

요리용 코딩 언어 개발



201924419 김도엽

202255661 박혜연

201924577 정종현

지도교수 조환규

목 차

1. 서론.....	1
1.1. 연구 배경 및 기존 문제점.....	1
1.2. 연구 목표.....	1
2. 연구 배경.....	1
2.1. 블록코딩.....	1
2.2. 관련된 도메인과 비교.....	2
3. 연구 내용.....	2
3.1. 블록 설계.....	2
3.1.1. 제어 블록.....	2
3.1.2. 동작 블록.....	3
3.1.3. 재료 블록.....	4
3.1.4. 병렬 연결 설계.....	6
3.2. Blockly 및 프론트엔드 구현.....	7
3.2.1. 웹서비스 프론트엔드 아키텍처.....	7
3.2.2. 서버와 클라이언트 간 데이터 동기화.....	8
3.2.3. Blockly 개요.....	8
3.2.4. Block-Dynamic-Connection.....	8
3.2.5. 블록 문법 설계와 연결 체계.....	10
3.3. 사용자 인증 및 권한 관리.....	22
3.3.1. 이메일 인증 구현.....	22
3.3.2. JWT.....	24

3.4.	데이터베이스	26
3.4.1.	MongoDB 소개 및 선택 이유.....	26
3.4.2.	DB 저장 구성.....	26
3.5.	배포 자동화	28
3.6.	프로젝트 기능 설명	31
3.6.1.	라우팅 구조.....	31
3.6.2.	페이지별 설명	31
3.6.3.	컴포넌트 및 주요 Asset.....	39
4.	연구 결과 분석 및 평가	40
4.1.	API 명세서.....	40
4.2.	구성원별 역할 분담	41
5.	결론 및 향후 연구 방향	41
5.1.	결론 및 기대효과.....	41
5.2.	향후 연구 방향.....	42
6.	참고 문헌	43

1. 서론

1.1. 연구 배경 및 기존 문제점

다양한 분야에서 데이터를 구조화해서 사용하고 있다. 그러나 요리 레시피는 여전히 자연어로 보관되며 컴퓨터가 이해하기 어려운 비정형 정보로 사용되고 있다. 동시에 코딩의 대중화로 많은 사람들이 프로그래밍을 접하고 있지만 텍스트 기반 프로그래밍 언어는 비개발자가 접근하기 어렵다는 점이 남아있다. 본 프로젝트는 요리를 프로그래밍할 수 있는가? 라는 질문에서 출발하여 블록형 코딩을 통해 요리 과정을 구조화하고 누구나 조작 가능한 형태로 전환하려는 시도를 담는다.

1.2. 연구 목표

이 프로젝트에서는 요리용 코딩언어를 Scratch 방식의 블록 인터페이스로 구현하여, 사용자가 시각적으로 각 요리 단계와 흐름을 쉽게 인식하고 구성할 수 있도록 돕는 도구를 개발하는 것을 목표로 한다. 각 블록은 크게 재료와 동작, 제어 블록으로 구성되며, 사용자는 이를 드래그 앤 드롭하여 요리 과정을 설계할 수 있다. 이러한 시스템을 통해 사용자는 요리법을 단순한 텍스트가 아닌 블록 기반의 구조화된 코드 형태로 직관적으로 표현하고, 조리 순서를 시각적으로 이해하고 조작할 수 있다. 또한 이 시스템은 레시피 자동화, 공유, 수정을 쉽게 만들어줄 뿐만 아니라 이후 디지털 조리 도우미나 스마트 주방 시스템과의 연계 가능성도 열어준다.

2. 연구 배경

2.1. 블록코딩

a. 스크래치

Scratch는 MIT Media Lab에서 2003년에 개발한 블록 기반의 시각적 프로그래밍 언어이다. 교육 분야에 활용되며, 학생들이 문제 해결 능력, 추상화, 절차적 사고력을 배울 수 있도록 돕는다. 블록으로 구성되어 있어 프로그래밍에 익숙하지 않은 사용자도 쉽게 접근할 수 있다. 다양한 플러그인을 제공하고 있어 많은 확장 가능성을 가지고 있다. 다양한 언어를 제공하여 여러 나라에서 사용할 수 있다.

2.2. 관련된 도메인과 비교

a. Fastfood

요리를 위한 규칙 기반(rule-based, 사람이 정한) 자연어 문장 자동 생성(Natural Language Generation, NLG) 프로그램이다. 요리 레시피를 담화 표현(discourse representation)으로 본다. FASTFOOD는 자연어 생성(Natural Language Generation) 중심 접근 연구이다. 하나의 시스템이라고 보기 보단 요리를 규칙 기반으로 정의하는 연구에 가깝다고 볼 수 있다.

b. 언어가 가져야 하는 조건

프로그래밍 언어는 소프트웨어 프로그램을 만들기 위해 사용되는 명령어와 문법의 집합이다. 주요 특징 중 하나는 제어 구조, 즉 프로그램의 흐름을 제어하는 구문으로, 반복문, 조건문 등을 포함한다.

3. 연구 내용

3.1. 블록 설계

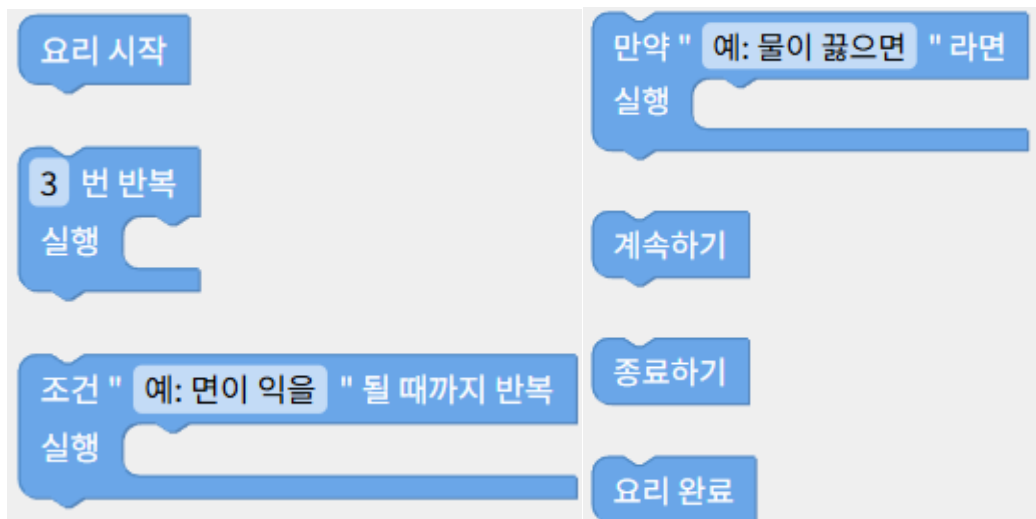
3.1.1. 제어 블록

조건문, 반복문을 한글로 바꾸어 나타내었다. 특히 반복문은 숫자로 반복하는 것과, 조건을 주어 특정 조건에서 종료하도록 하는 반복문으로 구별하였다.

계속하기와 종료하기를 넣어 반복을 사용할 때 더 유연하게 사용할 수 있도록 하였다. 다른 언어로 치면 `continue` 와 `break` 문이다.

요리 시작과 요리 완료도 표시하여 요리의 시작과 끝을 나타낼 수 있도록 하였다. 특히나 요리 완료와 종료하기가 구분되게 하여 요리가 완료되지 못할 때와 완료가 되는 것이 확실하게 구별되도록 만들었다.

제어 블록은 요리하는 것을 위해 설계했다기 보단 코딩언어에 맞추어 개발한 블록들이다. 2.2.3에 적힌, 언어가 가져야 하는 조건 중, 제어 구조를 만족하기 위해서 필요한 블록들이라고 판단하였다. 또한 이 블록들을 통해 사용자의 경험과 자유도를 높이고, 레시피의 완성도를 높일 수 있었다.



[그림 1] 제어 블록 첫번째

[그림 2] 제어 블록 두번째

3.1.2. 동작 블록

동작블록은 요리에서 하는 동작들에 관한 내용의 블록이다. 동작에는 재료 블록이 들어가게 되고, 동작 블록에는 들어갈 수 있는 재료 블록들을 제한하고 있다. 재료가 태그 별로 구분이 되며, 그 태그로 들어갈 수 있는지 없는지를 판단하게 된다. 태그는 영어로 설정하였다.

자르기 동작에는, liquid, powder, oil이 들어가면 안되며, 오직 solid 태그만 들어갈 수 있다.

볶기는 oil이 필수로 있어야 하고, 추가로 powder나 solid가 하나 이상 들어가야 한다. liquid는 들어갈 수 없는 속성이다.

섞기는, 태그는 상관없이 재료가 두개 이상 들어가야 한다.

넣기 동작은 태그와 재료 개수 둘 다 상관없다.

끓이기는 liquid가 필수로 들어가도록 하였다.

갈기는 solid가 필수로, powder가 있으면 안된다.

삶기는 liquid, solid가 필수로 들어가야 한다.

동작 블록은 statement 블록, 즉 조리 칸과 value 블록, 즉 조리값 칸으로 분리된다.

동작 블록을 이렇게 나눈 이유는, 동작도 동작 안에 들어가기로 하며, 이때 병렬적

으로 연결되거나 합쳐진 동작 블록을 동작 블록 안에 넣는 것도 가능하게 해야 하기 때문이다. 예를 들어, 자른 감자와 볶은 당근을 같이 삶고 싶을 때, 삶다에 두 가지 블록이 들어가야 한다. 이럴 때 사용할 수 있는 것이 value블럭과 statement블럭이다. 합치기 블록을 사용하여 value블럭들을 합치고, statement블럭을 사용하여 두 블록을 한 번에 넣어주는 것이다.

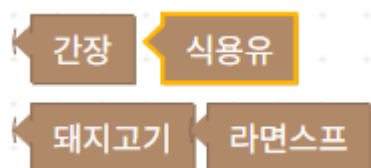
기다리기 블록도 존재하는데, 기다리기 블록은 이전 행동을 하고 얼마나 기다릴 것인지에 대한 정보를 입력하는 블록이다. 드롭다운으로 초, 분, 시를 선택할 수 있으며 시간을 숫자로 입력 가능하게 해두었다. 또한 기다리기 블록은 STATEMENT로만 필요하여 조리값에는 추가 블록이 들어가지 않는다.

자르기, 넣기, 갈기는 시간이 필요 없다고 판단하여 없이 설계했고, 나머지는 시간을 입력할 수 있도록 해두었다.



[그림 3] 동작 블록

3.1.3. 재료 블록



[그림 4] 재료 블록

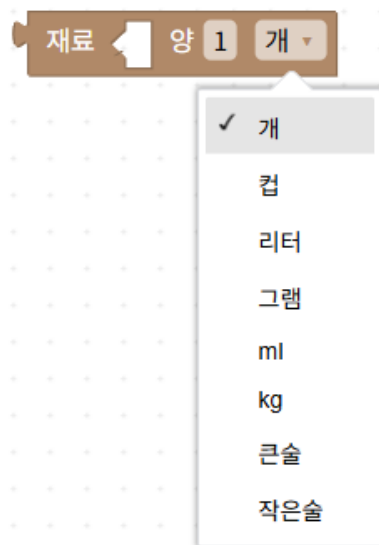
a. 검색 기능

검색 기능은, 재료가 많으므로, 사용자가 하나씩 다 찾아보지 않고 검색하여 재료를 찾을 수 있도록 하는 기능이다. 재료의 이름을 입력하면 해당하는 재료가 뜨게 된다.

b. Quantity 블록

가장 위에 있는 양에 관련한 블록은, 재료의 양을 상세히 적을 수 있도록 하는 블록이다. 레시피를 정량화할 수 있는 것이 우리의 목표인데, 이때 사용자가 마음대로 단위를 적게 되면, 너무 많은 단위가 나오게 되어 정량화를 할 수 없게 된다. 그러므로 세상에 있는 모든 단위를 넣을 수는 없었지만, 필요하다고 생각 되는 단위들을 모두 드롭다운으로 설계하였다.

이 블록을 사용하지 않으면, 다른 블록들에 재료는 연결할 수 없게 설계하였다. 얼마나 들어가는지의 대략적인 양을 입력 받아야 한다고 생각하여, 이 프로젝트에서는 블록의 연결 단자를 다르게 설계하였다. 기본 재료 블록의 연결단자는 세 모, 나머지는 퍼즐모양으로 하여, 기본 재료는 무조건 quantity블럭에 들어가야만 하도록 설계하였다. 오류 메시지를 다른 것처럼 띄울 수도 있었지만, 이것만큼은 사용자에게 쉽고 정확하게 전달하고 싶어 선택한 방법이다. 가장 직관적인 방법 이라고 생각하여 사용하게 되었다. 이로써 재료 블록과 모든 블록은 눈으로 확실 하게 구분이 된다.



[그림 5] quantity 블록

3.1.4. 병렬 연결 설계

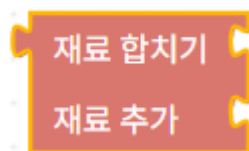
요리 과정을 블록으로 표현할 때, 하나의 동작에 여러 재료가 동시에 포함되는 경우가 많다. 예를 들어, “볶다”라는 동작에는 단일 재료만 들어가는 것이 아니라 식용유와 당근을 함께 사용하는 경우가 일반적이다. 그러나 기본적인 블록 구조에서는 “볶다” 블록에 재료를 하나씩만 넣을 수 있어, 식용유와 당근을 함께 표현하려면 “볶다” 블록을 두 번 배치해야 한다. 이렇게 되면 실제 요리 과정과 달리, 두 번 볶는 것처럼 보이게 된다.

이 프로젝트에서는 이러한 문제를 해결하기 위해, 한 블록 안에 여러 재료 블록을 병렬 연결할 수 있도록 하였다. 즉, 사용자는 “볶다” 블록에 식용유, 당근, 양파와 같은 여러 재료를 한 번에 추가할 수 있으며, 이는 실제 조리 과정과 더욱 유사한 표현을 가능하게 한다.

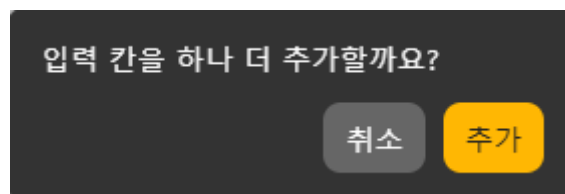
재료 뿐만 아니라, 요리 동작 또한 병렬적으로 표현할 수 있도록 하였다. 예를 들어, “끓이다”라는 동작 안에는 단순히 재료만 들어가는 것이 아니라, “자르다”, “볶다”와 같은 하위 동작을 동시에 포함할 수도 있다. 이를 가능하게 하기 위해 하나의 블록을 확장 가능한 형태로 설계하여, 사용자가 원하는 만큼의 동작 블록을 계속 추가할 수 있도록 구현하였다.

이는 재료의 전처리 과정을 위한 것으로, 주로 감자를 채 썰다와 같은 재료 준비 동작에 해당된다.

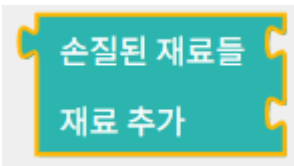
이러한 병렬 연결 설계를 통해, 블록 구조가 실제 요리 과정의 동시성과 복합성을 효과적으로 반영할 수 있게 되었으며, 결과적으로 더 직관적이고 실질적인 레시피 표현이 가능해졌다.



[그림 6] 재료 병렬 연결 블록



[그림 7] 재료 병렬 연결 입력 추가 팝업

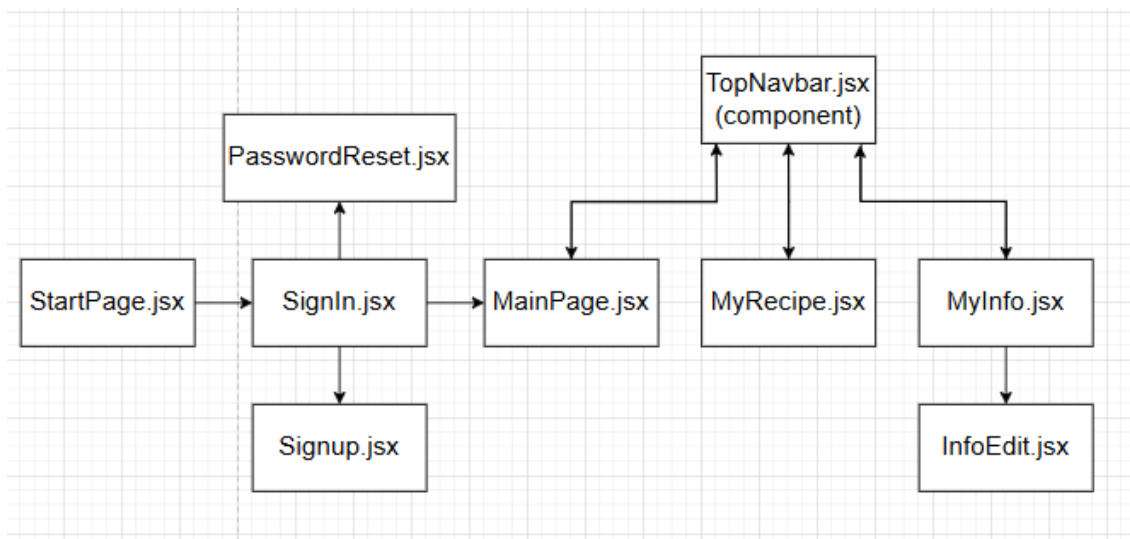


[그림 8] 동작 병렬 연결 블록

3.2. Blockly 및 프론트엔드 구현

3.2.1. 웹서비스 프론트엔드 아키텍처

BlockChef 프론트엔드는 React 기반 단일 페이지 애플리케이션(SPA)으로 구성하였으며, 전역 상태 관리 도구 없이 로컬 상태와 얇은 props 전달만으로 상태를 관리하였다. 컴포넌트 계층은 TopNavbar → MainPage(MyRecipe/Signin/MyInfo) → (페이지 내부 구성요소) 정도의 깊이로 단순하여, 전역 스토어를 사용하지 않고도 가독성과 유지보수성을 확보할 수 있다.



[그림 9] 프론트 아키텍처 구성도

핵심 구성요소

- 주요 페이지: MainPage(레시피 만들기), MyRecipe(나의 레시피), SignIn, Signup, MyInfo, InfoEdit

-
- b. 주요 컴포넌트: BlocklyArea(Blockly 작업영역), TopNavbar(상단 바), TagInput, LoginButton 등
 - c. 상호작용: MainPage가 BlocklyArea의 ref API(undo/redo/getXml/clear)를 호출하여 워크스페이스를 제어
 - d. 테마: 메인 동작/준비 동작/흐름 색을 명확히 구분

3.2.2. 서버와 클라이언트 간 데이터 동기화

BlockChef의 동기화는 단건 저장/불러오기와 목록 조회/삭제 중심으로 단순하다. 저장 성공 시 세션 스토리지에 dirty=0을 기록해 UI 상태와 동기화하며, 나의 레시피(MyRecipe)에서는 삭제 직후 목록과 태그 집합을 즉시 재계산하여 일관성을 유지한다.

3.2.3. Blockly 개요

Blockly는 맞춤설정 가능한 블록 기반 코드 편집기를 앱에 추가할 수 있는 웹 라이브러리이다. 이 편집기는 블록과 같은 퍼즐 조각을 사용하여 변수, 논리 표현식, 루프와 같은 코드 개념을 나타낸다. 이를 통해 사용자는 문법이나 명령줄의 위협에 대해 걱정하지 않고 프로그래밍할 수 있다.

퍼즐 연결과 입력란을 정의하면 Blockly에서 복잡한 렌더링, 드래그, 연결을 처리한다.

각 블록에 대해 생성되는 문자열 (일반적으로 코드)을 정의하면 Blockly에서 블록의 전체 문자열 연결을 처리한다. 이 결과를 어떻게 활용할지는 사용자에게 달려 있다. 미로 풀기, 캐릭터 애니메이션, 데이터 분석 등 무엇이든 할 수 있다.

Blockly를 사용하면 블록의 작동 방식에 대한 세부정보를 걱정하지 않고 도메인에 블록을 적용하는 데 집중할 수 있다.

3.2.4. Block-Dynamic-Connection

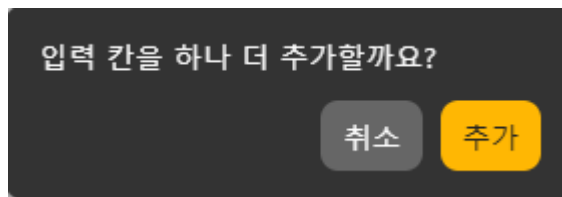
블록 동적 연결은 블록은 원래 모양이 정해지면 정해진 모양대로만 사용이 되어야 하는데 이렇게 설계하면 2칸이 연결이 필요한 경우 3칸, 4칸 등등 매번 필요한 모양을 새로 설계하고 만들어 내야 하다 보니 유지보수성이 떨어짐.

이에 블록을 연결시 필요한만큼 블록의 입력칸 개수를 갱신하여 추가할지 말지를 결정하여 블록의 모형을 동적으로 변형시켜 블록간 연결 개수를 쉽게 변경할 수 있도록 설계된 기법이다.

a. 재료 합치기 블록

재료가 단일로 쓰일수 없는 동작블록(ex: 섞기 블록은 최소 두가지 재료를 섞어야 함)들이 존재하기에 재료들을 여러 개 사용 할 수 있게끔 구현하였으며 재료 합치기 블록의 입력칸이 다차면 추가로 입력칸을 늘리겠냐는 팝업창이 뜨며 이는 block-dynamic-connection을 통해 구현이 된다.

구현 결과는 아래와 같다.



[그림 10] 입력 칸이 다 찼을 시 나타나는 팝업



[그림 11] 확인을 눌렀을 때 블록 모형

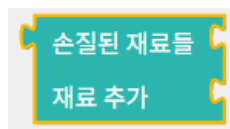


[그림 12] 취소를 눌렀을 때 블록 모형

b. 손질된 재료 블록

재료 카테고리에 있는 재료 합치기 블록과 마찬가지로 block-dynamic-connection으로 구현이 되었다. 재료 합치기 블록과 다른 점은 재료 계량 블록이 바로 손질된 재료블록에 직접적으로 결합할 수 없으며 해당 블록엔 재료 손질을 위한 동작블록(value)가 직접적으로 연결이 가능하다.

구현 결과는 아래와 같다.



[그림 13] 손질된 재료들 블록 모형



[그림 14] 손질된 재료들 블록 모형에 value 동작 블록이 연결된 형태

3.2.5. 블록 문법 설계와 연결 체계

1. 타입 계층과 카테고리

a. 타입 계층

ING_NAME: 원시 재료 이름(예: “감자”, “간장”).

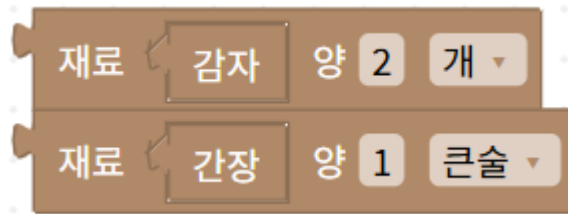
어디에도 직접 쓰이지 않고, 반드시 재료 계량 블록안의 NAME 슬롯에만 들어간다.



[그림 15] 재료 블록: 왼쪽에서부터 고체, 액체, 기름, 가루 형태의 재료

ING: 계량 완료된 재료(예: “감자 2 개”, “간장 1 큰술”).

모든 동작의 입력(ITEM)은 ING를 받는다.



[그림 16] 재료 계량 블록 안에 입력 칸으로 재료 블록이 들어간 형태

동작/재료(ACTION/ING)결과값: 재료 준비동작 카테고리에 있는 블록의 출력은 재료인가? 동작인가?를 체크하게 설정되어, 다음 동작의 입력 재료계량블록(ING)으로도, 또 상위의 준비된 재료 묶음으로도(ACTION) 흐를 수 있다.

b. 카테고리(블록 더미)와 연결 방향

메인 요리동작: 상/하 방향으로 연결. 주된 요리에 대한 동작에 대한 수행을 함.

입력칸: 입력은 값 슬롯 좌측으로 ING를 받지만, 블록 자체는 상/하로 연결되어 순서를 만든다.

재료 준비동작: 좌/우 방향으로 연결. 메인 요리동작들의 Value(값)와 손질된 블록이 여기에 속한다.

좌/우 방향블록(value_block)은 출력이 재료계량블록, 동작블록("ING", "ACTION")이라 “다음 준비 동작의 입력” 또는 “메인 요리동작의 재료”로 재사용 가능.

재료 합치기(combine_block): 값 블록. 입력 슬롯을 동적으로 늘려 여러 재료계량블록(ING)를 한 묶음으로 만든다.

최외각(가장 바깥의 블록) 입력(ITEM0, ITEM1, ...)에는 오직 재료계량블록만 허용.

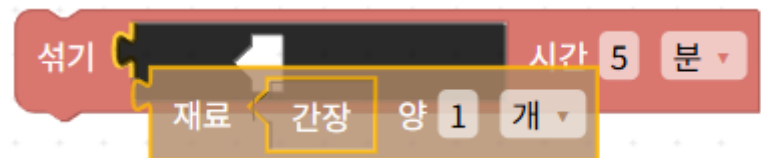
준비된 재료(action_combine_block): 값 블록. 여러 재료 준비동작(Value)을 한 묶음으로 만든다.

최외각(가장 바깥의 블록) 입력에는 동작 값/결과만 허용(즉 재료블록 및 재료 계량블록 금지).

2. 연결 전/후 판정과 사용자 피드백

a. 연결 미리보기

드래그 중 검은색 삽입 마커(insertion marker)가 보이면 “형상적으로 끼힐 수 있다”는 의미이다.



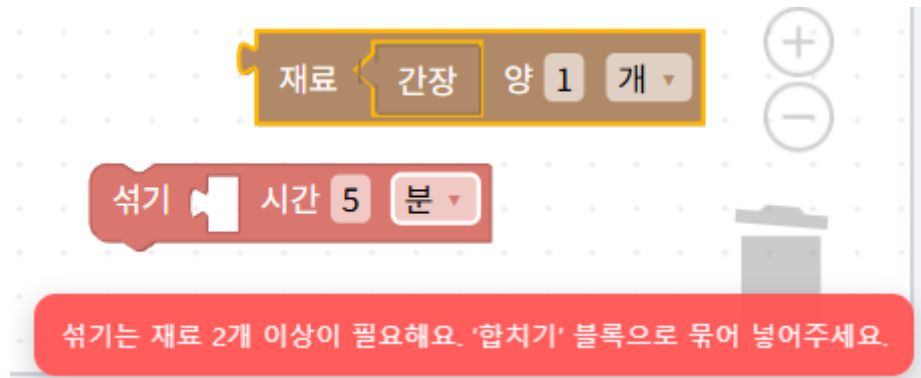
[그림 17] 블록 결합 시 연결 미리보기

그러나 최종 판정은 드랍 직후에 이루어진다.

b. 드랍 직후 의미 판정 → 실패 시 즉시 뒤로가기(undo())

의미 규칙을 위반한 결합은 즉시 이전 상태로 되돌린다(undo).

동시에 토스트(Toast)로 “왜 안 되는지”를 사용자에게 명확히 보여준다(예: “섞기는 재료 2 개 이상이 필요해요.”).



[그림 18] 문법적으로 어긋나는 블록 연결 시 이전 상황으로 되돌아가기 및 토스트 문자 유저에게 보이기

토스트 문자: 유저에게 즉각적으로 정보를 알리기 위해 우측 하단에 1~2 초정도 보여지는 문장형태

c. 동적 입력칸 추가 팝업

“재료 합치기”와 “동작 합치기(준비된 재료)”는 마지막 칸이 찼을 때 ‘입력 칸을 하나 더 추가할까요?’ 팝업을 띄워 사용자 의도를 확인하고, 확인 시 즉시 빈 칸을 1개 확장한다. Blockly 심화 부분에서 block-dynamic-connection 을 설명하며 이미지 첨부을 하였으니 해당부분 참조하길 바란다.

3. 연결 가능/불가능 예시

아래 각 항목은 성공/실패 시나리오를 함께 넣었다. 같은 동작 이름의 상하/좌우 연결블록버전 모두 같은 규칙이 적용된다(메인 요리동작/재료 준비동작 동일 규칙).

a. 재료 이름 → 재료 계량 (ING_NAME → ING)

성공:

재료블록 감자 → 재료 계량블록 입력칸으로 드랍 ⇒ OK

수량과 단위(정수, 드롭다운) 지정 후 재료계량블록 출력은 ING.

실패:

재료블록 감자를 동작의 ITEM 에 직접 드랍 ⇒ undo + 토스트

토스트: “재료 이름은 먼저 ‘재료’ 계량블록에 넣은 뒤 사용 가능합니다.”



[그림 19] 재료블록과 재료 계량 블록 연결 전 상태



[그림 20] 재료 블록을 재료 계량 블록의 입력 칸에 연결한 상태

b. 재료 합치기 (combine_block)

성공:

재료계량블록(감자 2 개), 재료계량블록(양파 1 개)를 차례로
재료합치기 력칸에 차례로 드랍 ⇒ OK

마지막 칸 채움 후 팝업 확인 시 입력칸 추가.

실패:

재료합치기 블록의 입력에 오는 최외각 블록이 재료계량블록이 아닌
다른 블록(예: 재료블록, 동작관련 블록)을 드랍 ⇒ undo + 토스트

토스트: “재료 합치기에는 ‘재료’ 계량블록만 연결할 수 있어요.”



[그림 21] 재료 합치기 블록에 알맞게 연결한 형태

c. 손질된 재료들 (action_combine_block)

성공:

메인 요리동작의 자르기 블록에 재료 계량블록으로 감자 1 개, 메인 요리동작의 갈기 블록에 재료 계량블록으로 무 1 개 등을 손질된 재료들 블록의 첫번째 두번째 입력칸에 드랍 ⇒ OK

이 블록의 출력은 값(좌/우)이므로 다음 동작의 ITEM 또는 다시 동작 합치기로 이어 붙일 수 있다.

실패

최외각 블록이 재료계량블록 또는 재료블록을 입력에 드랍 ⇒ undo + 토스트

토스트: “동작 합치기에는 동작(값) 블록이나 결과 값만 연결할 수 있어요. 재료는 먼저 ‘재료’ 계량블록에 넣어 값을 만든 뒤 사용하세요.”



[그림 22] 손질된 재료들 블록에 재료 계량 블록이 바로 연결될 수 없고 재료 준비 동작 블록이 먼저 연결된 모습

d. 섞기 (mix)

규칙: ING 개수 ≥ 2

ING 를 하나만 넣은 경우 실패.

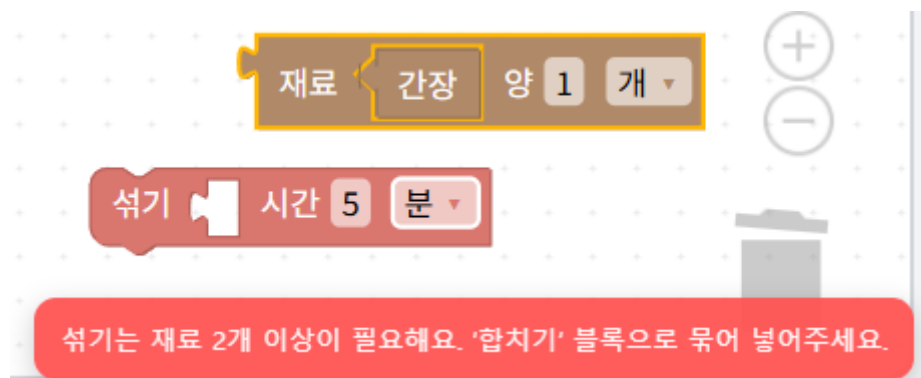
성공 예시:

재료합치기블록에(감자 1, 양파 1) → 두재료를 섞는 동작 ⇒ OK

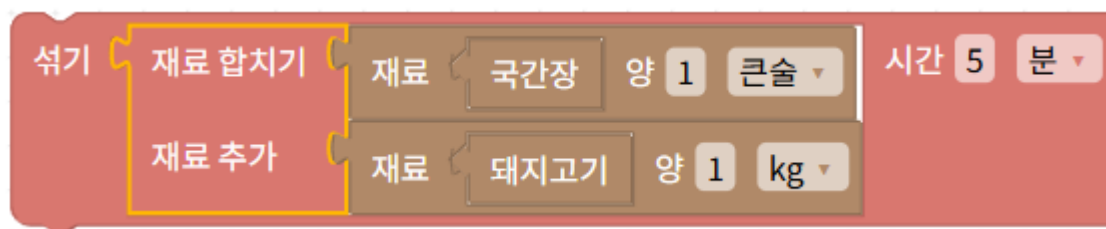
실패 예시:

재료 계량블록에(감자 1 개) 단독 → 한재료를 섞는 동작⇒ undo + 토스트

토스트: “섞기는 재료 2 개 이상이 필요해요. ‘합치기’ 블록으로 묶어 넣어주세요.”



[그림 23] 섞기 블록에 재료를 두개 이상이 아닌 단일로 섞으려 할 때의 모습



[그림 24] 올바르게 연결된 섞기 동작에 대한 모습

e. 볶기 (fry)

규칙: 기름(oil) 필수 + 고체(solid) 또는 가루(powder) 필수 + liquid 금지

성공 예시:

재료합치기블록에 (식용유, 베이컨) → 기름에 튀겨진 베이컨⇒ OK

재료합치기블록에 (버터, 김치, 소금) → OK

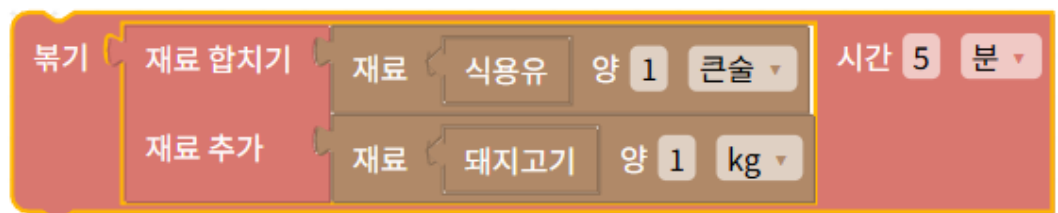
실패 예시:

오일이 없음: 재료계량블록에(김치) → undo + 토스트

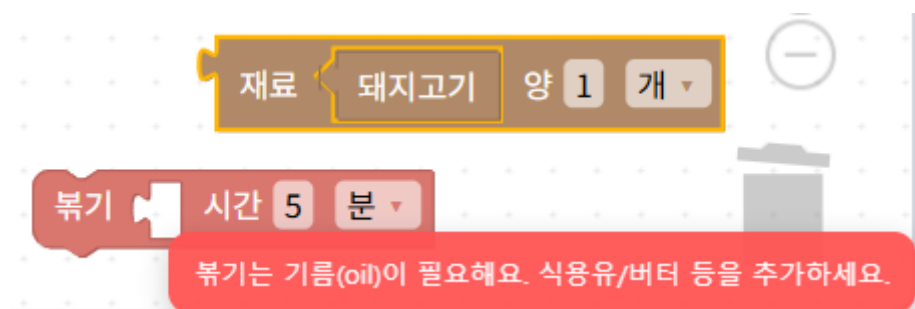
“볶기는 기름(oil)이 필요해요. 식용유/버터 등을 추가하세요.”

액체 포함: 재료합치기블록에(식용유, 간장) → undo + 토스트

“볶기에는 보통 액체는 넣지 않아요.”



[그림 25] 볶기 동작블록에 대해 옳게 연결된 모습



[그림 26] 기름 없이 단일 재료만 들어갔을 때의 모습



[그림 27] 기름이 있지만 액체를 볶으려고 하였을 때의 모습

f. 끓이기 (boil)

규칙: 액체(liquid) 필수

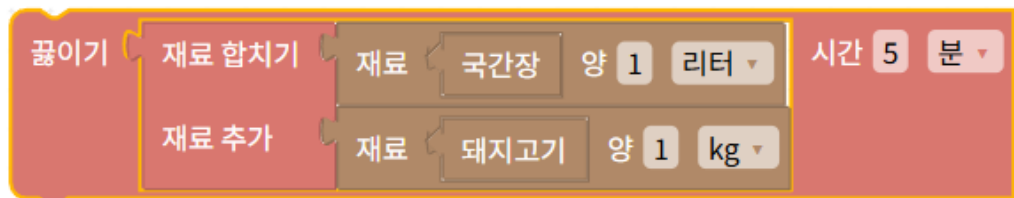
성공 예시:

재료합치기블록에 (물, 파스타면) → 끓여진 파스타면⇒ OK

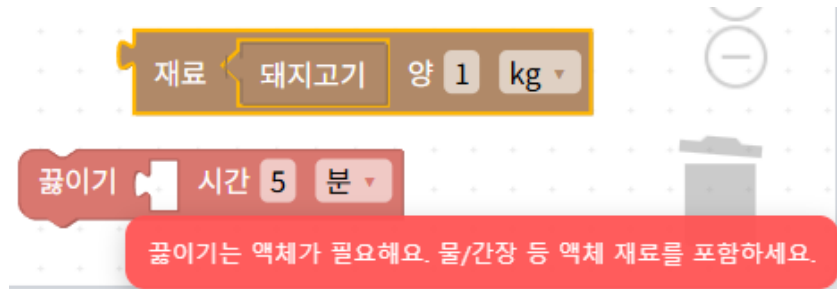
실패 예시:

액체가 없음: 재료계량블록에(파스타면) → undo + 토스트

“끓이기는 액체가 필요해요. 물/간장 등 액체 재료를 포함하세요.”



[그림 28] 끓이기 동작 블록에 대해 옳게 연결된 모습



[그림 29] 액체 없이 단일 재료가 들어갔을 때의 모습

g. 삶기 (simmer)

규칙: 액체(liquid) 필수 + 고체(solid) 필수

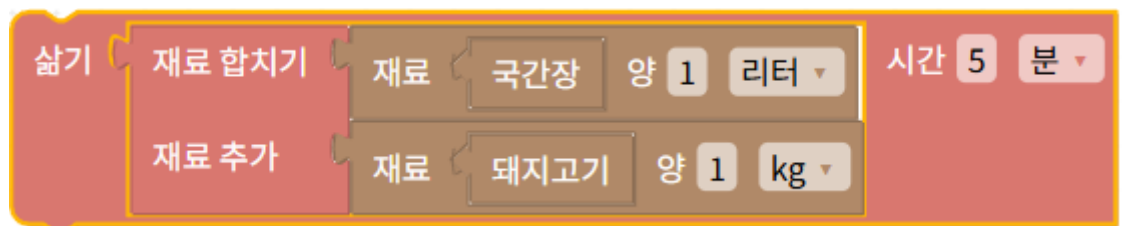
성공 예시:

재료합치기블록에 (물, 소고기) → 삶아진 소고기 ⇒ OK

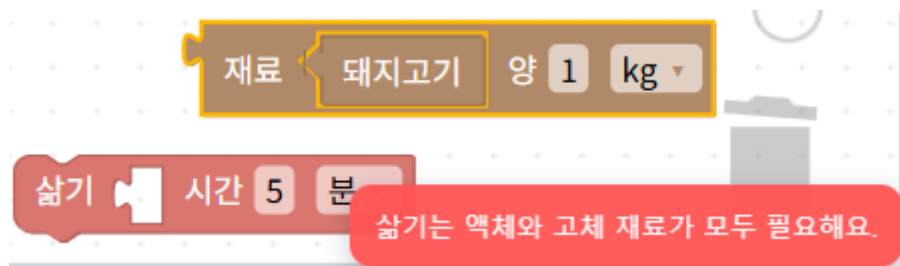
실패 예시:

액체만 있음 / 고체만 있음 → undo + 토스트

“삶기는 액체와 고체 재료가 모두 필요해요.”



[그림 30] 삶기 동작블록에 대해 옳게 연결된 모습



[그림 31] 고체 단일 재료로 삶기 블록을 썼을 때의 모습

h. 자르기 (slice) / 갈기(grind)

slice 규칙: 고체(solid)만 가능.

재료계량블록에 (김치) → 자른 김치 ⇒ OK,

재료계량블록에 (간장) → 자른 간장 ⇒ undo + 토스트

“자르기는 고체 재료에만 사용할 수 있어요.”

grind 규칙: 고체(solid)만 가능

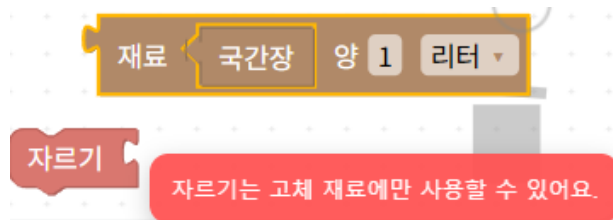
재료계량블록에 (김치) → 간 김치 ⇒ OK,

재료계량블록에 (간장) → 간 간장 ⇒ undo + 토스트

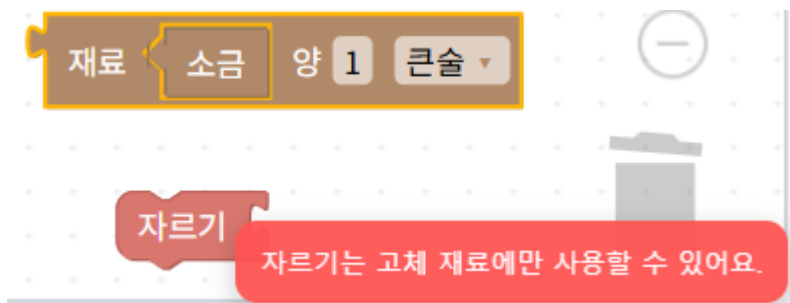
“갈기는 고체 재료에만 사용할 수 있어요.”



[그림 32] 자르기 동작블록에 대해 옳게 연결된 모습



[그림 33] 액체 재료 단일로 자르기 블록에 넣었을 때 모습



[그림 34] 자르기 동작블록에 가루 재료를 사용했을 때의 모습

(갈기도 마찬가지로 이미지 생략)

i. 넣기 (put)

규칙: 제한 없음(모든 ING 허용).

성공 예시:

넣기 블록에(재료 합치기 블록에 (식용유, 국간장, 돼지고기, 소금)

→ 여러 재료를 넣기 ⇒ OK



[그림 35] 넣기 블록에 다양한 종류의 재료가 성공적으로 들어가 있는 모습

상황	메시지	처리
ING_NAME을 동작 블록 ITEM에 직접 드랍	"재료 이름은 먼저 '재료' 계량블록에 넣은 뒤 사용 가능합니다."	undo
재료 합치기 입력에 ingredient_block 외의 블록	"재료 합치기에는 '재료' 계량블록만 연결할 수 있어요."	undo
동작 합치기 입력에 재료 /ING_NAME 드랍	"동작 합치기에는 동작(값) 블록이나 결과 값만 연결할 수 있어요. ..."	undo
mix 에 ING < 2	"섞기는 재료 2개 이상이 필요해요. '합치기' 블록으로 묶어 넣어주세요."	undo
fry에 oil 없음	"볶기는 기름(oil)이 필요해요. ..."	undo
fry에 liquid 포함	"볶기에는 보통 액체는 넣지 않아요."	undo
boil에 liquid 없음	"끓이기는 액체가 필요해요. ..."	undo
simmer에 liquid/solid 부족	"끓이는 액체와 고체 재료가 모두 필요해요."	undo

slice에 solid 아님	"자르기는 고체 재료에만 사용할 수 있어요."	undo
grind에 solid 아님	"갈기는 고체 재료에만 사용할 수 있어요."	undo

[표 1] 문법적 오류 정리 표

3.3. 사용자 인증 및 권한 관리

본 프로젝트에서는 사용자 인증 및 권한 관리를 위해서 이메일 인증을 통한 회원가입, 비밀번호 수정이 가능하도록 하였고 로그인 시에 JWT를 사용하도록 하였다.

3.3.1. 이메일 인증 구현

1. 이메일 인증 시스템 개요

이메일 인증 시스템은 사용자의 이메일 주소가 유효한지를 확인하는 기능으로 회원가입, 비밀번호 재설정 등 여러 인증 절차에 사용된다. 이 시스템은 고유한 인증 코드를 사용자의 이메일로 발송하고 해당 코드를 사용자가 입력하여 인증을 완료하는 방식으로 동작한다.

목적

사용자가 사용 가능한 이메일을 입력했는지 확인하고 이메일을 통해 제공되는 인증 코드를 통해 서비스 이용 여부를 결정하는 기능 제공

사용된 기술

- Spring Boot: 이메일 인증 시스템을 구현하는 기본 프레임워크
- JavaMailSender: 이메일 발송을 위한 Spring 기반 라이브러리
- ConcurrentHashMap: 인증 코드와 이메일 주소를 안전하게 관리하기 위해 사용된 자료구조
- ScheduleExecutorService: 인증 코드에 설정된 TTL (Time-To-Live)을 관리하여 코드의 유효기간을 자동으로 만료시키는 스케줄링 처리

2. 이메일 인증 시스템의 동작 원리

이메일 인증 시스템은 이메일 발송 → 인증 코드 검증 → TTL 처리, 총 3가지의 단계를 통해 구성된다.

a. 사용자가 이메일 입력 후 인증번호 요청

사용자가 이메일을 입력하면, 이메일 중복 여부를 먼저 확인한다. 중복되지 않으면 인증 코드를 입력된 이메일로 인증 코드를 발송한다.

b. 인증 코드 생성 및 관리

인증 코드는 랜덤 숫자 생성 방식을 사용하여 100000 부터 999999까지의 범위 내에서 6가지 숫자가 생성된다. 숫자가 무작위로 예측 불가능하게 생성되므로 보안성을 강화할 수 있다.

생성된 인증 코드는 ConcurrentHashMap에 이메일과 함께 저장되며 3분동안 유효하고 이 기간이 지나면 자동으로 삭제된다.

c. 이메일 발송

JavaMailSender를 사용하여 생성된 인증 코드를 포함한 이메일을 사용자에게 발송한다.

d. 인증 코드 검증

사용자가 이메일로 받은 인증 코드를 시스템에 입력하면, ConcurrentHashMap에서 해당 코드를 조회하고 일치 여부를 확인한다.

코드가 일치하면 인증이 성공적으로 처리되고, 다음 프로세스가 진행된다.

코드가 일치하지 않거나 만료되었으면 오류 메시지를 반환한다.

e. TTL 관리

인증 코드는 3분의 유효 기간을 가지며, ScheduledExecutorService를 사용하여 일정 시간이 지나면 자동으로 만료되도록 설정한다. 이는 보안성을 강화하고 불필요한 인증 코드 저장을 방지한다.

3. 이메일 인증의 필요성

본 프로젝트에서는 사용자 정보와 사용자가 생성한 레시피가 이메일로 연계되므로, 이메일은 단순 연락처가 아니라 사용자 식별자이자 데이터 소유권의 기준 키이다.

이에 따라 이메일 인증을 필수 절차로 두어 중복을 원천 차단하며, 검증 완료 후에는 변경이 불가능하도록 기본 정책을 설계하였다.

3.3.2. JWT

1. 개요

본 프로젝트는 사용자 계정과 사용자가 생성한 레시피를 이메일로 연계한다. 이에 따라 서버 세션을 두지 않고, 요청마다 신원을 증명할 수 있는 JWT(Java Web Token) 기반 무상태(Stateless) 인증을 채택했다.

JWT에는 최소한의 신원(이메일), 만료 정보만 담아 전송 비용과 노출 위험을 줄이고, 이메일 인증을 통과한 사용자에게만 토큰을 발급해 데이터 소유권과 무결성을 보장한다.

2. JWT 간단 정의

JWT는 인증에 필요한 정보를 JSON 형태의 토큰에 담고, 서명을 붙여 위변조를 방지하는 표준 형식(RFC7519) 이다. 토큰 자체에 신원과 유효기간을 포함하므로 서버 측 세션 없이도 인증을 처리할 수 있다.

JWT 자체가 인증/권한 부여 방법을 정하는 것은 아니며, 본 프로젝트에서는 인증 결과를 전달하는 그릇으로 사용한다.

3. 도입 배경

a. 도메인 특성

이메일이 사용자 식별자이자 레시피 소유권의 기준 키이다. 요청마다 이메일 단위로 소유자 일치 여부를 확인하기 용이해야 한다.

b. 아키텍처 요구

다수 클라이언트에서 공통 방식으로 사용 가능하고, 세션 공유 없이도 수평 확장이 쉬운 인증이 필요하다.

c. 운영 단순성

서버 재시작, 배포 시 세션 유지 부담이 없다.

4. 설계 원칙

무상태: 서버 세션 저장 없음

최소 클레임: 이메일(주체), 발급/만료만 포함

단기 만료: 짧은 유효기간 운영

키 보안: 비밀키는 시크릿으로 관리, 소스 저장 금지

5. 토큰 구조와 클레임

헤더: HS256 서명 알고리즘 사용

페이로드(본 프로젝트 기준)

sub: 사용자 이메일(검증 완료 계정의 식별자이자 레시피 소유권 키)

iat: 발급 시각

exp: 만료 시각(애플리케이션 설정으로 관리)

서명: 서버 보유 비밀키(HMAC)로 서명해 위변조 방지

6. 이메일 인증과의 결합

도메인 불변식: 레시피 생성/수정/공개 등 쓰기 권한은 이메일 인증 완료 + 유효한 JWT를 가진 사용자에게만 허용

검증 레이어링: 필터에서 신원 검증(이메일 추출) → 서비스 계층에서 소유자 일

3.4. 데이터베이스

3.4.1. MongoDB 소개 및 선택 이유

MongoDB는 대표적인 NoSQL 데이터베이스로, JSON과 유사한 문서 구조로 데이터를 저장한다. 이는 전통적인 관계형 데이터베이스와 달리, 고정된 스키마 없이 문서마다 유연한 구조를 가질 수 있다는 장점이 있다. 따라서 데이터 구조가 자주 변하거나, 다양한 형태의 데이터를 저장해야 하는 경우에 특히 적합하다.

본 프로젝트에서 사용된 Blockly는 사용자가 제작한 블록 프로그램을 직렬화하여 저장·불러오는 기능을 제공한다. Blockly에서는 XML과 JSON을 모두 지원하지만, 공식 문서에서 새로운 작업공간의 직렬화에는 JSON 사용을 권장하고 있다.

따라서 본 연구에서는 레시피를 Blockly 워크스페이스 JSON 그대로 직렬화하여 데이터베이스에 저장하고, 다시 불러올 수 있도록 설계하였다.

관계형 데이터베이스는 모든 데이터가 미리 정의된 테이블 스키마를 따라야 한다. JSON 데이터를 저장하려면 스키마에 맞게 분해하거나, 문자열로 변환해 저장해야 하는데, 이는 데이터 구조를 변경할 때마다 스키마 수정이 필요하다. 또한 블록 구조처럼 계층적이고 중첩된 데이터를 표현하기에는 비효율적이다.

이와 비교해 MongoDB는 JSON 데이터를 BSON으로 내부 저장하기 때문에 변환 없이 직관적으로 다룰 수 있고, 필요할 경우 JSON 내부 필드 기준으로 바로 쿼리, 검색이 가능하다. 따라서 블록 기반 레시피 데이터를 구조 그대로 보존할 수 있으며, 데이터 구조가 바뀌더라도 스키마를 강제하지 않아 유지보수가 용이하다.

데이터베이스의 배포, 운영, 보안을 자동화하여 관리 부담을 줄여준다. 특히 무료로 사용할 수 있는 클러스터도 제공하기 때문에, 본 프로젝트와 같이 개발 및 연구 목적의 서비스에서 빠르고 간편하게 활용할 수 있었다. Atlas를 사용함으로써 데이터베이스 서버를 직접 구축할 필요 없이, 안정적이고 확장 가능한 환경을 마련할 수 있었다.

3.4.2. DB 저장 구성

본 프로젝트에서는 데이터를 두 개의 주요 컬렉션(users, recipes)로 나누어 관리하였다.

```
_id: ObjectId('6881111c7d4487552c74ea36')
name: "홍길동"
email: "example@email.com"
password: "$2a$10$ZnfoLJAUiIFssjI.w1RKF.1NtDX5wfb9kLCssBgq/M4qeVpuk4A56"
_class: "com.firstsnow.blockchef.domain.User"
```

[그림 36] users 컬렉션 안 Document 예시

users 컬렉션 필드 설명

id: 각 사용자를 고유하게 식별하는 ID (MongoDB의 ObjectId).

name: 사용자의 이름.

email: 로그인 및 사용자 식별을 위한 고유 이메일.

password: 사용자 비밀번호로, Spring Security의 BCryptPasswordEncoder를 이용해 BCrypt 알고리즘으로 암호화하여 저장된다.

users 컬렉션은 레시피 서비스의 사용자 계정을 관리한다. 특히 email 필드는 다른 컬렉션과의 관계를 형성하는 주요 키로 활용된다.

```
_id: ObjectId('68b54ffa47423df227b32fc4')
title: "라면"
description: "간단한 기본 라면 끓이기"
ownerEmail: "hyyyh0x@pusan.ac.kr"
blockStructure: "<xml xmlns='https://developers.google.com/blockly/xml'><block type='bo..."
tags: Array (1)
updatedAt: 2025-09-01T07:49:14.676+00:00
_class: "com.firstsnow.blockchef.domain.Recipe"
```

[그림 37] recipes 컬렉션 안 Document 예시

recipes 컬렉션 필드 설명

id: 레시피 문서의 고유 식별자.

title: 레시피 이름.

description: 레시피 상세 설명(생략 가능).

ownerEmail: 레시피 소유자 이메일, users 컬렉션의 email과 연결되어 소유자를 구분한다.

blockStructure: 레시피를 표현하는 Blockly JSON 직렬화 데이터. 사용자가 만든 블록 구조가 그대로 저장된다.

tags: 레시피 저장 시 입력하는 태그. 나의 레시피에서 레시피를 태그별로 구분할 때 사용된다.

updatedAt: 레시피가 마지막으로 수정된 시각.

recipes 컬렉션은 실제로 사용자가 작성한 요리 과정을 저장하는 핵심 데이터셋이다. 블록 구조를 JSON 그대로 보관하기 때문에, 데이터 조회 시 다시 Blockly 워크스페이스로 복원할 수 있다.

3.5. 배포 자동화

1. 자동배포 도입 배경

개발 초기 단계에서는 프로젝트를 EC2 서버에 수동으로 배포하는 방식을 사용했다. 개발자가 직접 SSH로 접속하여 코드를 pull하고, 빌드 및 실행을 명령어를 통해 수작업으로 수행하는 방식은 비교적 단순해 보이지만, 실제로는 다음과 같은 문제점들을 야기했다.

첫째, 실수의 가능성이 높다. 명령의 입력 실수나 설정 누락 등으로 인한 배포 실패가 반복되었었다.

둘째, 반복 작업에 많은 시간이 소요된다. 특히 처음 배포 이후 개발 단계에서 프론트와의 협업 과정에서 코드가 자주 변경되어 배포 주기도 짧아졌고, 그에 따라 반복적인 수동 작업이 개발 생산성에 부정적인 영향을 주었다.

셋째, 팀원 간의 서로 다른 배포 방식이 문제가 되었다.

이러한 문제들을 해결하기 위해 자동화된 배포 파이프라인을 구축하게 되었다. GitHub Actions를 활용한 CI/CD 환경을 구성함으로써, 코드가 master 브랜치에

병합되는 시점에 자동으로 EC2 서버에 배포가 진행되도록 하였다. 이로 인해 배포 안정성과 반복 작업의 효율성을 크게 향상시킬 수 있었다.

2. 배포 대상 및 환경 구성

배포에 이용한 서버는 AWS EC2 인스턴스를 선택했다. EC2는 AWS에서 제공하는 가상 서버로, 다양한 크기와 성능 옵션을 제공하며, 실시간 모니터링, 유연하게 확장 가능한 인프라를 지원한다. EC2의 프리티어를 이용하여 개발 초기 단계에서 비용 부담을 최소화할 수 있다.

도메인은 blockchef.store로 지정하여, EC2 서버의 퍼블릭 IP를 가리키는 도메인 설정을 완료했다. AWS의 Route 53을 사용하여 도메인과 인스턴스를 연결하였고, SSL 인증서를 설정하여 보안 통신을 보장했다.

3. CI/CD 도구 구성

GitHub Actions 사용

master 브랜치 푸시 시 배포 자동 트리거

CI/CD (Continuous Integration / Continuous Deployment)는 소프트웨어 개발에서 코드 변경을 자동으로 빌드하고 배포하는 과정으로, 빠른 개발과 배포를 가능하게 해준다. 이를 통해 개발자는 코드 변경을 쉽게 반영하고 안정적인 배포 환경을 유지할 수 있다. 본 프로젝트에서는 GitHub Actions를 사용하였다.

GitHub Actions는 코드 변경 시 자동으로 빌드, 테스트, 배포 등을 실행할 수 있는 파이프라인을 정의할 수 있다.

- a. GitHub와의 통합: 깃헙 리포지토리와 바로 통합되어 별도의 설정 없이 소스 코드 변경시 자동으로 실행된다
- b. 확장성: 다양한 액션을 제공하여, 필요한 기능을 손쉽게 추가할 수 있으며, 커스터마이징도 가능하다.
- c. 유연성: Docker, AWS 등 다양한 환경과의 연동을 지원한다.

4. 자동 배포 흐름 요약

GitHub Actions에서 워크플로우는 `.github/workflows` 디렉토리 안에 정의된다. 워크플로우는 여러 개의 잡(jobs)로 구성되며 각각 특정 작업을 수행한다. 본 프로젝트에서는 배포 프로세스를 자동화하기 위해 다음과 같이 정의하였다:

- a. 배포 트리거: master 브랜치에 푸시할 때마다 자동으로 배포가 실행된다.
- b. 배포 준비: 코드가 푸시되면 GitHub Actions가 먼저 소스 코드를 받아오고, 필요한 의존성들을 설치한다.
- c. 빌드 및 테스트: 이후 Gradle 빌드 명령을 통해 프로젝트를 컴파일하고, 테스트를 수행한다. 테스트가 실패하면 배포가 중지된다.
- d. 배포 스크립트 실행: 빌드가 성공하면 EC2 서버에 SSH로 접속하여, 최신 코드를 배포하고, 서비스를 재시작한다.
- e. 배포 완료 알림: 배포가 완료되면 로그에 "배포 완료" 메시지가 출력되며, GitHub Actions 대시보드에서 상태를 확인할 수 있다.

5. 보안 및 구성 관리

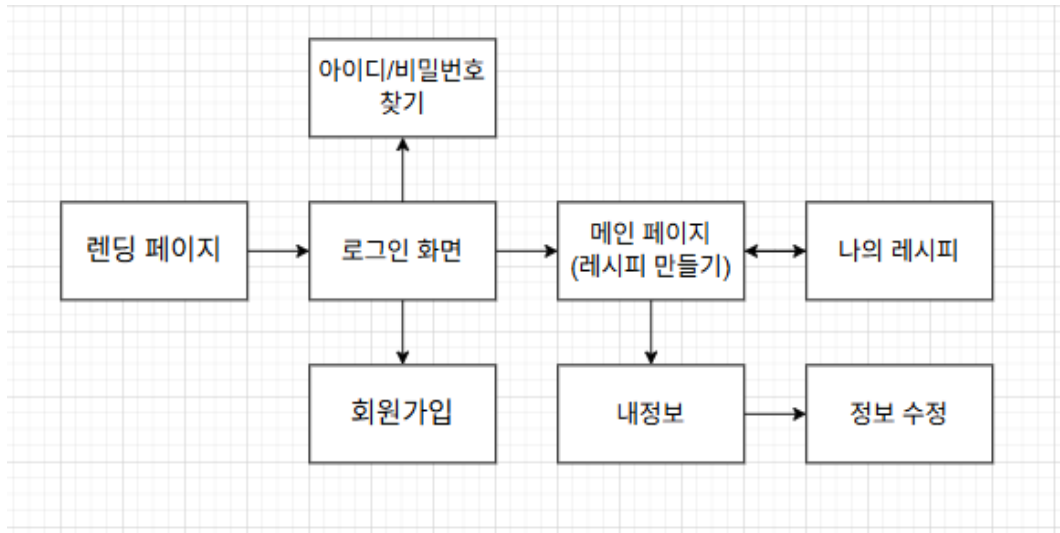
GitHub Actions에서는 민감 정보(예: SSH 키, 데이터베이스 비밀번호 등)을 안전하게 관리하기 위해 GitHub Secrets를 사용한다. secrets에 저장된 값은 환경 변수로 자동 주입되어 배포 과정에서 민감 정보를 코드에 노출시키지 않고 안전하게 사용할 수 있다. 주요 secrets 항목들은 다음과 같다:

- a. EC2_HOST: EC2의 퍼블릭 IP 주소
- b. EC2_USERNAME: EC2 인스턴스의 사용자명
- c. EC2_KEY: SSH 접속을 위한 개인 키
- d. APP_SECRET_PROPERTIES: 민감한 설정 파일의 내용(예: 이메일, 비밀번호 등)

3.6. 프로젝트 기능 설명

3.6.1. 라우팅 구조

서비스 전체 라우팅 구조는 다음과 같다.



[그림 38] 페이지간 라우팅 구조도

3.6.2. 페이지별 설명

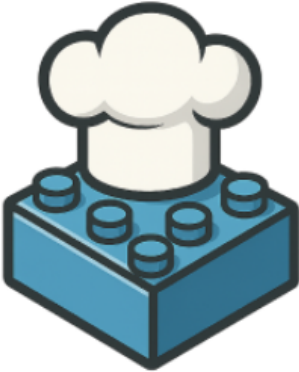
1. StartPage



[그림 39] 랜딩페이지

-
- a. 랜딩페이지로 사용자가 본 프로젝트의 첫 웹사이트에 들어왔을 때 보이는 페이지로 본 프로젝트에 대한 간단한 소개와 블록을 이용해 레시피를 만드는 요리용 코딩언어 튜토리얼이 있음

2. SignIn



로그인

이메일

비밀번호

로그인

[Forgot ID or password?](#)

아직 계정이 없으신가요? [Create ID](#)

[그림 40] 로그인 페이지

- a. 가입한 아이디와 비밀번호를 입력함으로써 JWT 발급을 위한 로그인 폼. 성공 시 토큰을 로컬 스토리지에 저장하고 /main으로 이동. 이메일 형식 유효성 검사 및 비밀번호는 보이지 않게 처리하며 불일치시 유저에게 바로 UI로 볼 수 있게 설계
- b. 간단한 디자인으로 유저에게 기능적으로 쉽게 적응할 수 있도록 설계

3. SignUp

회원가입

general706@pusan.ac.kr

남은 시간: 2:36

인증번호

인증번호 보내기 인증하기

[그림 41] 회원가입 페이지(인증페이지)

회원가입

general706@pusan.ac.kr

이름

비밀번호

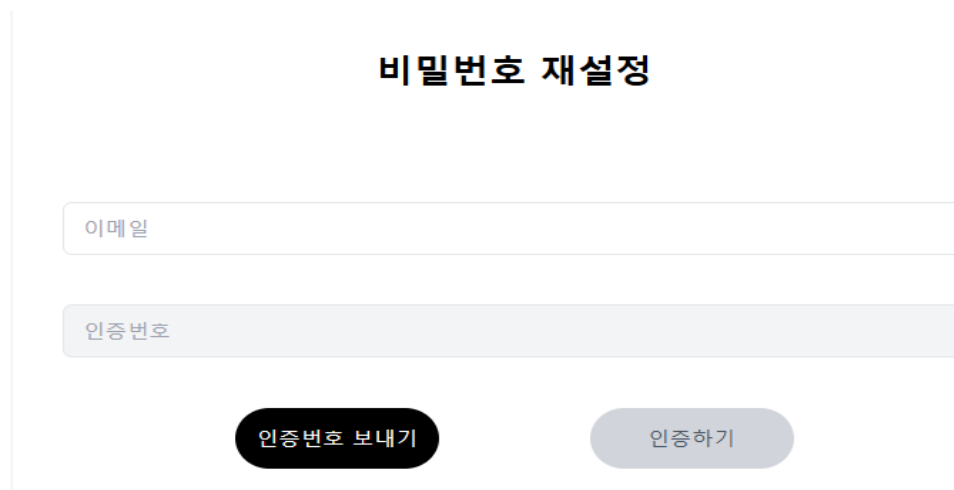
비밀번호 재입력

회원가입

[그림 42] 회원가입 페이지(인증 후 개인정보 입력 페이지)

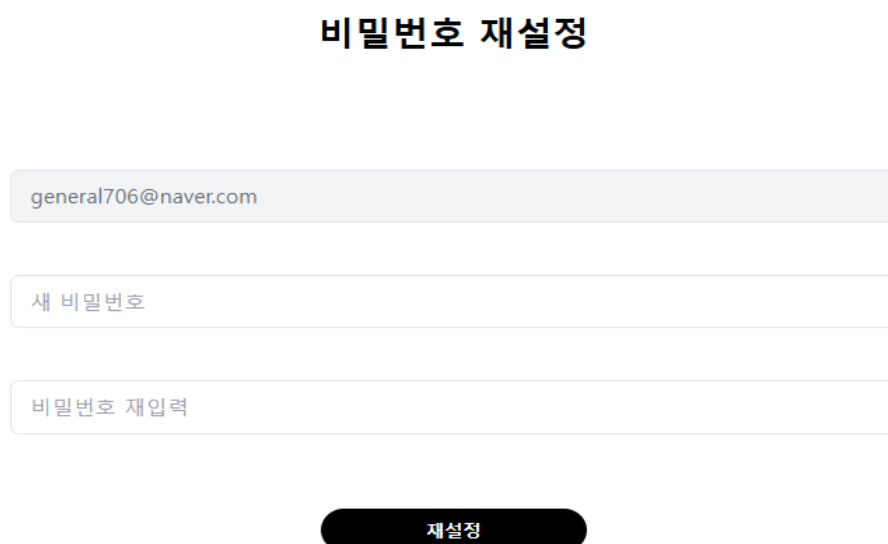
-
- a. 회원가입을 하기 위한 페이지. 사용자의 이메일을 입력하고 인증번호 보내기 버튼을 누르면 입력한 이메일로 인증번호가 전송되며 비활성화 되어있던 인증하기 버튼이 활성화 되며 인증번호를 입력 후 인증하기버튼을 누르면 인증이 완료되며 본격적인 회원가입을 위한 페이지로 넘어간다.
 - b. DB 에 유저를 이메일, 이름, 비밀번호 등으로 저장을 하기에 해당 정보를 입력하게 한 후 여기서도 비밀번호에 대해 유효성 검사를 진행 후 회원가입을 진행하게 된다.

4. PasswordReset



The image shows a web form titled "비밀번호 재설정" (Reset Password). It contains two input fields: "이메일" (Email) and "인증번호" (Verification Code). Below the "이메일" field is a black button labeled "인증번호 보내기" (Send Verification Code). Below the "인증번호" field is a grey button labeled "인증하기" (Verify). The form is enclosed in a light grey border.

[그림 43] 비밀번호 재설정 페이지(인증페이지)

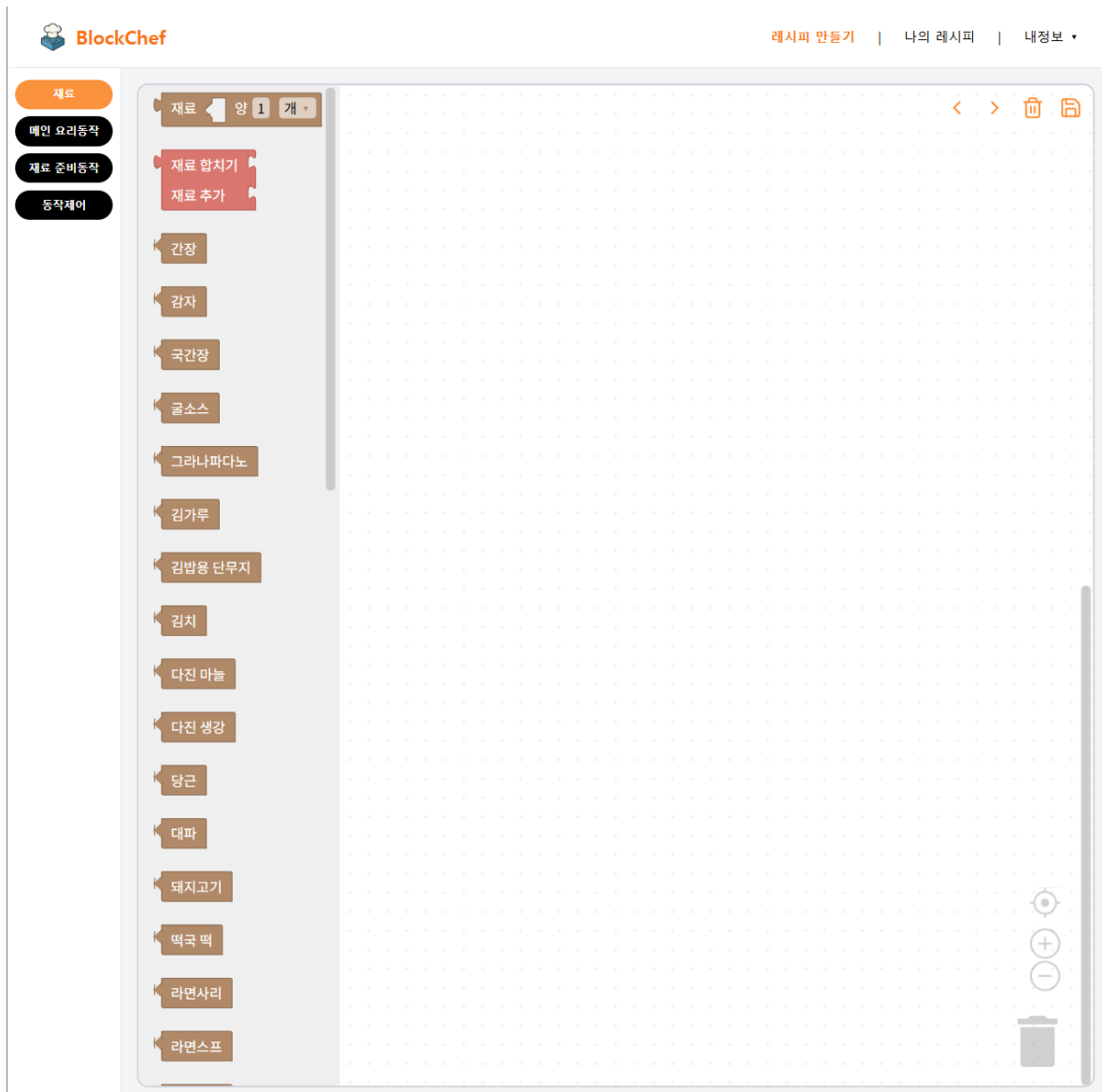


The image shows a web form titled "비밀번호 재설정" (Reset Password). It contains three input fields: "general706@naver.com" (Email), "새 비밀번호" (New Password), and "비밀번호 재입력" (Re-enter Password). Below the "새 비밀번호" and "비밀번호 재입력" fields is a black button labeled "재설정" (Reset). The form is enclosed in a light grey border.

[그림 44] 비밀번호 재설정 페이지(인증 후 새 비밀번호로 재설정하는 페이지)

- a. 비밀번호를 재설정하기 위한 페이지. 이메일을 입력 후 인증번호 보내기 버튼을 누르면 입력한 이메일로 인증번호가 전송되며 비활성화 되어있던 인증하기 버튼이 활성화 되며 인증하기 버튼을 누르면 인증이 완료됨.
- b. 그후 바로 비밀번호를 재설정 할 수 있게 하며 이때도 유효성 검사를 통해 비밀번호를 재설정 할 수 있도록 한다.

5. MainPage(레시피 만들기)



[그림 45] 로그인 직후 보이는 메인 페이지(레시피 만들기 페이지)

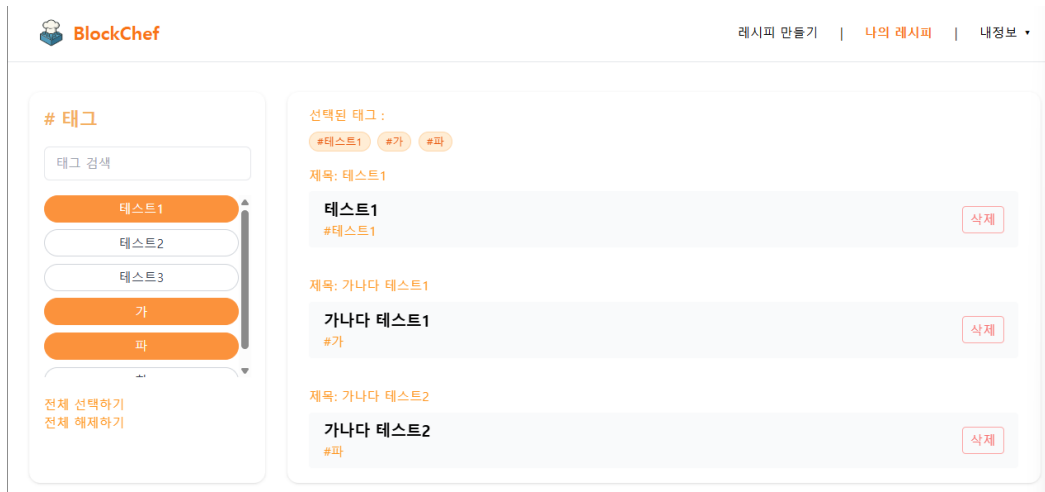
- a. 본 프로젝트의 가장 핵심이 되는 페이지로 블록을 연결해가며 레시피를 만들고 저장 시 태그를 같이 입력하는 페이지. 메인 페이지에는 Blockly 라는 구글에서

만든 라이브러리를 활용하여 React 기반 개발환경에서 블록들을 '재료', '메인 요리동작', '재료 준비동작', '동작제어' 등의 카테고리로 분류 하였음

- b. 각각의 블록들 사이에 짜임새 있는 문법을 정하여 서로 결합 가능한 블록과 그렇지 못한 블록들에 대해 연결부분의 모양에 따라 사용자가 구분할 수 있는 요소도 있으며 연결부분의 모양이 같지만 연결이 안되는 경우엔 우측하단의 토스트 문자로 유저에게 즉각적으로 알려줄 수 있게 설계하였음.
- c. 사용하기전 블록이 놓여져 있는 '팔레트영역'과 그 팔레트 영역에 놓여져 있는 블록을 드래그 앤 드롭 형식으로 드래그 해와서 블록으로 작업하는 '작업영역'으로 영역을 나누어 각영역에 대해 블록에 대한 정의와 쓰임새를 명확히 구분함으로 블록끼리 연결한 작업물이 저장 시 혼선을 일으키지 않게 함.
- d. 작업영역의 우측상단의 4 개의 버튼들이 있는데 왼쪽에서부터 차례대로 Undo(뒤로 가기), Redo(앞으로 가기), Delete(작업영역에 있는 모든 블록 삭제), Save(제목, 태그 입력하며 저장)기능을 가진다.
- e. 블록과 문법에 대해선 뒤에 자세히 기술하겠음.
- f. 저장 시 태그 입력하는 팝업창(제목과 태그는 입력이 필수로 설계)

[그림 46] 레시피 저장 시 뜨는 팝업 창(제목 및 태그 입력)

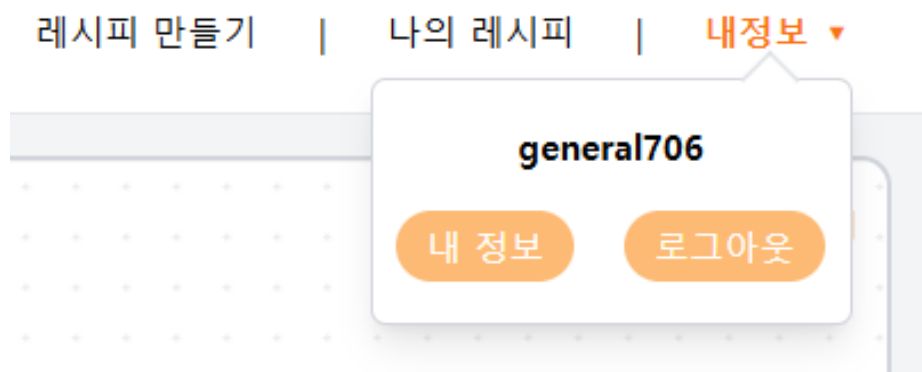
6. MyRecipe



[그림 47] 나의 레시피 페이지

- 메인페이지에서 저장했던 나의 레시피를 볼 수 있는 페이지.
- 레시피 저장시 태그가 같이 저장되는데 그때 입력했던 태그들이 해당 페이지 좌측에 형성이 되고 위의 이미지와 같이 선택한 태그들이 사용자에게 명확히 구분가게 태그가 오렌지색으로 보이게 설계하였음.
- 태그 전체선택/해제 기능을 구현하였으며 태그가 전체 해제된다면 레시피 제목을 가나다 형식으로 정렬함.
- 저장된 레시피를 클릭하면 작업했던 블록들이 위치 그대로 유지된채 메인페이지(레시피 만들기)로 이동함.
- 태그 검색 기능 구현함.

7. MyInfo



[그림 48]. 내 정보 클릭 시 드롭다운 형식으로 보이는 말풍선 모양 팝업 창

BlockChef

레시피 만들기 | 나의 레시피 | 내정보 ▼

내 정보

이름: 정종현

이메일: general706@naver.com

비밀번호: ●●●●●●

회원탈퇴 수정하기

[그림 49] 내 정보 페이지

- 상단 우측의 메뉴 카테고리들 중 내정보를 누르게 되면 작은 말풍선 모양의 팝업창이 생김.
- 이때 내정보를 누르게 되면 내 정보를 볼수있는 페이지.
- 회원가입시 입력했던 이름, 이메일 그리고 비밀번호가 있는데 비밀번호는 보안을 위해 이미지에 보이는 바와 같이 처리함.

8. InfoEdit

BlockChef

레시피 만들기 | 나의 레시피 | 내정보 ▼

내 정보 수정

이름: 정종현

이메일: general706@naver.com

비밀번호: ●●●●●●

비밀번호 확인: ●●●●●●

수정 확인

[그림 50] 회원 정보 수정 페이지

내 정보 수정

이름: 정종현

이메일: general706@naver.com

비밀번호: ●●●●●●

비밀번호 확인: ●●●●●●

비밀번호 불일치 수정 확인

[그림 51] 비밀번호 불일치시

내 정보 수정

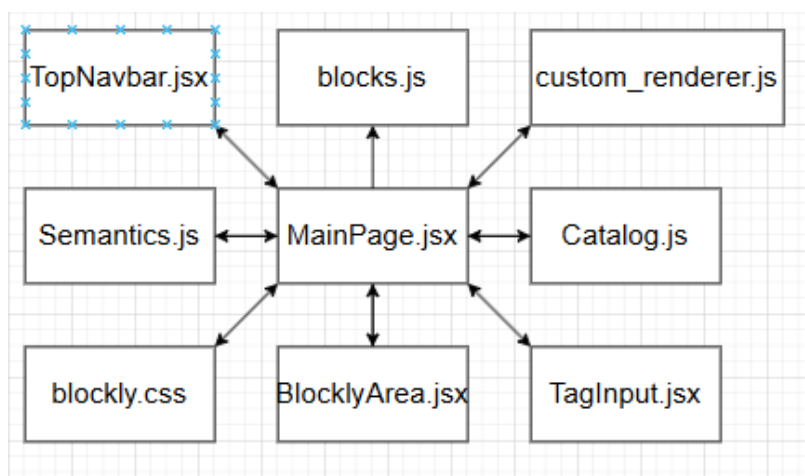
이름:	정종현
이메일:	general706@naver.com
비밀번호:
비밀번호 확인:

비밀번호 일치 수정 확인

[그림 52] 비밀번호 일치시

- MyInfo 페이지에서 수정하기 버튼을 누르고 들어와 회원정보를 수정할 수 있는페이지.
- 이메일은 수정할 수 없게 처리하여 이메일은 블러처리.
- 그외 이름과 비밀번호를 변경할 수 있으며 이때 비밀번호 일치 불일치와 비밀번호 유효성 검사를 진행하게 됨.
- 입력후 수정확인 버튼을 누르면 html 자체의 팝업창이 나오며 수정확인 의사를 묻고 Y/N 중 클릭하여 의사를 결정할 수 있게 함.

3.6.3. 컴포넌트 및 주요 Asset



[그림 53] 주요 컴포넌트 간 관계도

-
- a. blocks.js: 블록들에 대한 기능적, 시각적 정의를 담당
 - b. Semantics.js: 블록들간 문법적 정의를 담당하며 블록간 연결가능 여부를 판단
 - c. BlocklyArea.jsx: 메인 페이지에서 블록에 대한 영역을 나누어 블록이 사용되기 전 진열되어 있는 '팔레트영역'과 블록들을 드래그 앤 드롭 형태로 끌고와 작업하는 영역을 나누어 관리하는 역할을 담당
 - d. custom_renderer.js: 연결부분의 모양을 커스텀하여 직접 모양설계를 할 수 있게 해주는 역할
 - e. TopNavbar.jsx: 메인페이지에서 상단에 위치한 컴포넌트로 레시피 만들기, 나의 레시피, 내정보 드롭다운 팝업 창으로 내정보 및 로그아웃 할 수 있게 해주는 역할을 담당
 - f. Catalog.js: 블록들을 4가지 카테고리로 분류할 수 있게 해주는 역할
 - g. blockly.css: 블록들의 스타일에 대해 정의
 - h. TagInput.jsx: 레시피 저장시 뜨는 팝업 창에서 태그를#으로 시작해서 스페이스바나 ","를 입력하면 태그로 분류하여 하나의 태그로 간주하게 해주는 역할을 담당

4. 연구 결과 분석 및 평가

4.1. API 명세서

기능	HTTP 메서드	API path
로그인	POST	/auth/login
회원가입	POST	/auth/signup
회원가입 이메일 인증 코드 전송	POST	/auth/email/signup/send-code
비밀번호 재설정 이메일 인증 코드 전송	POST	/auth/email/reset-password/send-code
이메일 인증코드 확인	POST	/auth/email/verify-code
비밀번호 재설정	POST	/auth/reset-password

레시피 저장 및 수정	PUT	/recipes
레시피 목록 조회	GET	/recipes/my
레시피 삭제하기	DELETE	/recipes/{id}
레시피 상세 조회	GET	/recipes/{id}
내 정보 조회	GET	/users/me
내 정보 수정	PUT	/users/me
회원 탈퇴	DELETE	/users/me

4.2. 구성원별 역할 분담

201924419 김도엽	백엔드 개발(이메일 인증, 사용자 도메인 등), 백엔드 배포 및 자동 배포, 블록 설계
202255661 박혜연	DB 연결, 프론트엔드 자동 배포, 블록 설계, swagger 구현, 피그마 디자인
201924577 정종현	프론트엔드 개발(서비스 화면, Blockly 구현 등), 블록 설계, 피그마 디자인

5. 결론 및 향후 연구 방향

5.1. 결론 및 기대효과

본 프로젝트를 통해 기존 자연어 기반의 레시피를 블록 형태로 코딩할 수 있는 서비스를 성공적으로 개발하였다. Blockly를 활용한 이 플랫폼은 사용자들이 요리 레시피를 새로운 관점으로 바라볼 수 있는 시각을 제공할 것으로 기대된다. 주요 기대 효과는 다음과 같다.

1. 레시피 구성 요소 확인 재료 블록과 동작 블록으로 레시피를 코딩하면서 사용자는 요리의 각 구성 요소를 명확히 이해하고, 요리 과정에서 필요한 각 단계들이 어떻게 연결되는지 직관적으로 알 수 있다. 이는 요리의 구조를 체계적으로 파악하고 단계별로 분석 및 개선할 수 있는 기회를 제공한다.
2. 코딩에 대한 접근성 확대 실생활과 밀접하게 연결되어있는 요리라는 도메인을 블록코딩과 연결시킴으로 코딩에 익숙하지 않은 사용자들의 접근성을 대폭 확대할 수 있다.

-
3. 요리와 교육적 가치의 융합 블록코딩을 통해 요리 과정을 교육적으로 학습할 수 있는 기회를 제공한다. 특히 어린이나 초보자들이 요리의 각 단계를 체계적으로 이해하고 창의적인 요리 방법을 개발할 수 있도록 돕는다. 이는 코딩 교육과 요리 교육을 동시에 제공한다.

5.2. 향후 연구 방향

본 프로젝트를 바탕으로, 다음과 같은 향후 연구 방향을 생각해 볼 수 있다.

1. 사용자 간의 상호작용 기능 추가 현재 프로젝트에서는 서비스와 사용자 간의 상호작용만 가능하다. 하지만 이후 게시판 기능이나 다양한 소셜 미디어 기능들과 결합하여 여러 사용자 간의 상호작용이 가능하도록 한다.
2. 데이터베이스 구조 세분화 현재 DB의 구조는 레시피 정보를 하나의 컬렉션에서 전부 저장하고 있다. 이 경우 사용자가 많아지면 해당 레시피를 찾는데 처리 시간이 길어지는 문제를 발생시킨다. 따라서 DB의 구조를 세분화하여 유저마다 레시피 컬렉션을 따로 운영하도록 한다.
3. 시각적 실행 결과 추가 현재 레시피를 만들고 저장하는 서비스를 제공하고 있다. 추후 레시피 만들고 실행하면 애니메이션으로 요리 과정을 만들어 준다거나, 사용된 재료, 동작 블록들을 정리하여 개요를 작성해준다거나 하는 시각적 실행 결과를 추가한다.
4. 기계 행동 제어 사용자가 실제 동작하는 요리용 기계를 제어하기 위한 언어로 사용될 수 있다. 예를 들어, 스마트 오븐이나 냄비와 같은 기기와 연동하여, 블록코딩으로 요리 과정의 각 단계를 기계에게 전달할 수 있는 시스템을 개발 할 수 있다. 이를 통해 사용자에게 자동화된 요리 경험을 제공할 수 있으며, 요리 효율성을 높일 수 있다.
5. 다양한 요리 환경 지원 현재는 기본적인 재료와 동작들만을 지원하지만, 향후 요리 도구와 같은 다양한 요리 환경을 지원할 수 있도록 개발 범위를 확장할 수 있다.
6. AI 기반 추천 시스템 레시피 작성에 대한 데이터를 바탕으로 인공지능을 활용한 추천 시스템을 도입할 수 있다. 사용자의 선호도, 재료의 유효성을 기반으로 맞춤형 레시피를 추천하거나, 사용자가 빼먹은 재료들을 감지하고 제시하는 시스템을

개발할 수 있다. 이를 통해 사용자에게 더 개인화된 요리 경험을 제공할 수 있다.

7. 다국어 지원 다양한 언어를 지원하여 글로벌 사용자들이 이 플랫폼을 이용할 수 있도록 한다. 다국적 요리 문화를 고려하여 다국어 번역 및 해당 문화적 요소를 반영한 레시피를 추천하고, 이를 통해 글로벌 사용자층을 형성할 수 있다.

6. 참고 문헌

논문 내용에 직접 관련이 있는 문헌에 대해서는 관련이 있는 본문 중에 참고문헌 번호를 쓰고 그 문헌을 참고문헌란에 인용 순서대로 기술한다. 참고문헌은 영문으로만 표기하며 학술지의 경우에는 저자, 제목, 학술지명, 권, 호, 쪽수, 발행년도의 순으로, 단행본은 저자, 도서명, 발행소, 발행년도의 순으로 기술한다.

[1] Scratch [Online]. Available: <https://scratch.mit.edu/>

[2] Louis Mahon, Carl Vogel, "The Proof is in the Pudding: Using Automated Theorem Proving to Generate Cooking Recipes," 10, Mar 2022

[3] geeksforgeeks "Introduction to Programming Languages" [Online]. Available: <https://www.geeksforgeeks.org/computer-science-fundamentals/introduction-to-programming-languages>

[4] Blockly guide [Online]. Available: <https://developers.google.com/blockly/guides/get-started/what-is-blockly?hl=ko>

[5] MongoDB introduction [Online]. Available: <https://www.mongodb.com/ko-kr/resources/basics/databases/document-databases>