

프로젝트 결과 보고서

프로젝트 명	냉장고를 부탁해
--------	----------

성명	소속	학년	학번	연락처	e-mail
신용준	컴퓨터공학과	2	21101195	01095176889	4rest528@naver.com
정세영	컴퓨터공학과	2	21101227	01093139489	jso8831@naver.com

Project 개요 및 예상결과물

- 개요: 사용자가 보유하고 있는 식재료를 기반으로 요리할 수 있는 음식을 추천해 주는 웹 사이트입니다.
- 목표: 기본 식재료(돼지고기, 고등어, 감자 등) 목록에서 현재 가지고 있는 식재료를 선택하면 추천하는 메뉴의 목록이 아래에 나열됩니다. 사용자가 메뉴를 선택하면 메뉴의 간단한 레시피, 추천 영상, 필요한 재료 등을 포함하는 음식 페이지로 이동하도록 구현하였습니다.

● 예상 디자인

(메인 페이지) (오늘의 메뉴) (최근 본 메뉴) (메뉴 추가하기)

냉장고를 부탁해

먹고 싶은 음식을 입력해주세요. (검색 창)

돼지고기	고등어	감자	당근	양파	...
------	-----	----	----	----	-----

돼지고기김치찜	고등어조림	감자볶음	김치볶음밥
애호박나물	비빔밥	계란찜	...

저작권 표시

(음식 페이지) 홈으로

음식 사진	음식 이름	
	재료	
레시피		요리 영상

- 기능:
 - 가지고 있는 재료를 기반으로 음식을 추천하고 레시피와 함께 조리 영상을 보여줍니다.
 - 사용자가 메뉴를 추가할 수 있는 기능을 제공합니다.
- 예상업무
 1. 식재료와 음식의 개수와 목록 선정
 2. 데이터 수집
 3. 웹 사이트 메인 페이지 제작 (HTML, CSS)
 4. 음식페이지 제작 (HTML, CSS)
 5. 페이지끼리 연결
 6. 추천 알고리즘 개발 (JAVASCRIPT)
 7. 옵션(부가기능) 추가

- 업무분담

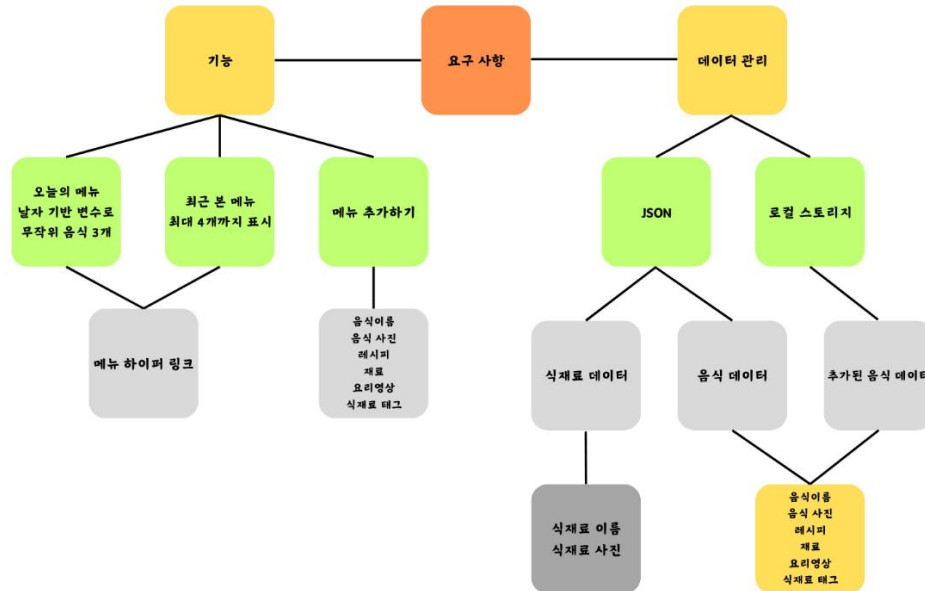
공동업무	1. 식재료와 음식의 개수와 목록 선정 2. 데이터 수집 3. 웹 사이트 메인 페이지 제작 (HTML, CSS) 7. 옵션(부가기능) 추가
신용준	6. 추천 알고리즘 개발 (JAVASCRIPT)
정세영	4. 음식페이지 제작 (HTML, CSS) 5. 페이지끼리 연결

- 일정

11.04 ~ 11.10	1. 식재료와 음식의 개수와 목록 선정
11.11 ~ 11.17	2. 데이터 수집
11.18 ~ 11.24	3. 웹 사이트 메인 페이지 제작 (HTML, CSS) 4. 음식페이지 제작 (HTML, CSS)
11.25 ~ 12.01	5. 페이지끼리 연결 6. 추천 알고리즘 개발 (JAVASCRIPT)
12.02 ~ 12.08	7. 옵션(부가기능) 추가
12.09 ~	마지막 점검

Project 설계

● 요구 다이어그램



- 사이트에서 필요한 요구사항은 크게 두가지로 나눌 수 있습니다. 사용자에게 제공하는 기능(서비스)과 기능을 데이터의 관리입니다. '냉장고를 부탁해'에서 제공하는 서비스는 메뉴정보 열람, 오늘의 메뉴 추천, 최근 본 메뉴 열람, 메뉴 추가하기까지 현재 총 네 가지가 있습니다. 메뉴 정보 열람은 별도의 html로 분리해서 구현하였고, 메인 페이지에서 나머지 기능을 구현하였습니다.
- 오늘의 메뉴는 날짜를 해싱하여 난수를 생성하여 메뉴를 추천합니다. 이를 통해 사용자는 매일 새로운 메뉴를 추천 받을 수 있습니다.

```

204 // 날짜 기반으로 고유한 메뉴 랜덤화
205 function getDailyMenu() {
206     const today = new Date();
207     const seed = today.toISOString().slice(0, 10); // YYYY-MM-DD 형식
208     const random = Math.abs(hashCode(seed)) % foods.length;
209
210     // 3개의 메뉴 추천 (랜덤 선택)
211     const dailyMenu = [];
212     for (let i = 0; i < 3; i++) {
213         dailyMenu.push(foods[(random + i) % foods.length]); // 연속적인 메뉴 추천
214     }
215     return dailyMenu;
216 }
217
218 // 문자열을 숫자로 변환하는 함수 (시드 값을 해시로 변환)
219 function hashCode(str) {
220     let hash = 0;
221     for (let i = 0; i < str.length; i++) {
222         const character = str.charCodeAt(i);
223         hash = (hash << 5) - hash + character;
224     }
225     return hash;
226 }
  
```

- 최근 본 메뉴에서는 로컬 스토리지를 활용하여 최대 n개의 메뉴를 표시합니다. n개에 메뉴가 이미 저장되어 있다면 열람한지 가장 오래된 메뉴를 리스트에서 제거합니다.

```

242 // "최근 본 메뉴" 버튼 클릭 이벤트
243 document.getElementById('recent-menu').addEventListener('click', () => {
244     showPopup("최근 본 메뉴", (container) => {
245         const recentMenu = JSON.parse(localStorage.getItem('recentMenu')) || [];
246         recentMenu.forEach(foodName => {
247             const food = foods.find(item => item.name === foodName);
248             if (food) addFoodCard(container, food);
249         });
250     });
251 });
252
253 // 페이지 로드 시 URL에서 음식 이름 처리
254 document.addEventListener('DOMContentLoaded', () => {
255     const urlParams = new URLSearchParams(window.location.search);
256     const foodName = urlParams.get('name');
257     if (foodName) {
258         saveToRecentMenu(foodName); // 메뉴 저장
259     }
260 });
261
262 // URL에서 음식 이름(name 파라미터) 추출 및 최근 본 메뉴에 저장
263 function handleFoodNameFromURL() {
264     const urlParams = new URLSearchParams(window.location.search);
265     const foodName = urlParams.get('name');
266     if (foodName) saveToRecentMenu(foodName);
267 }
268
269 // 페이지가 처음 로드될 때 URL에서 음식 이름 처리
270 document.addEventListener('DOMContentLoaded', handleFoodNameFromURL);
271
272 // 브라우저 히스토리에서 상태(popstate) 변경 시 URL에서 음식 이름 처리
273 window.addEventListener('popstate', handleFoodNameFromURL);

```

- 메뉴 추가하기에서는 form을 이용하여 사용자에게 메뉴이름, 레시피, 영상URL, 등을 입력 받습니다. 이를 food의 형태로 로컬 스토리지에 저장하여 새로운 메뉴를 기존 메뉴와 함께 표시합니다.

```

31 <!-- 메뉴 추가 팝업 -->
32 <div id="add-menu-popup" class="popup">
33     <div class="popup-content">
34         <h2>메뉴 추가하기</h2>
35         <form id="food-form" action="/submit" method="POST" enctype="multipart/form-data">
36             <!-- 메뉴 이름 -->
37             <label for="menu-name">메뉴 이름</label>
38             <input type="text" id="menu-name" name="menu_name" placeholder="메뉴 이름을 입력하세요" required>
39             <br>
40
41             <!-- 재료 -->
42             <label for="ingredients">재료 (선택으로 구분)</label>
43             <input type="text" id="ingredients" name="ingredients" placeholder="예: 고기, 양파, 소금" required>
44             <br>
45
46             <!-- 레시피 -->
47             <label for="recipe">레시피</label>
48             <textarea id="recipe" name="recipe" rows="5" placeholder="레시피를 입력하세요" required></textarea>
49             <br>
50
51             <!-- 요리 영상 주소 -->
52             <label for="video-url">요리 영상 URL</label>
53             <input type="url" id="video-url" name="video_url" placeholder="예: https://www.youtube.com/...">
54             <br>
55
56             <!-- 음식 이미지 업로드 -->
57             <label for="food-image">음식 이미지 경로</label>
58             <input type="text" id="food-image" name="food_image" placeholder="이미지 URL을 입력하세요">
59             <br>

```

```

60
61      <!-- 태그 -->
62      <fieldset>
63          <legend>태그를 선택하세요</legend>
64          <label><input type="checkbox" name="tags" value="감자"> 감자</label>
65          <label><input type="checkbox" name="tags" value="계란"> 계란</label>
66          <label><input type="checkbox" name="tags" value="김치"> 김치</label>
67          <label><input type="checkbox" name="tags" value="닭고기"> 닭고기</label>
68          <label><input type="checkbox" name="tags" value="해산물"> 해산물</label>
69          <label><input type="checkbox" name="tags" value="돼지고기"> 돼지고기</label>
70          <label><input type="checkbox" name="tags" value="두부"> 두부</label>
71          <label><input type="checkbox" name="tags" value="햄"> 햄</label>
72          <label><input type="checkbox" name="tags" value="소고기"> 소고기</label>
73      </fieldset>
74      <br>
75
76      <!-- 제출 버튼 -->
77      <button type="submit" id="add-menu-button">제출</button>
78  </form>
79  <div id="add-close-popup">x</div>
80
81 </div>

```

- 데이터 관리는 json과 로컬 스토리지를 사용합니다. json으로는 기존에 추가해 놓은 음식과 식재료의 데이터를 관리하고, 로컬 스토리지는 새로운 음식 데이터, 최근 본 메뉴, 등에 활용합니다.

[ingredients.json]

```

1  [
2      { "name": "감자", "image": "식재료_이미지/감자.png" },
3      { "name": "계란", "image": "식재료_이미지/계란.png" },
4      { "name": "김치", "image": "식재료_이미지/김치.png" },
5      { "name": "닭고기", "image": "식재료_이미지/닭고기.png" },
6      { "name": "해산물", "image": "식재료_이미지/해산물.png" },
7      { "name": "돼지고기", "image": "식재료_이미지/돼지고기.png" },
8      { "name": "두부", "image": "식재료_이미지/두부.png" },
9      { "name": "햄", "image": "식재료_이미지/햄.png" },
10     { "name": "소고기", "image": "식재료_이미지/소고기.png" }
11 ]

```

[foods.json]

```

1  [
2      {
3          "name": "감자전",
4          "ingredients": [
5              "감자"
6          ],
7          "recipeingredients": [
8              "감자 2-3개(400g)",
9              "꽃소금 약간",
10             "식용유 4큰술",
11             "물 2컵(360ml)",
12             "청양고추 적당량(고명용)",
13             "청양고추 1개(10g)",
14             "진간장 3큰술(30g)",
15             "식초 1큰술(8g)"
16         ],
17         "instructions": "<b>감자전</b><br>1. 감자는 껍질을 벗겨 적당한 크기로 자른다.<br>2. 믹서기에 자른 감자와 물을 넣고 곱게 간다.<br>3. 가는 체에 간 감자를 거른 후, 전분이 가라앉도록 약 10~15분 정도 둔다.<br>4. 물에서 분리된 전분이 가라앉으면 조심히 물을 따라내고 전분만 남긴다.<br>5. 전분만 남은 볼에 체에 걸러 둔 감자를 섞는다.<br>6. 반죽에 꽃소금을 넣어 골고루 섞는다.<br>7. 청양고추는 가능하게 송송 썬다.<br>8. 넓은 팬에 식용유를 넉넉히 두르고 중 불에서 팬을 달군 후 감자전 반죽을 한 국자씩 올린다.<br>9. 감자전 위에 송송 썬 청양고추를 올린다.<br>10. 감자전을 뒤집어 가며 노릇하게 부친다.<br>11. 잘 익힌 감자전을 양념장과 함께 낸다.<br><b>양념장</b><br>1. 청양고추는 두께 0.3cm 정도로 송송 썬다.<br>2. 그릇에 진간장, 청양고추, 식초를 섞어서 양념장을 만든다.",
18         "image": "음식_이미지/감자전.jpg",
19         "video": "https://www.youtube.com/embed/N82h3p15cMM?si=vb4UPNqEr7DBQX09"
20     },

```

[로컬 스토리지]

Key	Value
menus	[{"name": "사과파이", "ingredients": [], "recipeingredients": ["사과", "소금", "파이"]}]
recentMenu	["계란말이", "계란찜", "감자튀김", "감자조림"]

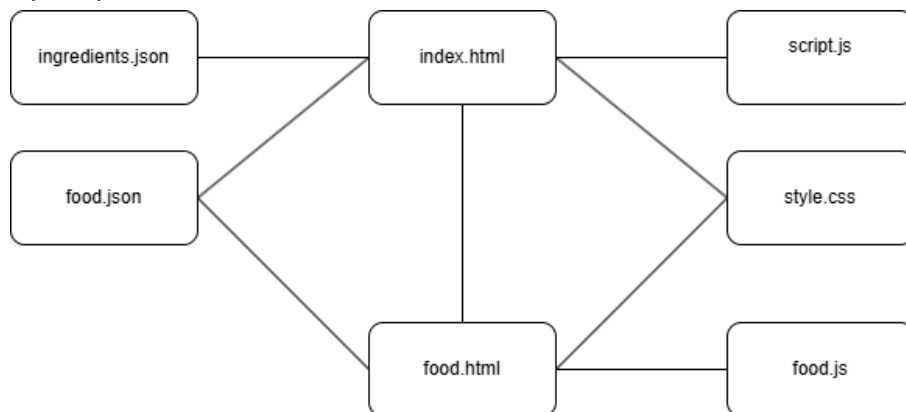
```

▼ [{"name": "사과파이", "ingredients": [], "recipeingredients": ["사과", "소금", "파이"], ...}]
▼ 0: {"name": "사과파이", "ingredients": [], "recipeingredients": ["사과", "소금", "파이"], ...}
  image: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAQ Show more (14.2 kB) Copy
  ingredients: []
  instructions: "사과파이 레시피 1<br>사과파이 레시피 2<br>사과파이 레시피 3"
  name: "사과파이"
  recipeingredients: ["사과", "소금", "파이"]
  video: "https://www.youtube.com/embed/21a1FiZKvY?si=JFQbdDQRdo6svn-n"

```

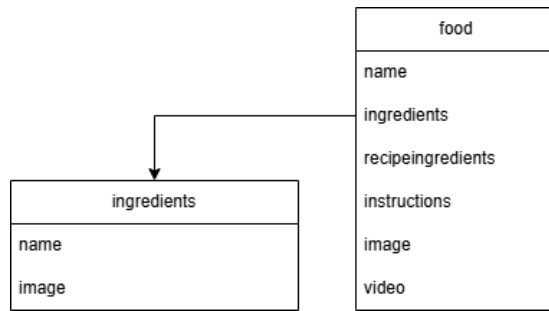
로컬 스토리지에 최근 본 메뉴와 사용자가 직접 추가한 메뉴들의 정보가 저장되어 있는 것을 확인할 수 있습니다.

● 파일 구조



- 프로젝트는 총 7개의 파일로 구성되어 있습니다. 메인 페이지인 index.html은 script.js와 style.css를 이용하여 동적 웹을 구성합니다. 음식의 상세 정보를 표시하는 food.html에서는 style표준을 맞추기 위하여 메인 페이지와 동일한 css파일을 사용하고, 페이지 내의 콘텐츠 표시와 동적 콘텐츠를 위하여 food.js를 활용합니다. 또한 음식 데이터와 식재료 데이터를 json파일로 분리하여 구성하였기 때문에 언제든지 json파일에 새로운 데이터만 추가한다면 새로운 음식과 식재료를 기반으로 웹이 동작할 수 있습니다.

- 클래스 구조



- 식재료인 ingredients는 식재료 이름과 이미지파일 정보를 가지고 있습니다. 음식을 나타내는 food는 음식 이름, 레시피, 재료, 이미지와 동영상파일 정보를 가지고 있고, ingredients항목에서 어떤 식재료를 포함하고 있는지 따로 저장하고 있습니다.

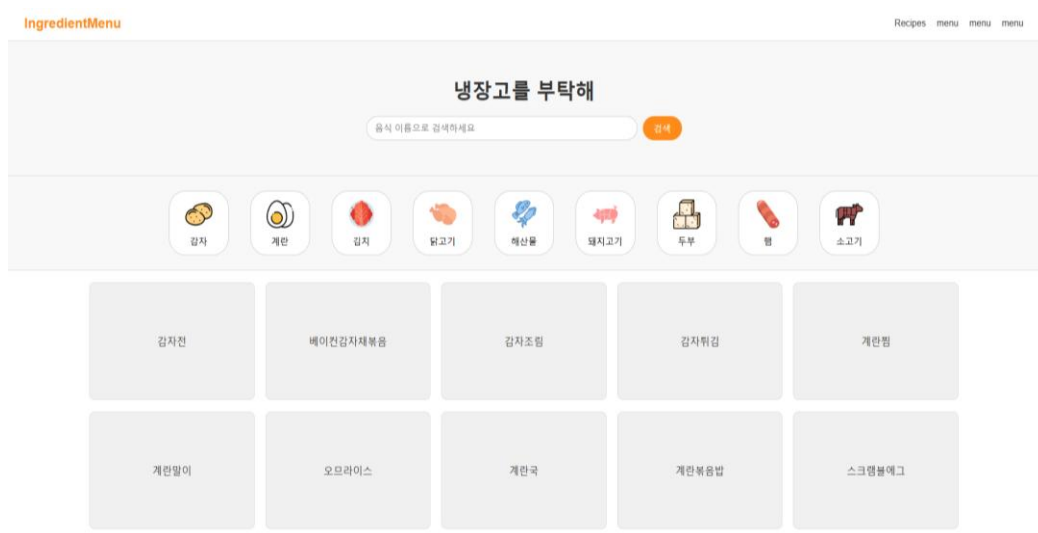
Project 진행내용

- 1주차: 재료 및 요리를 선정하였습니다.
 - 대부분의 냉장고에 남아 있는 주요 재료를 카테고리화하고, 이를 통해 요리 목록을 구성하였습니다

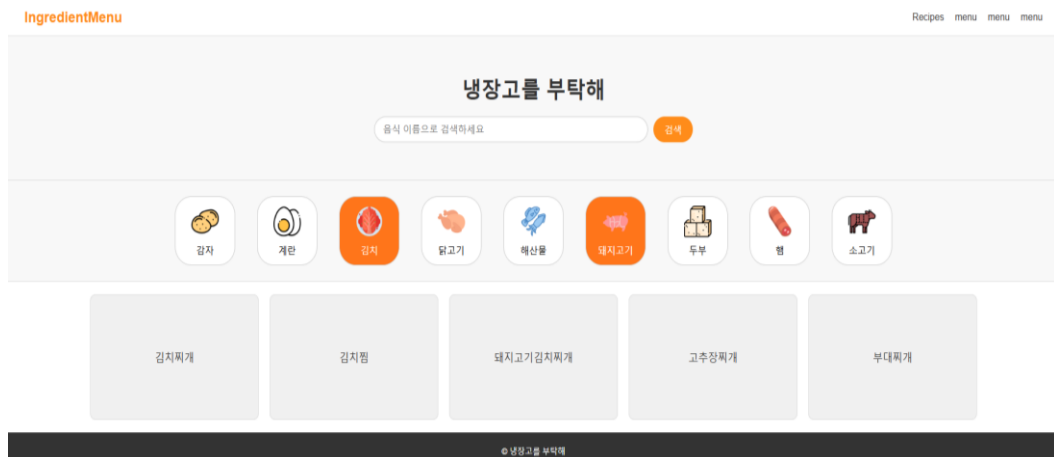
Recipe	간장	감자	계란	고사리	고추장	고춧가루	국간장
김치볶음밥							
두부김치	○					○	
부대찌개						○	○
김치찌개	○					○	
김치전							
제육볶음					○	○	
소세지 야채볶음							
계란말이			○				
오므라이스		○	○				
순두부찌개						○	
짜글이		○			○	○	○
베이컨감자채볶음		○					
마늘 닭강정	○						
닭볶음탕	○	○			○	○	
닭곰탕							
찜닭	○	○					
잡채	○						
육개장				○		○	○
감자전		○					
소불고기	○						
소고기 무국							○
갈비찜	○	○					
두부조림	○					○	
마파두부	○					○	
계란찜			○				

굴소스	굵은소금	김치	김치국물	깨	꿀	냉삼	느타리버섯	다시다
		○				○		
		○						
		○	○					○
		○	○					
○								
○								
○								
					○			
	○							
○				○			○	
○								
○								

- 2주차: 음식 재료와 메뉴를 보여줄 수 있는 메인 페이지를 제작하였습니다.
 - 필터링 기능과 검색 기능을 추가하였으며, 식재료 이미지를 표시하여 직관성을 높였습니다.



[식재료 필터링 기능]



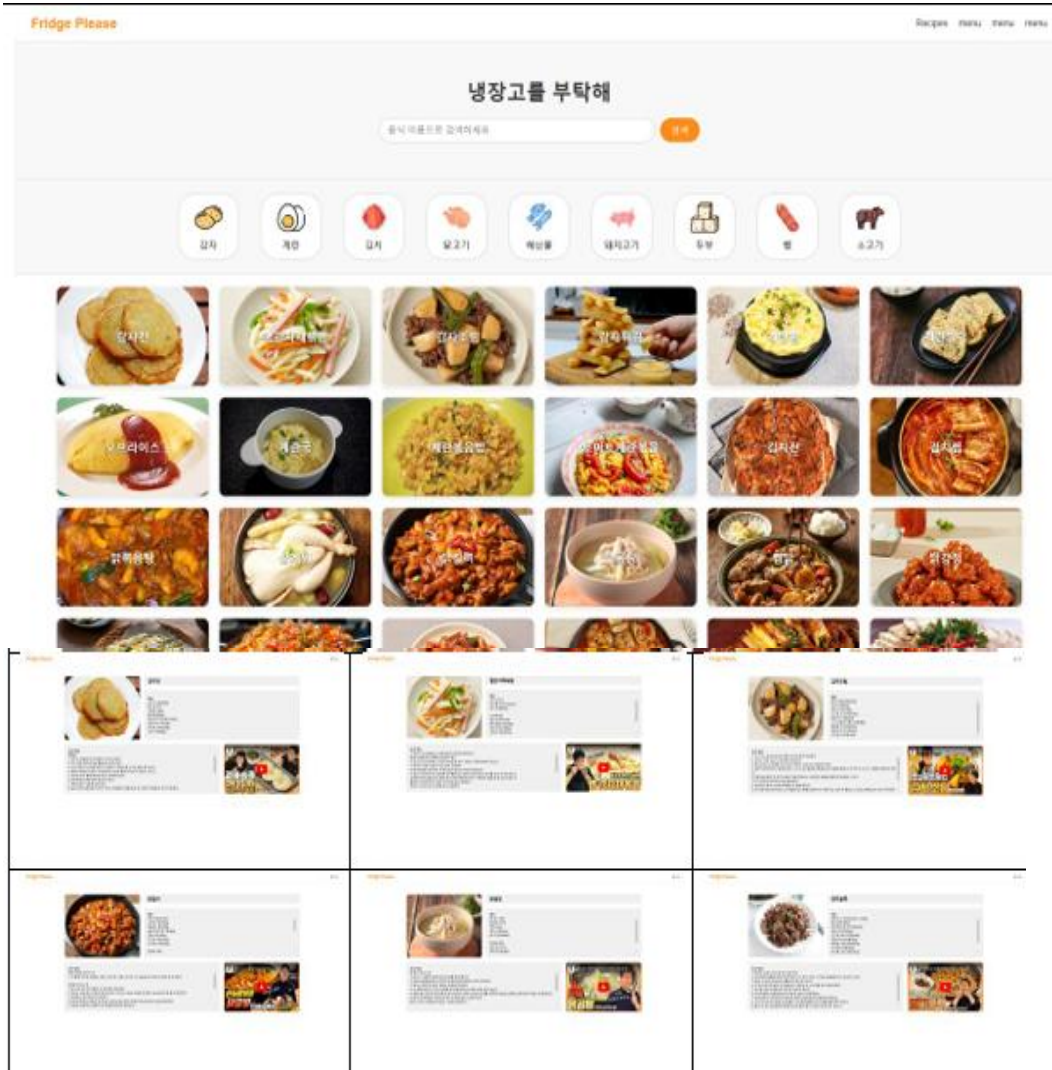
- 3주차: 음식 상세 페이지를 제작하고, 메인 페이지의 음식 카드를 클릭하면 해당 페이지로 이동하도록 구현하였습니다.



[음식 상세 페이지]

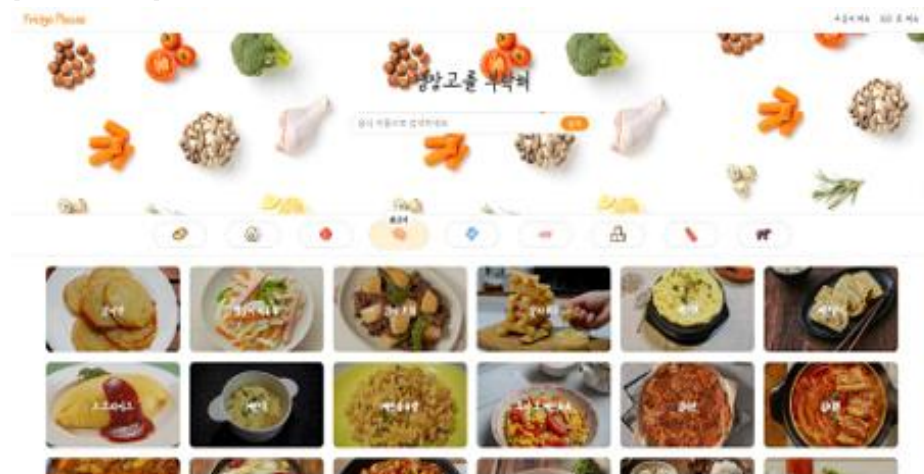


- 4주차: 메인 페이지 음식 카드에 음식 이미지와, 음식 이름이 이미지 위에 뜨도록 구현했습니다. 또한 3주차 때 만든 음식 상세페이지에 음식에 대한 정보가 나타나도록 데이터를 추가했습니다. 추가로 왼쪽 상단 사이트 로고를 누르면 메인 페이지로 이동하도록 만들었습니다.



- 5주차: 메인 페이지의 header영역에 신선해 보이는 식재료 배경 이미지를 넣어 메인 페이지에 깨끗하고 정돈된 느낌을 만들었습니다. 또한 네비게이션 바에 하루에 3개씩 랜덤으로 변하는 오늘의 메뉴, 최근에 봤던 음식들을 확인할 수 있는 최근 본 메뉴를 만들었습니다.

[헤더 영역]



[오늘의 메뉴]

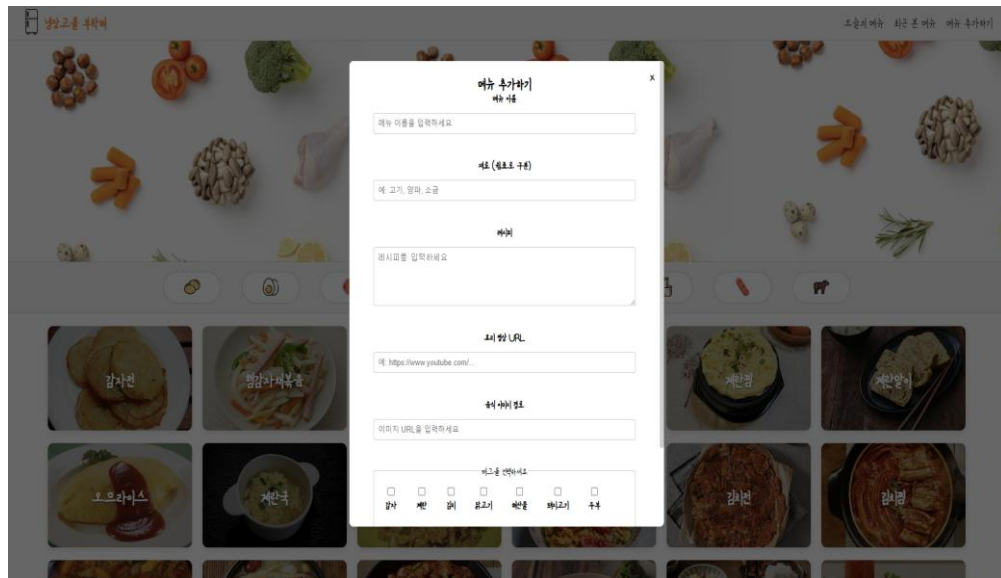


[최근 본 메뉴]



- 6주차: 네비게이션 바에 메뉴 추가하기를 만들어서, 개인이 등록하고 싶은 음식, 음식 이미지, 재료, 요리방법, 영상을 추가할 수 있게 만들었습니다. 등록된 음식은 현재 로컬 저장소에 저장되어 음식을 등록한 사람만 볼 수 있습니다.

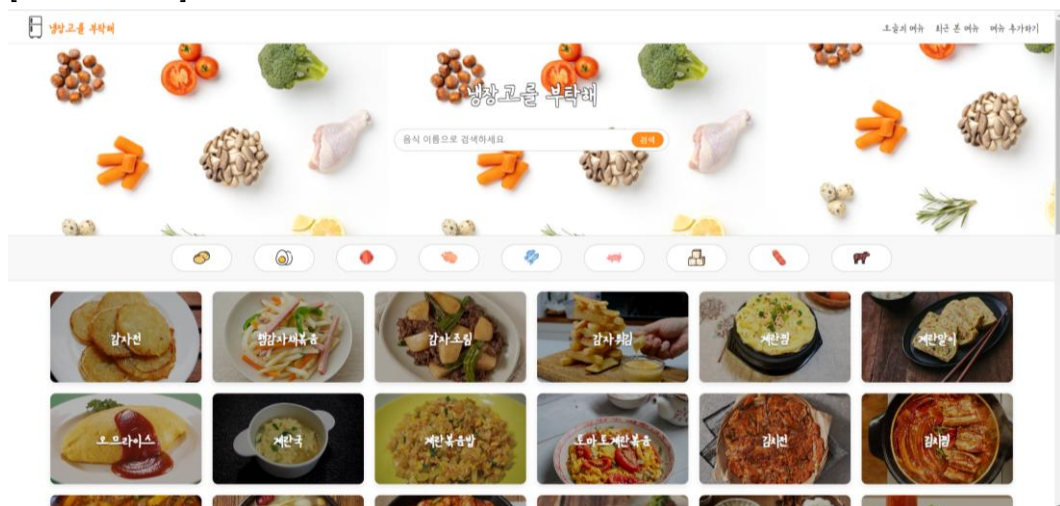
[메뉴 추가하기]



Project 수행 결과

- 최종 프로젝트 결과:

[메인 페이지]



- 메인 페이지의 네비게이션 탭에서는 오늘의 메뉴, 최근 본 메뉴, 메뉴 추가하기 기능을 사용할 수 있습니다. 또한 식재료의 이름을 기반으로 검색하는 기능과 식재료가 포함되었는지 버튼을 눌러 간단하게 필터링할 수 있는 기능을 사용할 수 있습니다. 검색 또는 필터링의 결과로 나열되어 있는 음식 카드를 클릭하면 해당 음식의 정보를 URL로 전달하여 상세 페이지에서 열람할 수 있습니다.

또한 웹의 모든 콘텐츠는 javascript를 사용하여 동적으로 제공하기 때문에 json파일만 수정하는 것으로 간단히 데이터를 추가할 수 있습니다.

[script.js]

```
1  // 전역 변수 선언
2  let foods = [];
3
4  // foods.json 데이터 불러오기
5  fetch('foods.json')
6    .then(response => response.json())
7    .then(data => {
8      foods = data;
9
10     // 로컬스토리지에서 사용자 추가 메뉴 가져오기
11     const userMenus = JSON.parse(localStorage.getItem('menus')) || [];
12     const allMenus = [...foods, ...userMenus];
13
14     // 초기 화면에 모든 음식 표시
15     displayFoods(allMenus);
16   })
17   .catch(error => console.error('Error loading foods:', error));
```



```


58 // ingredients.json 데이터 불러오기
59 fetch('ingredients.json')
60   .then(response => response.json())
61   .then(data => {
62     const ingredientsList = document.getElementById('ingredients-list');
63     data.forEach(ingredient => {
64       const btn = document.createElement('button');
65       btn.classList.add('ingredient-btn');
66       btn.setAttribute('data-title', ingredient.name); // 툴팁에 표시할 이름 추가
67
68       const img = document.createElement('img');
69       img.src = ingredient.image;
70       img.alt = ingredient.name;
71       img.classList.add('ingredient-img');
72
73       const name = document.createElement('span');
74       name.textContent = ingredient.name;
75
76       btn.appendChild(img);
77
78       btn.addEventListener('click', () => {
79         btn.classList.toggle('selected');
80         updateFoodList(); // 선택된 재료에 따라 음식 목록 업데이트
81       });
82
83       ingredientsList.appendChild(btn);
84     });
85   });
86
87 .catch(error => console.error('Error loading ingredients:', error));

```

[상세 페이지]

Fridge Please


홈으로



감자전

재료
 감자 2-3개(400g)
 밀가루 약간
 식용유 400ml
 물 200ml(350~400ml)
 청양고추 1개(고명용)
 청양고추 1개(30g)
 식염수 300ml(30g)
 식용유 100ml(100g)

요리 방법
 감자전
 1. 감자는 껍질을 벗겨 적당한 크기로 자른다.
 2. 밀가루에 자른 감자를 골고루 묻혀 준다.
 3. 자른 감자 2-3개를 기름 속, 식용유 가열된 후 약 10~15분 정도 튀긴다.
 4. 물에 푼 식염수 300ml를 가열된 후 약 10~15분 정도 끓인다.
 5. 식힌 후 남은 물에 300ml를 넣어 끓인다.
 6. 팬에 기름을 붓고 감자전을 튀긴다.
 7. 튀김이 노릇 노릇해질 때까지 튀긴다.



- 상세페이지에서는 음식의 사진, 이름, 재료, 요리법, 요리영상을 열람할 수 있습니다. 홈으로 또는 왼쪽 위의 로고를 클릭하면 메인 페이지로 복귀할 수 있습니다.
- 상세페이지의 콘텐츠는 메인 페이지에서 제공한 URL정보와 javascript를 사용하여 동적으로 제공하기 때문에 하나의 html페이지로 수많은 음식들의 정보를 소개할 수 있습니다.

[food.js]

```
1 // 상세 페이지 관련 코드
2 window.addEventListener('DOMContentLoaded', () => {
3     const urlParams = new URLSearchParams(window.location.search);
4     const foodName = urlParams.get('name');
5     console.log(foodName);
6     document.title=foodName+" 레시피";
7     if (!foodName) {
8         alert('음식 이름이 URL에 제공되지 않았습니다.');
```

14 .then(response => response.json())
15 .then(data => {
16 const localMenus = JSON.parse(localStorage.getItem('menus')) || [];
17 const allMenus = [...data, ...localMenus];
18 const food = allMenus.find(item => item.name === foodName);
19
20 if (!food) {
21 alert('해당 음식을 찾을 수 없습니다.');

```
9         return;
10     }
11
12     // foods.json + 로컬스토리지 데이터 합치기
13     fetch('foods.json')
14     .then(response => response.json())
15     .then(data => {
16         const localMenus = JSON.parse(localStorage.getItem('menus')) || [];
17         const allMenus = [...data, ...localMenus];
18         const food = allMenus.find(item => item.name === foodName);
19
20         if (!food) {
21             alert('해당 음식을 찾을 수 없습니다.');
```

24 displayFoodDetails(food);
25
26 })
27 .catch(error => {
28 console.error('foods.json 데이터를 불러오는 중 오류:', error);
29 alert('음식 데이터를 불러오는 중 문제가 발생했습니다.');

```
30     });
31
32 });
33
34 // 음식 데이터를 화면에 표시하는 함수
35 function displayFoodDetails(food) {
36     // 음식 이름
37     const foodNameElement = document.querySelector('.food-name h2');
38     if (foodNameElement) foodNameElement.textContent = food.name;
39
40     // 음식 사진
41     const foodImageElement = document.querySelector('.food-image img');
42     if (foodImageElement) {
43         foodImageElement.src = food.image || 'default-image.jpg';
44         foodImageElement.alt = food.name || '음식 이미지';
45     }
46
47     // 음식 재료
48     const ingredientsList = document.querySelector('.food-ingredients ul');
49     if (ingredientsList) {
50         ingredientsList.innerHTML = '';
51         if (food.recipeingredients && food.recipeingredients.length > 0) {
52             food.recipeingredients.forEach(ingredient => {
53                 const li = document.createElement('li');
54                 li.textContent = ingredient;
55                 ingredientsList.appendChild(li);
56             });
57         } else {
58             ingredientsList.innerHTML = '<li>재료 정보가 없습니다.</li>';
59         }
60     }
}
```


설계 요소

평가

● UI/UX 디자인

- 검색 기능 및 재료 필터링이 사용자 친화적이며 직관적인 인터페이스 제공.
- 반응형 웹 디자인으로 다양한 디바이스에서 원활한 사용 가능.
- 음식 카드 디자인은 간결하면서도 시각적으로 매력적임.

● 성능

- 장점:
 - ◆ JavaScript 기반 비동기 처리로 빠른 검색과 데이터 표시.
 - ◆ 로컬 스토리지를 활용한 데이터 저장으로 클라이언트에서 즉각적인 반응 제공.
- 단점:
 - ◆ 대규모 데이터를 처리하거나 동기화하는 데 한계 존재.
 - ◆ 음식 데이터가 적어 필터링 결과가 제한적일 수 있음.

● 안전성

- 장점:
 - ◆ 클라이언트-사이드 로컬 스토리지 사용으로 데이터 무결성 유지.
- 단점:
 - ◆ 보안에 민감한 데이터가 저장될 경우 노출 가능성.

● 보안성

- 장점:
 - ◆ 데이터 접근이 클라이언트-사이드로 제한되어 간단한 인증이 필요 없음.
- 단점:
 - ◆ XSS 공격 등에 취약할 수 있음. 예를 들어, 사용자 입력 값에 대한 검증 부족.

● 구현 단가

- 기본 HTML/CSS와 JavaScript를 활용하여 추가적인 라이선스 비용이 발생하지 않음.

● 기능확장

■ 음식 데이터 추가:

- ◆ 음식을 카테고리별(한식, 양식, 디저트 등)로 분류하고, 다양한 레시피를 포함한 방대한 데이터를 확보.
- ◆ 사용자 메뉴 추가 기능을 통해 수집된 데이터를 분석하여 음식 목록에 자동 반영.

■ 추천 시스템:

- ◆ 최근 본 메뉴와 검색 패턴을 기반으로 한 개인화된 추천 기능 추가.

■ 검색 기능 개선:

- ◆ 실시간 검색 자동완성 및 카테고리별 필터링 추가.
- ◆ 여러 재료를 조합한 고급 검색 옵션 제공.

● 성능 및 데이터 처리

- 정적 JSON 데이터를 클라우드 기반 데이터베이스(API)로 전환하여 음식 데이터 실시간 동기화
- Lazy Loading을 도입해 대규모 데이터 로딩 시 성능 최적화.

● 보안 강화

- 사용자 입력 데이터에 대한 XSS 방지 처리.
- HTTPS 프로토콜을 사용해 데이터 전송 보호.

● 확장 가능성

- 여러 사용자 계정을 지원하기 위해 인증 및 권한 관리 기능 추가.
- 다국어 지원을 통해 사용자 기반 확장.

종합 토의

이번 프로젝트는 사용자 중심의 음식 검색 및 관리 플랫폼으로, 제한된 데이터에서도 직관적이고 편리한 UI/UX를 통해 사용자 경험을 극대화했습니다. 특히, 검색 기능과 메뉴 추가 기능은 사용자에게 높은 활용성을 제공했습니다.

다만, 현재 음식 데이터의 개수가 적어 사용자에게 다양한 옵션을 제공하지 못하는 한계가 있습니다. 데이터 확장 및 추천 시스템 도입이 이루어진다면 사용자 참여를 높이고, 플랫폼의 실용성을 더욱 강화할 수 있습니다.

또한, 보안 강화를 통해 더 많은 사용자 데이터를 안정적으로 처리할 수 있도록 개선이 필요합니다. 이는 정적 로컬 스토리지 구조에서 클라우드 데이터베이스로 전환하는 과정에서 자연스럽게 이루어질 것입니다.

결론

본 프로젝트는 적은 비용과 제한된 자원으로 성공적으로 구현되었으며, 데이터 확장 및 성능 개선을 통해 음식 관리 플랫폼으로서 더 큰 잠재력을 발휘할 수 있습니다.

첨부

- 프로젝트에 관련된 자료는 모두 본 프로젝트의 레포지토리에서 확인할 수 있습니다.

https://github.com/Y0ngjun/Fridge_Please