

## <밖에서 먹기> 선택

```
{% extends 'base_layout.html' %}  
{% block page_title %}외식 추천{% endblock %}  
{% block content %}  
  
<div class="content_section">  
    <h2>냠냠 밖에서 먹자!</h2>  
    <p class="guide_text">어떤 메뉴를 드시고 싶으신가요? 메뉴를 입력하시면 주변 맛집을 추천해 드립니다.</p>  
  
    <form method="POST" action="{{ url_for('out_restaurant_recommendation') }}">  
        <label for="menu_input" class="sr_only">먹고 싶은 메뉴:</label>  
        <input  
            type="text"  
            id="menu_input"  
            name="menu"  
            placeholder="예: 파스타, 김치찌개, 치킨"  
            required  
            aria-label="먹고 싶은 메뉴 입력"  
        >  
        <input type="submit" value="추천 받기" />  
    </form>  
  
{% if error_message %}  
    <p class="error_message">{{ error_message }}</p>  
{% endif %}
```

## out\_restaurant\_recommendation.html



```

[% extends 'base_layout.html' %]
[% block page_title %]외식 추천[% endblock %]
[% block content %]

<div class="content_section">
    <h2>냠 밖에서 먹자!</h2>
    <p class="guide_text">어떤 메뉴를 드시고 싶으신가요? 메뉴를 입력하시면 주변 맛집을 추천해 드립니다.</p>

    <form method="POST" action="{{ url_for('out_restaurant_recommendation') }}">
        <label for="menu_input" class="sr_only">먹고 싶은 메뉴:</label>
        <input
            type="text"
            id="menu_input"
            name="menu"
            placeholder="예: 파스타, 김치찌개, 치킨"
            required
            aria-label="먹고 싶은 메뉴 입력"
        >
        <input type="submit" value="추천 받기">
    </form>

    {% if error_message %}
        <p class="error_message">{{ error_message }}</p>
    {% endif %}

    {% if recommended_restaurants_list %} #{ 추천 식당 목록 0 #}
        <h3>추천 식당 리스트</h3>
        <div class="restaurant_card_grid">
            {% for restaurant in recommended_restaurants_list %} #{ 추천 식당 목록 반복 표시 #
                <div class="card_element restaurant_card">
                    <div class="restaurant_image_container">
                        
                    </div>
                    <a href="{{ restaurant.link }}" target="_blank" rel="noopener noreferrer" class="restaurant_link_wrapper">
                        <div class="restaurant_info_text">
                            <h4>{{ restaurant.name }}</h4>
                            <p class="restaurant_address">{{ restaurant.address }}</p>
                        </div>
                    </a>
                {% endfor %}
            </div>
    {% elif recommended_restaurants_list is not none and not recommended_restaurants_list %} #{ 추천 식당 목록 X #
        <p class="info_message">죄송합니다. 입력하신 메뉴에 대한 식당을 찾을 수 없습니다.</p>
    {% endif %}
</div>

```

<밖에서 먹기>선택

- 추천 맛집 리스트 링크 연결

The screenshot shows the user interface for the 'out\_restaurant\_recommendation.html' page. At the top, there is a header with the text '냠 밖에서 먹자!' (Eat outside!). Below it is a form with a text input field containing '예: 파스타, 김치찌개, 치킨' (e.g.: pasta, kimchi, chicken) and a red '추천 받기' (Get recommendation) button. To the right of the form, there is a section titled '추천 식당 리스트' (Recommended Restaurant List) with a red button labeled 'Click Link'. Below this button, the text '['치킨' 맛집 검색 결과' (Search results for chicken restaurant) and '자세한 정보는 아래 링크를 클릭하세요.' (Detailed information can be found by clicking the link below) is displayed. At the bottom right of the page, there is a small button labeled '첫 페이지로 돌아가기' (Go back to the first page).

## out\_restaurant\_recommendation.html

```

[% extends 'base_layout.html' %]
[% block page_title %]외식 추천[% endblock %]
[% block content %]

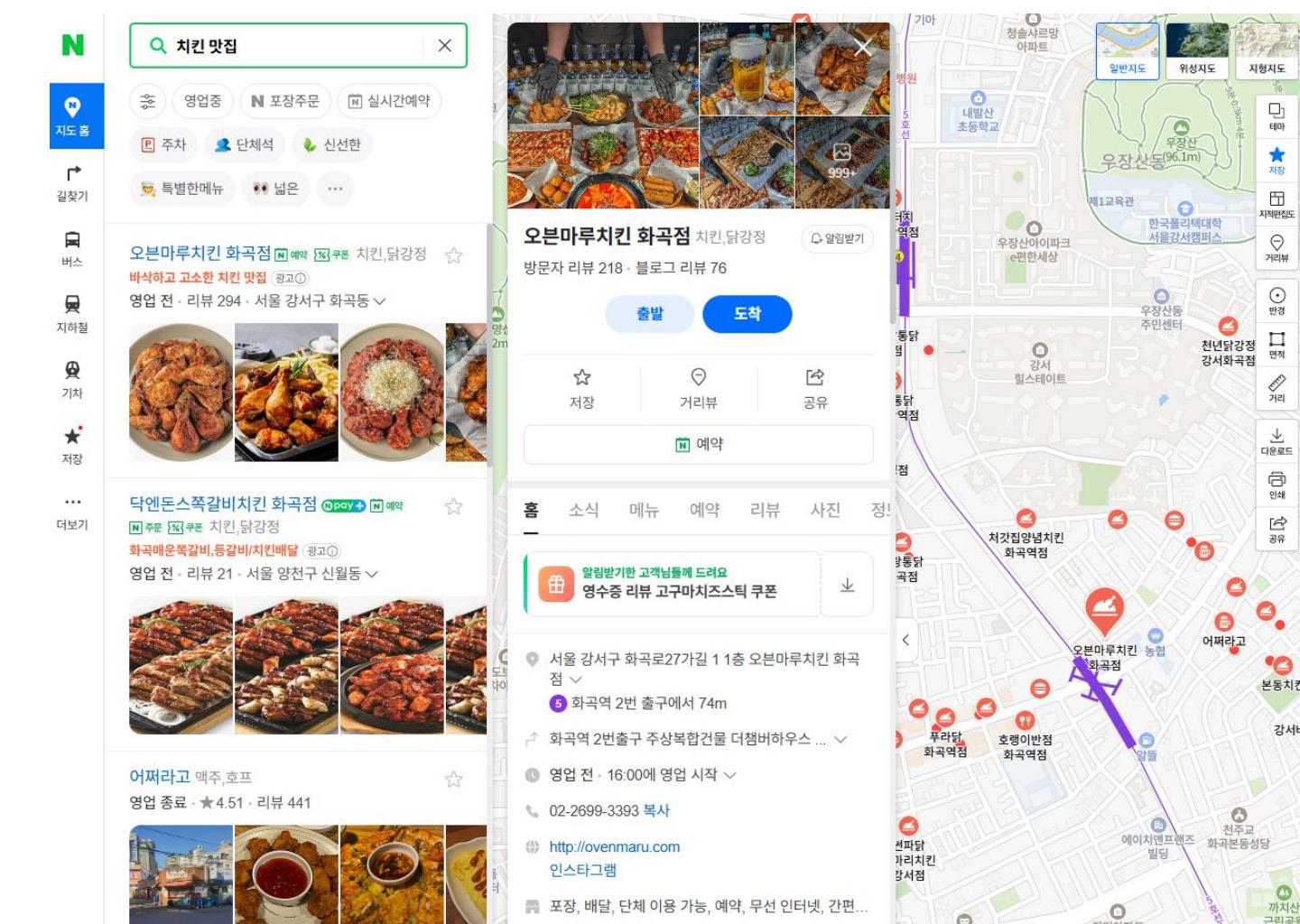
<div class="content_section">
  <h2>냠 놔에서 먹자!</h2>
  <p class="guide_text">어떤 메뉴를 드시고 싶으신가요? 메뉴를 입력하시면 주변 맛집을 추천해 드립니다.</p>

  <form method="POST" action="{{ url_for('out_restaurant_recommendation') }}">
    <label for="menu_input" class="sr_only">먹고 싶은 메뉴:</label>
    <input
      type="text"
      id="menu_input"
      name="menu"
      placeholder="예: 파스타, 김치찌개, 치킨"
      required
      aria-label="먹고 싶은 메뉴 입력"
    >
    <input type="submit" value="추천 받기" />
  </form>

  [% if error_message %]
    <p class="error_message">{{ error_message }}</p>
  [% endif %]

  [% if recommended_restaurants_list %] # 추천 식당 목록 0 #
    <h3>추천 식당 리스트</h3>
    <div class="restaurant_card_grid">
      <% for restaurant in recommended_restaurants_list %> # 추천 식당 목록 반복 표시 #
        <div class="card_element restaurant_card">
          <div class="restaurant_image_container">
            
          </div>
          <a href="{{ restaurant.link }}" target="_blank" rel="noopener noreferrer" class="restaurant_link_wrapper">
            <div class="restaurant_info_text">
              <h4>{{ restaurant.name }}</h4>
              <p class="restaurant_address">{{ restaurant.address }}</p>
            </div>
          </a>
        </div>
      <% endfor %>
    </div>
  [% elif recommended_restaurants_list is not none and not recommended_restaurants_list %] # 추천 식당 목록 X #
    <p class="info_message">죄송합니다. 입력하신 메뉴에 대한 식당을 찾을 수 없습니다.</p>
  [% endif %>
</div>

```



&lt;밖에서 먹기&gt;선택

- 추천 맛집 리스트 링크 연결

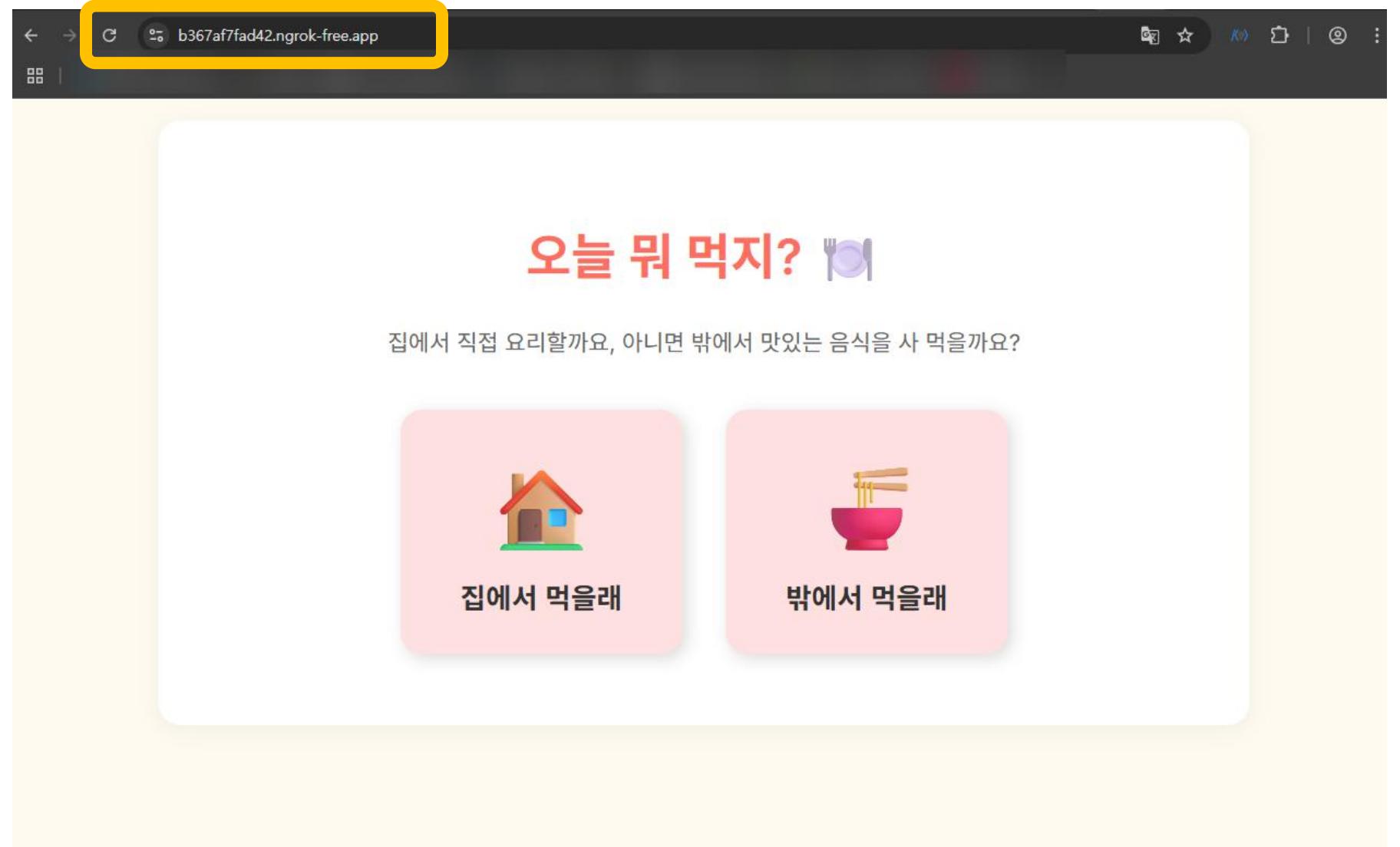
## 배포 - ngrok

```
관리자: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> ngrok http http://localhost:5000
```

```
관리자: Windows PowerShell
ngrok
Take our ngrok in production survey! https://forms.gle/aXiBFWzEA36DudFn6
Session Status      online
Account
Update             (Plan: Free)
Version            3.22.1
Region             Japan (jp)
Latency            60ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://b367af7fad42.ngrok-free.app -> http://localhost:5000
Connections        ttl  opn    rt1   rt5   p50   p90
                    0     0    0.00  0.00  0.00  0.00
```



## 3-2. 코드 설명

### ( Python code )

# Making\_db

웹 사이트 크롤링

```
for next_page in range(1, len(diff_page)):
    while True:
        try:
            try:
                page_list = driver.find_elements(By.CLASS_NAME, "list_navi_pg_cur")
                for page_num in range(len(page_list)+1):
                    food_lists = driver.find_elements(By.CSS_SELECTOR, 'span.link > a[href*="goRecipeView"]') # 현재 페이지에 있는 음식 데이터
                    food_lists = [elem for elem in food_lists if elem.text.strip() != ""]
                    for food_index in range(len(food_lists)):
```

```
food_number_list.append(i)
food_name_list.append(food_name)
types_cooking_list.append(food_type)
national_classification_list.append(food_country)
food_diff_list.append(food_diff)
food_cal_list.append(food_cal)
food_portion_list.append(food_portion)
food_main_ingredients_list.append(food_main_ingredients)
food_sub_ingredients_list.append(food_sub_ingredients)
food_spices_ingredients_list.append(food_spices_ingredients)
food_recipe_list.append(food_recipe)
```

중첩 for문을 통해 사이트를 반복적으로 이동하면서 페이지  
안의 필요한 정보를 크롤링  
이후 크롤링한 데이터들을 각 빈 리스트에 추가.

# Making\_db

웹사이트 크롤링

```

with open("./cook_recipe.csv", 'w', newline='', encoding="utf-8-sig") as f:
    wtr = csv.writer(f)

    # 1) header (컬럼명) 쓰기
    wtr.writerow(["food_number", "food_name", "recipe", "serving_number", "level"])

    # 2) for문을 이용한 여러 행 쓰기
    food_number_list = food_number_list
    food_name_list = food_name_list
    recipe_list = food_recipe_list
    serving_number_list = food_portion_list
    level_list = food_diff_list

    for i in range(len(food_name_list)):
        wtr.writerow([food_number_list[i], food_name_list[i], recipe_list[i], serving_number_list[i], level_list[i]])

```

	food_number	food_name	types_cooking	ingredient_main	ingredient_sub
1	1	견과류 영양솥밥	밥,죽,스프	백미 1과3/4컵, 은행 12개, 밤 8개, 대추 4개, 잣 1큰술	물 1과3/4컵
2	2	굴밥	밥,죽,스프	굴 300g, 쌀 200g, 보리쌀 80g	어묵(찐것) 50g, 표고버섯 3장, 당근 50g, 완두콩(통조림) 30g, 물 1과1/2컵
3	3	김치볶음밥	밥,죽,스프	김치(잘게썬것) 1/3포기, 밥 2공기	감자 1/4개, 양파 1/2개, 햄 60g, 식용유 약간, 계란 2개
4	4	꽃상추쌈	밥,죽,스프	상추 4장, 밥 1그릇	다진쇠고기 100g, 다진양파 1/2개

```

import pandas as pd
import sqlite3

df = pd.read_csv("cook_ingredient.csv")
conn = sqlite3.connect("cook_ingredient.db")
df.to_sql("cook_ingredient", conn, if_exists="replace", index=False)
conn.close()

```

이 리스트들을 csv라이브러리를 이용해  
Csv로 변환.

이후 sql을 이용해 이를 db로 변환.

## Recommend\_food.py Main backend

생성한 cook.db에서 필요한 재료data들을 정리해 리스트를 형성

```
data = []
for row in rows:
    entry = {
        'food_number': row['food_number'],
        'food_name': row['food_name'],
        'types_cooking': row['types_cooking'],
        'ingredient_main': row['ingredient_main'],
        'ingredient_sub': row['ingredient_sub'],
        'seasoning': row['seasoning'],
        'national_classification': row['national_classification'],
        'image_url': row['image_url']
    }
    # kcal 절수 변환
    try:
        kcal_raw = str(row['kcal'])
        entry['kcal'] = int(re.sub(r'^0-9', '', kcal_raw))
    except:
        entry['kcal'] = 0

    # 재료 리스트 생성 (정규화된 이름)
    ings = []
    ings += normalize_ingredient_name(entry['ingredient_main'])
    ings += normalize_ingredient_name(entry['ingredient_sub'])
    ings += normalize_ingredient_name(entry['seasoning'])
    entry['ingredients_list'] = ings

    # 원본 재료 리스트 생성 (문자열 포함된 원본 재료)
    original_ings = []
    if entry['ingredient_main']:
        original_ings.extend([p.strip() for p in entry['ingredient_main'].split(',') if p.strip()])
    if entry['ingredient_sub']:
        original_ings.extend([p.strip() for p in entry['ingredient_sub'].split(',') if p.strip()])
    if entry['seasoning']:
        original_ings.extend([p.strip() for p in entry['seasoning'].split(',') if p.strip()])
    entry['original_ingredients_list'] = original_ings

    data.append(entry)
return data
```

생성한 cook.db에서 필요한 음식레시피 data들을 정리해 리스트를 형성

```
# --- DB에서 cook_recipe 테이블 로드 ---
def load_recipes_from_db(db_path: str = 'cook.db') -> list:
    conn = sqlite3.connect(db_path)
    conn.row_factory = sqlite3.Row
    cur = conn.cursor()

    cur.execute('SELECT * FROM cook_recipe')
    rows = cur.fetchall()
    conn.close()

    data = []
    for row in rows:
        entry = {
            'food_number': row['food_number'],
            'food_name': row['food_name'],
            'recipe': row['recipe'],
            'serving_number': row['serving_number'],
            'level': row['level']
        }
        data.append(entry)
    return data
```

## Recommend\_food.py Main backend

앞서 생성한 리스트중 재료 리스트에서, 입력과 비교하여 레시피를 찾아 출력

```
def find_recipes_by_ingredients(user_ingredients_list):
    # 1) 사용자 입력 재료 정규화 및 소문자화
    normalized_users = []
    for ing in user_ingredients_list:
        parts = normalize_ingredient_name(ing)
        if parts:
            normalized_users.append(parts[0].lower())
    # 예: ['콩', '밥']
    results = []
    for info in COOK_INGREDIENT_DATA:
        # 레시피에 저장된 재료들 소문자화
        recipe_ings = [ri.lower() for ri in info['ingredients_list']]
        # 모든 사용자 재료가 어느 하나의 recipe_ing에 포함되는지 검사
        if all(any(user in recipe_ing for recipe_ing in recipe_ings)
              for user in normalized_users):
            # image_url이 비어 있으면 플레이스홀더 채워두기
            if not info.get('image_url'):
                info['image_url'] = (
                    'https://placehold.co/250x150/cccccc/333333?text=0|미지+없음')
            results.append(info)
    return results
```

또한 부족한 재료 리스트를 형성하기 위해 입력 받은 문자열과 비교

```
actual_missing_names = []
for ingredient_in_recipe in selected_recipe_ingredients:
    if ingredient_in_recipe.lower() not in user_have_lower:
        actual_missing_names.append(ingredient_in_recipe)

if actual_missing_names:
    # 부족한 재료 price_searching 을 수행합니다.
    missing_ingredients_data = price_searching(actual_missing_names)
else:
    info_message = "모든 재료가 충분합니다! 별도로 구매할 재료가 없어요."

else:
    # 세션 데이터가 없는 경우 (예: 새로고침 또는 직접 URL 접근)
    info_message = "재료 정보를 다시 불러올 수 없습니다. 레시피 페이지에서 다시 시도해주세요."
    return redirect(url_for('main_page')) # 메인 페이지로 리다이렉트

return render_template('missing_ingredients.html',
                      missing_ingredients=missing_ingredients_data,
                      recipe_food_name=recipe_food_name,
                      info_message=info_message)
```

## Recommend\_food.py Main backend

Youtube link를 실시간 크롤링

```
# --- 외부 API 시뮬레이션 함수 ---
def youtube_link(food_name: str) -> dict:
    options = wb.ChromeOptions()
    options.add_argument('headless')
    options.add_argument('window-size=1920x1080')
    options.add_argument("disable-gpu")

    driver = wb.Chrome(options=options)

    driver.get("https://www.youtube.com/")
    driver.maximize_window()
    # 검색
    search = driver.find_element(By.CSS_SELECTOR, ".ytSearchboxComponentSearchForm>input")
    search.click()
    search.send_keys(f"{food_name} 레시피")
    search.send_keys(Keys.ENTER)

    # 결과 로딩 대기
    elements = WebDriverWait(driver, 10).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, "#video-title"))
    )

    shorts_vids = []
    normal_vids = []
    for el in elements[:10]:
        href = el.get_attribute("href")
        title = el.get_attribute("title") or el.text

        # **여기** 꽃 ``title`` 과 ``link`` を 저장!
        vid_info = {
            "title": title,
            "link": href
        }

        if "/shorts/" in href and len(shorts_vids) < 2:
            shorts_vids.append(vid_info)
        elif "/watch?" in href and len(normal_vids) < 2:
            normal_vids.append(vid_info)

    driver.quit()
    return {"shorts": shorts_vids, "normal": normal_vids}
```

크롤링한 youtube link를 웹페이지에 반환

```
def to_embed_url(raw_url: str) -> str:
    # watch?v=abc123 → abc123
    m = re.search(r"(?P<v>=|\/shorts\/)([\w\-\_]+)", raw_url)
    if not m:
        return raw_url
    vid = m.group(1)
    return f"https://www.youtube.com/embed/{vid}"

def get_youtube_links_from_api(dish_name: str) -> list:
    vids = youtube_link(dish_name)
    out = []
    for section in ("shorts", "normal"):
        for info in vids[section]:
            out.append({
                "title": info["title"],
                # raw link → embed link
                "link": to_embed_url(info["link"])
            })
    return out
```

Find\_place\_to\_eat 함수로 지도 페이지 크롤링

```
def get_restaurants_from_api(dish_name):
    search_result_url = finding_place_to_eat(dish_name)
    return search_result_url
```

Map.py 안의 find\_place\_to\_eat 함수

```
def finding_place_to_eat(food_name):
    dining = "https://www.naver.com/"
    options = webdriver.ChromeOptions()
    prefs = {"profile.default_content_setting_values.geolocation": 1} # 위치 정보 권한을 허용으로 설정
    options.add_experimental_option("prefs", prefs)
    options.add_argument('headless') # Headless 모드 활성화
    options.add_argument('window-size=1920x1080') # 창 크기 설정 (headless에서도 중요)
    options.add_argument("disable-gpu")
    options.add_argument("no-sandbox") # 샌드박스 비활성화 (Docker 등 환경에서 필요할 수 있음)
    options.add_argument("disable-dev-shm-usage") # /dev/shm 사용 비활성화 (Docker 등 환경에서 필요할 수 있음)
    driver = webdriver.Chrome(options=options)
    driver.get(dining)
    time.sleep(2)
    driver.maximize_window()

    #WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.CSS_SELECTOR, "input_search")))
    search_dining = driver.find_element(By.CSS_SELECTOR, ".search_input")
    search_dining.click()
    search_dining.send_keys(f"{food_name} 맛집")
    search_dining.send_keys(Keys.ENTER)
    cur_url = driver.find_element(By.CSS_SELECTOR, ".bSoi3>a").get_attribute("href")
    return cur_url
```

부족한 재료의 정보를 search\_price함수를 이용해 크롤링

```
if actual_missing_names:
    # 부족한 재료 price_searching 을 수행.
    missing_ingredients_data = price_searching(actual_missing_names)
```

Price\_searcing.py안의 price\_searching 함수의 크롤링 파트

```
try:
    image_element = driver.find_element(By.CSS_SELECTOR, ".wrapImg>a")
    image_link = image_element.get_attribute('href')
except Exception as e:
    print(f'{ingred}의 이미지 링크를 찾을 수 없습니다: {e}')

try:
    best_product_element = driver.find_element(By.CSS_SELECTOR, ".price_discount")
    best_product_price = best_product_element.text
except Exception as e:
    print(f'{ingred}의 베스트 상품 가격을 찾을 수 없습니다: {e}')
```

웹페이지에 적용을 위한 return 형태 고정

```
seller_info = []
# --- 라인값만 변경하는 부분 시작 ---
if image_link != '정보 없음' and best_product_price != '정보 없음':
    seller_info.append({
        'seller_name': '오마시스',
        'image_url': product_img_url,
        'link': image_link,
        'price': best_product_price,
        'type': '베스트 상품'
    })
elif image_link != '정보 없음':
    seller_info.append({
        'seller_name': '오마시스',
        'image_url': product_img_url,
        'link': image_link,
        'price': '가격 정보 없음',
        'type': '베스트 상품'
    })
elif best_product_price != '정보 없음':
    seller_info.append({
        'seller_name': '오마시스',
        'image_url': 'https://placehold.co/100x100/cccccc/333333?text=사진없음',
        'link': '#',
        'price': best_product_price,
        'type': '베스트 상품'
    })
```

## 4. 프로젝트 결과

# 프로젝트 결과

## case1) 집에서 먹을래 선택

오늘 뭐 먹지? 🍲

집에서 직접 요리할까요, 아니면 밖에서 맛있는 음식을 사 먹을까요?



🛒 사용할 재료 개수는?

집에 있는 재료가 몇 가지인지 숫자로 입력해주세요. (2개에서 10개 사이)

3

다음 단계로

첫 페이지로 돌아가기

➊ 재료를 입력해주세요

집에 있는 재료들을 하나씩 입력해주세요. 입력하신 재료를 바탕으로 요리를 추천해 드립니다.

김치

계란

양파

요리 찾기

내가 입력한 재료:

김치, 계란, 양파

이 재료로 만들 수 있는 요리:



김치볶음밥  
밥, 죽, 스oup | 560 kcal



김치수제비  
면, 두부, 면종 | 410 kcal



순두부찌개  
찌개, 친교, 스oup | 133 kcal



김치게란피자  
피자 | 320 kcal



돼지고기김치롤튀김  
튀김, 커플릿 | 380 kcal



김치핫도그  
도시락, 간식 | 280 kcal

# 프로젝트 결과

## case1) 집에서 먹을래 선택

### 레시피

인분: 2인분 기준

난이도: 초보환영

#### 재료

✓ 김치(잘게썰것) 1/3포기

✓ 밥 2공기

✓ 감자 1/4개

✓ 양파 1/2개

✓ 햄 60g

✓ 식용유 약간

✓ 계란 2개

✓ 소금 약간

✓ 후춧가루 약간

#### 영상 보기

영상 섬네일이 안나오면 클릭을 해주세요!



모자란 재료 구입하기



#### 조리 순서

1. 김치는 1cm 크기로 썰고 햄, 양파, 감자도 1cm 정도의 크기로 잘게 썬다.
2. 프라이팬에 기름을 두르고 김치와 감자를 볶다가 햄과 양파를 넣고 볶아준다.
3. 양파가 어느 정도 익으면 밥을 넣고 펴듯이 볶아준다.
4. 소금과 후춧가루로 간을 맞추고 계란프라이를 얹어 낸다.



"김치볶음밥"을(를) 만들기 위한 모자란 재료!

이 요리를 만들려면 아래 재료들이 더 필요해요!

#### 밥



베스트 상품

5,250원

구입링크

#### 감자



베스트 상품

6,700원

구입링크

#### 햄



베스트 상품

5,600원

구입링크

#### 식용유



베스트 상품

32,000원

구입링크

#### 소금



베스트 상품

13,800원

구입링크

#### 후춧가루



베스트 상품

3,500원

구입링크

첫 페이지로 돌아가기

# 프로젝트 결과

## case1) 집에서 먹을래 선택

### 레시피

인분: 2인분 기준

난이도: 초보환영

#### 재료

✓ 김치(잘게썰것) 1/3포기

✓ 밥 2공기

✓ 감자 1/4개

✓ 양파 1/2개

✓ 햄 60g

✓ 식용유 약간

✓ 계란 2개

✓ 소금 약간

✓ 후춧가루 약간

#### 영상 보기

영상 섬네일이 안나오면 클릭을 해!



#### 오!감동 특가



3,990원

4,980원

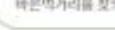
4,120원

3,990원

4,980원

4,120원

바른식거리를 찾으세요?



oasis.co.kr/product/detail/63253?categoryId=



로그인 회원가입 자주구매 주문/배송 조회



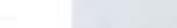
전체 카테고리 베스트 특가전 대용량특가 연기 뼈아끼리 맛집그대로 농축산물 할인지원 소설공연 만들마루 전통식품 청남장터 정기구독



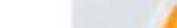
▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



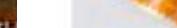
▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



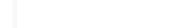
▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발



▶ 촉산 > 빅가공/恻발

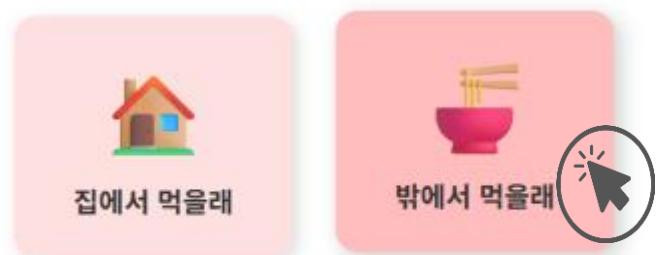


# 프로젝트 결과

## case2 ) 밖에서 먹을래 선택

오늘 뭐 먹지? 🍽️

집에서 직접 요리할까요, 아니면 밖에서 맛있는 음식을 사 먹을까요?



▶ 첫 페이지로 돌아가기

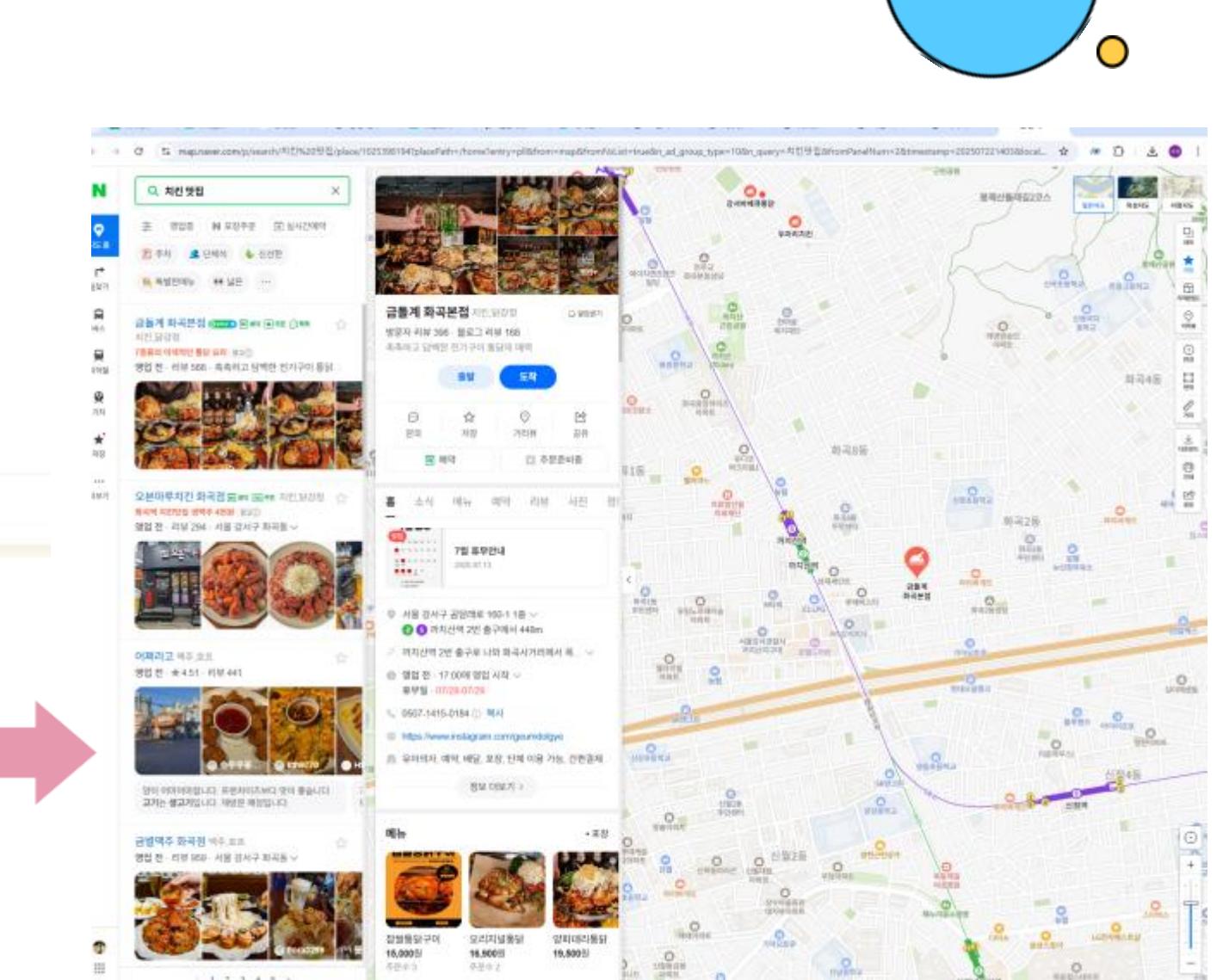
▶ 밖에서 먹자!

어떤 메뉴를 드시고 싶으신가요? 메뉴를 입력하시면 주변 맛집을 추천해 드립니다.

치킨

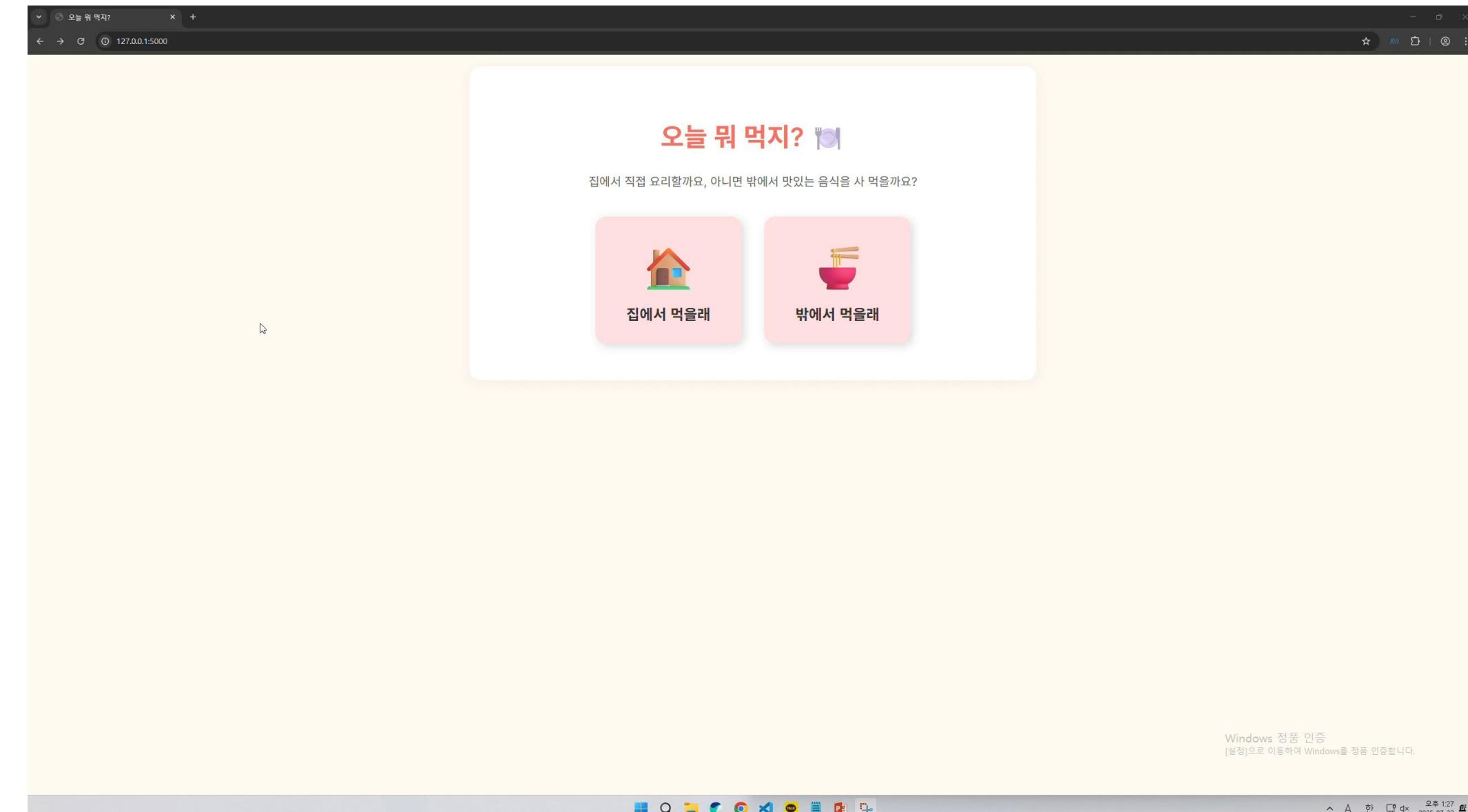


▶ 첫 페이지로 돌아가기



## 프로젝트 결과

# 구현영상



## 5. 트러블슈팅

# Trouble Shooting

## Timeout error

1. 문제 발생 상황 : 외식을 선택해서 지도 검색 결과를 url로 전달

2. 문제 사항 : 크롤링 과정중 CSS선택자가 로딩이 되지 않아 timeouterror 발생

### 3. 문제 원인

- 지도 사이트(naver map...)의 경우 우선적으로 지도를 loading
- 크롤링시 headless모드를 사용하는데, 이때 지도 ui가 로딩 되지 않음
- 따라서 우선순위에 따라 다른 html 정보가 로딩 되지 않기 때문에 선택자를 찾지 못하고, 해당 선택자가 로딩될때 까지 기다리는 함수인 EC.pre가 timeout을 발생시킴

### 4. 해결 과정

- 4-1.지도의 로딩이 빠른 다른 웹페이지 선택 = 동일한 에러 발생
- 4-2.headless 해제=웹사이트 로딩중 해당 사이트가 팝업되어 버리는 상황
- 4-3.Selenium의 disalbe-gpu해제 = 큰 변동 사항 없이 어김 없이 에러 발생

### 5. 해결 방안

직접 지도 사이트를 스크롤하는게 아닌, 네이버 검색 포탈에 food\_name 맛집을 검색, 이후 지도 사이트로 연결되는 선택자인 .bSoi3>a에 기여자인 href를 크롤링 이를 통해 지도 gui를 직접 로딩하지 않고 링크만 크롤링하는데 성공.

# Trouble Shooting

## Nametag error

```
[9]: food_lists = driver.find_elements(By.CLASS_NAME, "link a")
for elem in food_lists:
    print(elem.text)

견과류 영양솥밥
쫄밥
김치쫄밥
꽃상추쌈
녹두죽
단팥죽
달래간장과 모듬버섯밥
닭날개주먹밥
닭안심죽
대구계란죽
돌솥비빔밥
두부밥부침
무죽
무청밥
미니김밥
```

```
food_lists = driver.find_elements(By.CLASS_NAME, "link a")
for elem in food_lists:
    print(elem.text)

명란찜
모듬나물계란찜
모듬해산물채소찜
바지락콩나물찜
밥계란찜
버섯두부선
소꼬리찜
쇠고기가지찜
쇠고기청고추찜
알찜
양배추&호박찜
콘새우계란찜
티포크쌈
편육파채무침
갖은나물
```

```
food_lists = driver.find_elements(By.CSS_SELECTOR, 'span.link > a[href*="goRecipeView"]')
for elem in food_lists:
    print(elem.text)

명란찜
모듬나물계란찜
모듬해산물채소찜
바지락콩나물찜
밥계란찜
버섯두부선
소꼬리찜
쇠고기가지찜
쇠고기청고추찜
알찜
양배추&호박찜
콘새우계란찜
티포크쌈
편육파채무침
갖은나물
```

### 1. 문제가 발생한 상황

다중 for문 이전 각 for문에서는 정상적으로 작동되는 것을 확인  
이를 합치기 시작

### 2. 문제사항

1 페이지에 있는 음식 데이터를 크롤링 한 후 2페이지로 넘어가야 하나  
넘어가지 않고 다음 페이지 버튼을 클릭하여 11페이지로 이동. 이후 timeout이 발생

### 3. 문제 원인

food\_lists를 확인했을 때 안에 공백이 들어가 있어 문제가 발생함을 확인  
확인해보니 다음 페이지 버튼이 list안에 같이 들어가 있었음

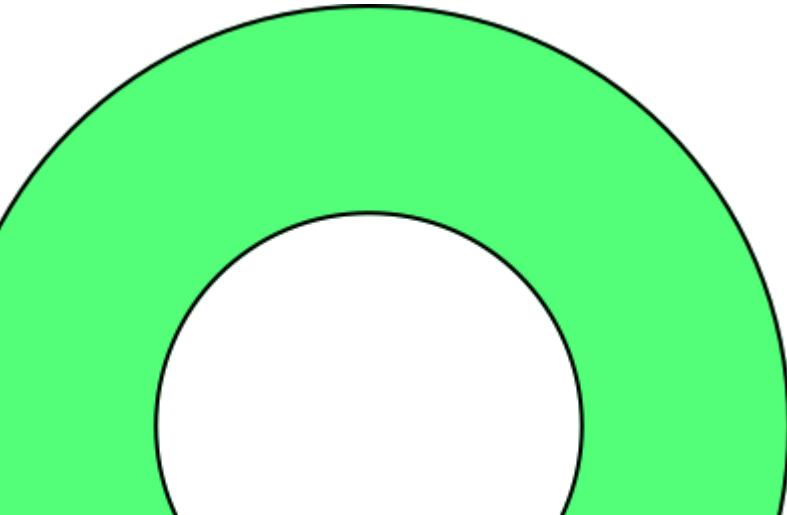
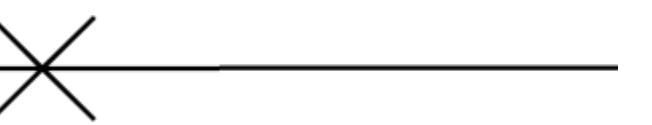
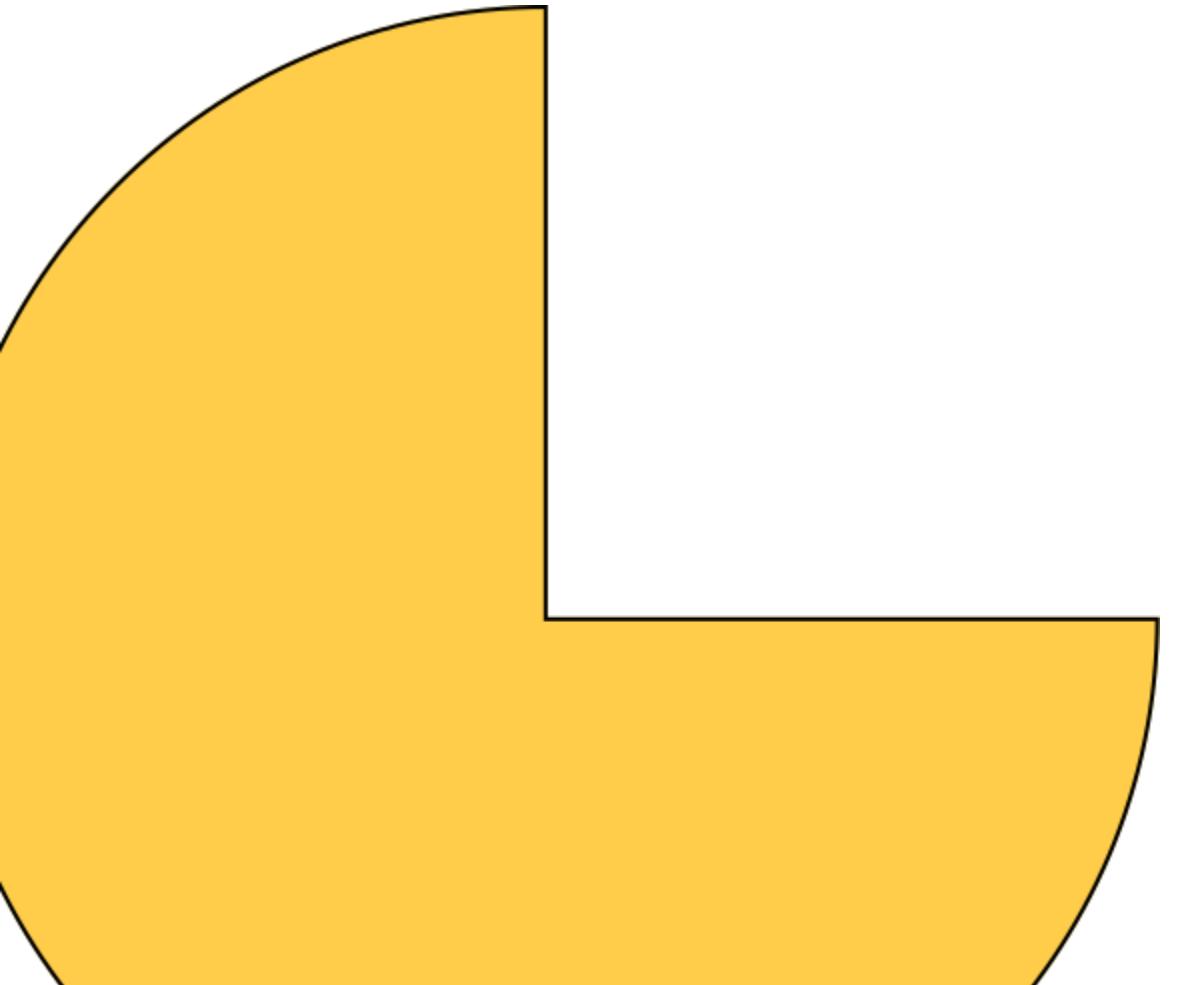
### 4. 해결과정

4-1) for문의 range를 len(page\_list)-1로 설정함

이전페이지 버튼이 존재할 경우 똑같이 list안에 들어가 있어 문제가 다시 발생

4-2) 태그를 link 클래스 안에 있는 a태그중에서 href에 goRecipeView가  
포함된 요소만 list에 저장하도록 설정

# 6. 소감



## 팀원소개 / 역할분배



김00

DB생성을 위한 데이터 크롤링,  
DB구축, html수정, ppt제작



김00

유튜브 크롤링 기초 완성,  
상품 정보 크롤링 완성,  
맛집 지도 완성, 발표



김은성

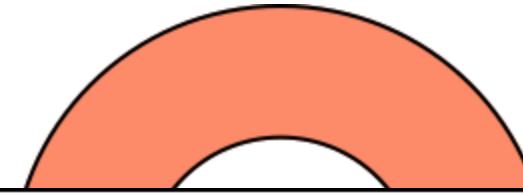
html 기초 구축 및 완성,  
유튜브 크롤링 기초 완성,  
ngrok 배포, 발표



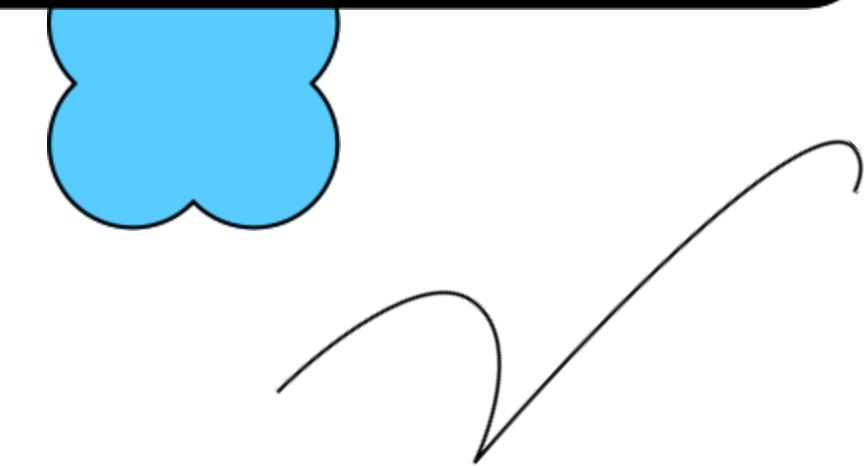
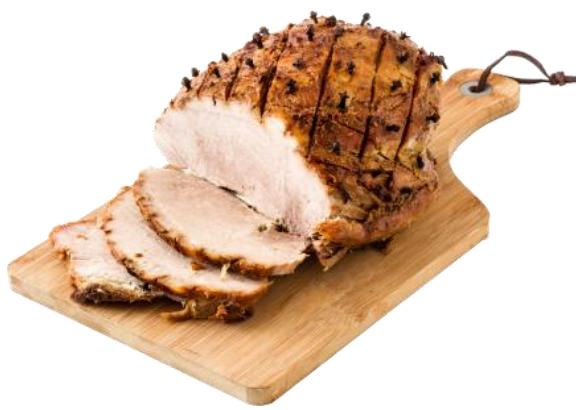
곽00

CSV생성 / CSV→DB생성 ,  
음식 리스트 img파일 작성,  
html 수정, ppt제작

소감



소감



김00

프로젝트를 진행하면서 python의 활용도를 늘릴 수 있었고, 팀원간의 협력하는 방법을 충분히 늘린것 같습니다. 다른 팀원분들 고생하셨습니다.

곽00

약 한 달간 배운 Python의 여러 기능을 종합적으로 복습하고, 이를 실제 프로젝트에 적용해보며 실전에 필요한 응용력을 키울 수 있었습니다. 특히 DB 데이터를 통해 웹페이지에 정보를 연동하고 시각화하는 과정을 직접 경험하면서, 데이터 흐름과 백엔드 개발의 구조를 보다 구체적으로 이해할 수 있었습니다. 또한 공통된 목표를 향해 팀원들과 협력하고, 함께 문제를 해결해 나가는 과정을 통해 소통과 협업의 중요성을 깨달았습니다.

김은성

미니 프로젝트를 통해 한 달간 배웠던 내용을 직접 적용해볼 수 있어서 유익했습니다. 프로젝트 진행 중 다양한 문제가 발생했지만 해결해나가는 과정을 통해 더 성장할 수 있어 의미 있는 시간이었습니다.

김00

배웠던 것들을 모두 써보지는 못했지만, 크롤링과 html만큼은 충분히 사용해본 것 같습니다. 프로젝트를 통해 python에 대해 많이 친숙해질 수 있었던 좋은 시간이었습니다

# Thank you

2조(밥조)

