

졸업자격실험보고서

식재료 유통기한을 고려한 레시피 추천
시스템 개발

Development of Recipe Recommendation System
Considering Ingredient Expiry Dates

지도교수 오 승 현

동국대학교 과학기술대학 컴퓨터공학과

신 중 근

2023

졸업자격실험보고서

식재료 유통기한을 고려한 레시피 추천
시스템 개발

Development of Recipe Recommendation System
Considering Ingredient Expiry Dates

신 중 근

지도교수 오 승 현

본 보고서를 졸업자격 실험보고서로 제출함.
2023년 11월 23일

신중근의 졸업자격 실험보고 통과를 인준함.
2023년 11월 23일

주	심	<u>도재수</u>	(인)
부	심	<u>박소현</u>	(인)
부	심	<u>오승현</u>	(인)

동국대학교 과학기술대학 컴퓨터공학과

목 차

1. 서론	4
1.1 연구배경 및 목적	4
1.2 연구범위	4
2. 이론적 배경	5
2.1 기존 연구 및 참고 자료	5
2.2 참고 애플리케이션	5
2.3 기술적 요소	6
3. 시스템 설계	7
3.1 요구사항	7
3.2 아키텍처 설계	7
3.3 상세설계	12
4. 프로그램 구현	13
4.1 시스템 환경	13
4.2 시스템 구성	13
4.3 시스템 구현	14
5. 실험	15
5.1 시스템 구성	15
5.2 성능 테스트	17
6. 결론	19

참고문헌	20
------------	----

부록	21
----------	----

그 립 목 차

[그림 1] 시스템 구성도	8
[그림 2] 유스케이스 다이어그램	8
[그림 3] 클래스 다이어그램	10
[그림 4] ER 다이어그램	11
[그림 5] 로그인 화면	15
[그림 6] 회원가입 화면	15
[그림 7] 홈 화면	15
[그림 8] 네비게이션 바	15
[그림 9] 식재료 입력 화면	15
[그림 10] 바코드 촬영 화면	15
[그림 11] 보유 식재료 화면	16
[그림 12] 랜덤 레시피 출력	17
[그림 13] 레시피 재료, 순서	17
[그림 14] 레시피 검색 화면	17
[그림 15] 레시피 검색 결과	17
[그림 16] 식재료 선택	18
[그림 17] 레시피 추천	18

1. 서론

1.1 연구배경 및 목적

현대 사회에서는 빠르게 변화하는 라이프스타일과 식생활 패턴의 변화로 인해 음식물 낭비가 두드러지게 증가하고 있다. 식료품의 낭비는 환경 문제 뿐만 아니라 경제적인 손실을 야기하며, 지속 가능한 소비에 대한 필요성이 강조되고 있다. 이에 따라, 사용자들이 보유한 식재료를 최대한 효율적으로 활용하여 음식물 낭비를 줄이는 방안이 고려되어야 한다.

본 연구는 이러한 맥락에서 출발하였다. 사용자가 가지고 있는 다양한 식재료들을 고려하여 레시피를 추천하는 시스템을 개발함으로써, 개인의 식생활에 맞춘 소비를 촉진하고 음식물 낭비를 최소화하는 것을 목표로 하고 있다. 이러한 목표를 통해 사용자들이 보다 지속 가능하고 경제적인 음식 소비를 할 수 있도록 돕고자 한다.

1.2 연구범위

본 연구에서의 연구 범위는 주로 사용자의 보유한 식재료를 고려한 레시피 추천 시스템의 개발에 중점을 두고 있다. 이를 위해 사용자가 가진 식재료의 유통기한과 수량 등을 고려하여 가중치를 부여한 새로운 추천 알고리즘을 제안하고자 한다. 또한, 사용자 편의성을 고려하여 바코드를 활용한 자동 입력 기능과 같은 실용적인 기능을 구현하여 사용자가 식재료 입력을 보다 쉽게 해 줄 것이다. 그러나 이 연구에서는 사용자 건강에 맞는 레시피 추천 등의 내용까지 포함하지는 않을 것이며, 이는 향후 연구의 확장 가능성을 남겨두고 있다.

2. 이론적 배경

2.1 기존 연구 및 참고 자료

2.1.1 레시피 데이터 기반 식재료 추천 시스템

과거의 연구 중에서 레시피 데이터를 기반으로 한 식재료 추천 시스템에 대한 연구들이 있었다. 민성희, 오유수의 연구인 "레시피 데이터 기반의 식재료 궁합 분석을 이용한 레시피 추천 시스템 구현"[1]에서는 사용자가 선호하는 음식과 섭취하지 않은 음식을 우선적으로 추천하는 시스템을 소개하고 있다. 이 연구는 사용자 친화적인 추천 시스템을 제공하며, 본 연구에서는 사용자보다 식재료에 중점을 두어 레시피를 추천한다.

2.1.2 텍스트 분석과 식재료 계층구조를 활용한 레시피 추천

홍지현, 이희정의 연구인 "텍스트 분석 기법과 식재료 계층구조를 활용한 음식 레시피 추천 방법"[2]에서는 식재료들과의 궁합이 좋은 레시피 추천, 특이한 식재료를 이용한 레시피 추천, 대체 식재료를 이용한 레시피 추천 등에 대해 다루고 있다. 이 연구는 레시피의 난이도를 표시하여 요리 미숙자도 레시피를 따라할 수 있도록 하는 등의 기능을 제공하고 있다. 이러한 아이디어는 본 연구에서의 사용자 편의성 개선에 영감을 줄 수 있다.

2.1.3 머신러닝을 이용한 이미지 및 문자열 데이터 전처리

김현지, 오유수의 연구인 "머신러닝 프로세스의 이미지 및 문자열 데이터 입력에 대한 전처리 방법 추천 시스템 설계"[3]에서는 문자열 데이터 중 식재료명 데이터의 전처리 방법을 설계하여 레시피에 필요한 식재료에 가중치를 부여하고 있다. 이 프로젝트는 사용자가 선택한 식재료들이 포함된 개수에 따라 레시피에 가중치를 주어 추천하는 방식을 제시하고 있다.

2.2 참고 애플리케이션

2.2.1 Yummly

Yummly[4]는 사용자의 식문화와 취향을 고려하여 맞춤형 레시피를 제공하는 애

플리케이션으로, 수많은 레시피 데이터를 기반으로 사용자에게 최적화된 음식 추천을 제공한다. 또한, 식재료의 선호도와 유통기한 등을 고려한 특화된 추천 알고리즘을 활용하여 음식물 낭비를 최소화하는 데 기여하고 있다.

2.2.2 Too Good To Go

Too Good To Go[5]는 음식점이나 가게에서 남은 음식을 할인된 가격에 판매하는 플랫폼으로, 음식물 낭비를 줄이고 소비자에게 경제적인 이점을 제공하는 애플리케이션이다. 이러한 소비 문화 변화를 통해 음식물 낭비를 주제로 한 연구에서 참고할 만한 측면이 있다.

2.3 기술적 요소

본 연구에서 사용된 중요한 기술적 요소 중 하나는 바코드 인식 기술이다. 바코드를 활용하여 사용자가 보유한 식재료를 자동으로 입력하는 기능은 사용자 편의성에 기여하며, 이는 Too Good To Go 등의 애플리케이션에서도 활용되고 있다. 바코드 인식 기술은 레시피 추천 시스템에서의 효율적인 데이터 입력을 지원하는 중요한 기술 중 하나이다. 이론적 배경에서 소개한 연구와 애플리케이션들은 본 연구의 기초 자료로 활용되었으며, 이를 토대로 레시피 추천 시스템의 특징과 차별성을 더욱 부각시킬 것이다.

3. 시스템 설계

3.1 요구사항

3.1.1 로그인, 회원가입 기능

사용자는 로그인 화면에서 개인 식별을 위해 아이디와 비밀번호를 입력할 수 있다. 또한, 회원가입 화면에서는 신규 사용자가 시스템에 가입하기 위해 필요한 아이디, 비밀번호, 실명, 이메일, 전화번호 등의 정보를 입력할 수 있다. 이를 통해 사용자는 개인 계정을 생성하고 로그인하여 개인화된 서비스를 이용할 수 있다.

3.1.2 식재료 입력

사용자는 식재료 입력 화면에서 새로운 식재료를 등록할 수 있다. 이때 입력해야 하는 정보에는 식재료의 이름, 유통기한, 수량, 구매날짜, 종류 등이 포함된다. 뿐만 아니라, 바코드 스캔 기능을 활용하여 사용자가 직접 입력하는 수고를 덜 수 있다.

3.1.3 보유 식재료 출력

시스템은 사용자가 보유한 식재료를 유통기한과 수량을 고려하여 우선순위에 따라 화면에 표시한다. 또한, 사용자는 각 식재료의 유통기한과 수량을 쉽게 확인하고, 필요에 따라 수정 또는 삭제할 수 있다. 이를 통해 사용자는 보다 효율적으로 식재료를 관리하고 음식물 낭비를 줄일 수 있다.

3.1.4 레시피 검색 및 출력

사용자가 선택한 식재료를 활용한 레시피를 검색하여 최대 10개까지 출력한다. 각 레시피는 이름, 요리 이미지, 간략한 정보, 카테고리, 필요한 식재료, 레시피 내용, 그리고 레시피 이미지 등 다양한 정보를 포함하고 있다. 사용자는 이를 통해 식재료를 효과적으로 활용하여 맛있는 요리를 준비할 수 있다.

3.2 아키텍처 설계

3.2.1 시스템 구성도

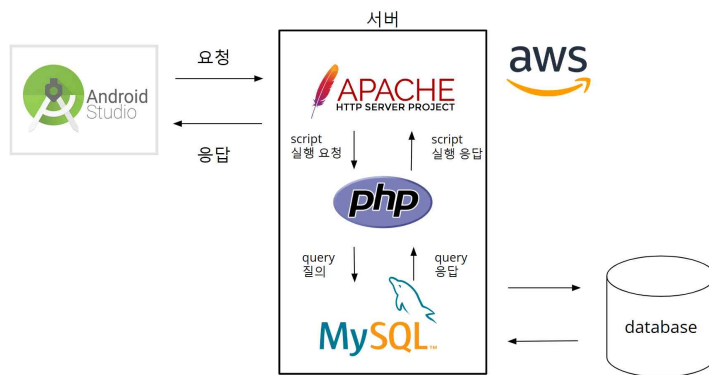


그림 1 시스템 구성도

3.2.1 유스케이스 다이어그램

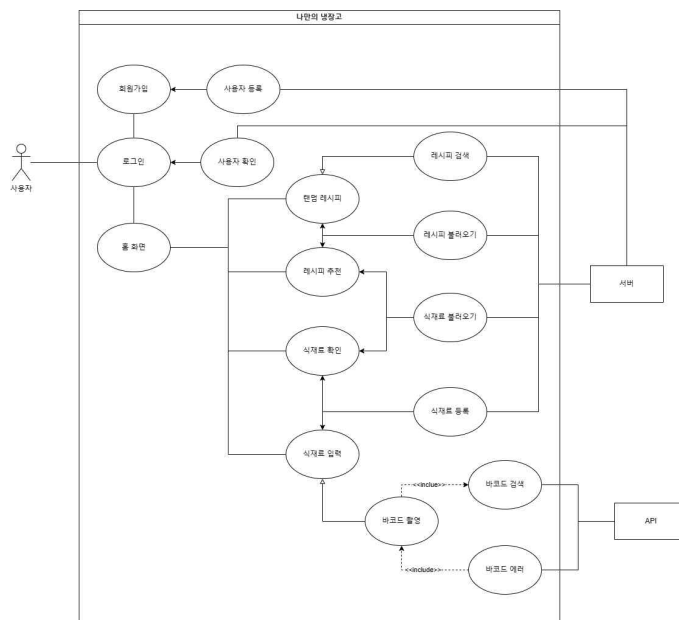


그림 2 유스케이스 다이어그램

3.2.2 유스케이스 명세서

1) 로그인

1.1) 기본 플로우

어플리케이션을 실행한 사용자는 “로그인” 화면으로 이동한다.

사용자는 이미 가입한 계정의 ID와 비밀번호를 입력한다.

시스템은 입력된 ID와 비밀번호를 확인하고, 일치하는 계정이 있는 경우 사용자를 홈 화면으로 로그인시킨다.

1.2) 예외 플로우

사용자가 입력한 ID나 비밀번호가 일치하지 않는 경우, 시스템은 “로그인에 실패했습니다. 다시 시도해주세요.” 메시지를 표시한다.

2) 회원가입

2.1) 기본 플로우

사용자 정보가 없는 사용자는 로그인 화면에서 회원가입 버튼을 클릭하여 “회원가입” 화면으로 이동한다.

사용자는 ID, 비밀번호, 이름, 이메일, 전화번호를 입력한다.

시스템은 입력된 정보를 확인하고, 모든 필수 정보가 제공된 경우 새로운 계정을 생성합니다.

계정 생성 후, 시스템은 사용자 로그인 화면으로 이동시킨다.

2.2) 예외 플로우

사용자가 필수 정보 중 하나 이상을 입력하지 않은 경우, 시스템은 “모든 정보를 입력해주세요” 메시지를 표시한다.

이미 존재하는 ID를 사용해 회원가입하려고 하는 경우, 시스템은 “이미 가입된 ID입니다.” 메시지를 표시한다.

3) 레시피 추천

사용자는 네비게이션 바에서 “레시피” 버튼을 클릭한다.

시스템은 사용자가 보유 중인 식재료 중 유통기한이 짧고 수량이 많은 식재료를 높은 우선순위로 표시한다.

사용자는 보유 중인 식재료 중 사용하고 싶은 식재료를 선택한다.

시스템은 저장된 레시피 중 사용자가 선택한 식재료를 많이 사용하는 순으로 표시한다.

사용자는 추천된 레시피를 선택하여 상세 정보를 확인할 수 있다.

4) 랜덤 레시피

사용자는 네비게이션 바에서 “랜덤 레시피” 버튼을 클릭한다.

시스템은 서버에 저장된 레시피 중 랜덤하게 10개의 레시피를 표시한다.

사용자는 “랜덤 추천” 버튼을 클릭하여 새로운 10개의 레시피를 랜덤으로 추천받는다.

5) 레시피 검색

3.2.4 ER 다이어그램

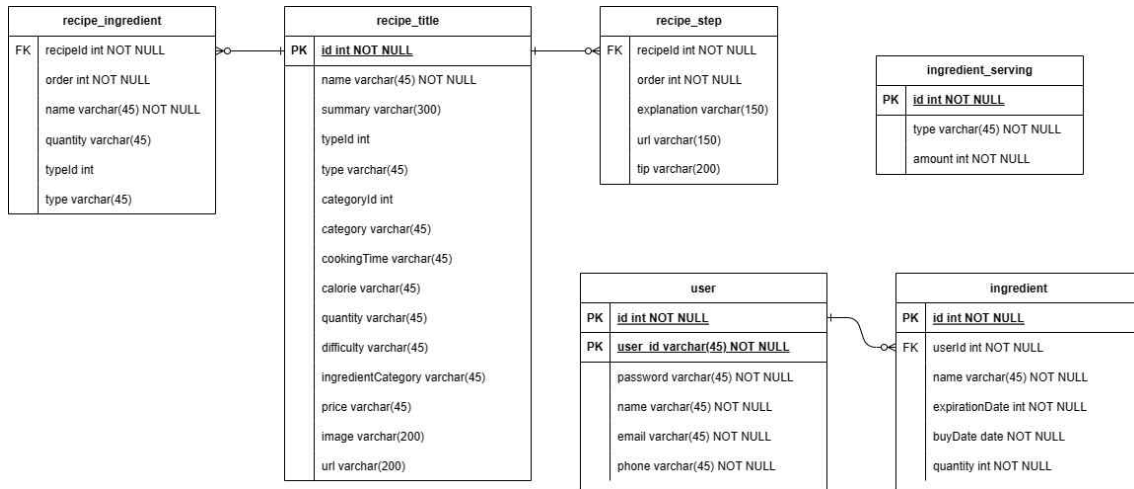


그림 4 ER 다이어그램

3.3 상세설계

3.3.1 식재료 추천 알고리즘

식재료 추천 알고리즘은 사용자에게 우선적으로 사용해야 할 식재료를 추천하기 위해 유통기한과 수량을 고려하여 가중치를 부여한다. 가중치는 아래와 같은 식으로 계산된다

$$\frac{1}{(1 + \text{유통기한})^2} \times \left(\frac{\text{수량}}{1\text{인분}} \right)$$

이로써, 남은 유통기한이 짧을수록, 수량이 많을수록 높은 가중치를 부여하여 사용자에게 효과적인 추천을 제공한다. 자세한 코드는 부록 [1] 가중치 부여 코드에 작성되어 있다.

3.3.2 레시피 식재료 검색 알고리즘

레시피 식재료 검색 알고리즘은 사용자가 선택한 식재료를 활용한 레시피를 찾기 위해 Levenshtein distance 알고리즘을 사용한다. 이 알고리즘은 단어 간의 유사성을 검사하여 사용자가 선택한 식재료와 유사한 레시피를 검색한다. 또한, 검색하는 단어의 앞 글자 중성이 ‘ㄴ’인 경우, 검색 결과에 해당 단어를 추가하는 중성 검사 알고리즘을 구현했다. 자세한 코드는 부록 [3] Levenshtein distance과 [4] 중성 확인에 작성되어 있다.

3.3.3 Thread와 콜백 함수를 이용한 서버 통신

서버와의 효율적인 통신을 위해 Thread를 이용하여 EC2 인스턴스에서 비동기적으로 데이터를 받아오도록 설계하였다. 또한, 서버에서 정보를 불러온 후 UI 변경을 처리하기 위해 콜백 함수를 도입하여 비동기 문제를 효과적으로 해결하였다. 자세한 코드는 부록 [5] 서버 통신에 작성되어 있다.

3.3.4 레시피 추천 알고리즘

레시피 추천 알고리즘은 다음과 같은 절차로 동작한다. 레시피 식재료 검색 결과를 기반으로 레시피 검색 결과의 빈도를 저장하고 처리한다. 동시에, 레시피 검색 작업이 완료된 후 레시피를 추천하기 위해 AtomicInteger 변수를 활용하여 동시성 문제를 효과적으로 해결하였다.

자세한 코드 내용은 부록 [6]인 "레시피 추천"에 작성되어 있으며, 해당 코드는 다수의 식재료에 대한 레시피 검색과 빈도수 계산을 비동기적으로 수행한다. 이를 통해 효율적으로 여러 작업을 동시에 처리하면서도 안정적인 결과를 얻을 수 있다.

4. 프로그램 구현

4.1 시스템 환경

본 프로그램은 안드로이드 기반의 애플리케이션으로 제작되었다. 다음은 시스템 환경에 대한 자세한 설명이다.

플랫폼: Android

하드웨어 구성: 모바일 기기(스마트폰 또는 태블릿)

개발 환경 소프트웨어: Android Studio, Visual Studio, MySQL Workbench

개발 언어: Java, XML, php

데이터베이스: AWS에서 Linux 기반 EC2 인스턴스를 생성하여 해당 서버에 MySQL 설치 및 사용

네트워크 통신: HTTP 통신을 통해 외부 서버와 통신

4.2 시스템 구성

프로그램의 시스템 구성은 다음과 같다.

안드로이드 애플리케이션: 사용자 인터페이스 및 클라이언트 역할을 수행한다. 안드로이드 스튜디오를 사용하여 개발되었으며, 사용자가 식재료를 입력하고 관리할 수 있는 기능을 제공한다.

서버 애플리케이션: 안드로이드 애플리케이션과 데이터베이스 간의 효율적인 통신을 담당한다. PHP를 사용하여 개발되었으며, 안드로이드 애플리케이션으로부터 받은 요청에 따라 데이터베이스에서 필요한 정보를 조회하고 전송한다.

MySQL 데이터베이스: 사용자의 회원 정보, 식재료 정보, 레시피 정보 등을 저장하는 데 사용된다. 안드로이드 애플리케이션 및 서버 애플리케이션과의 상호 작용을 통해 데이터를 관리하고 검색한다.

바코드 인식 API: 안드로이드 애플리케이션에서 바코드를 스캔하여 제품 정보를 가져오는 데 사용된다. 안드로이드에서는 IntentIntegrator와 같은 라이브러리를 사용하여 바코드를 촬영하고 서버에 전송하여 필요한 정보를 받아온다.

AWS EC2 인스턴스: 안드로이드 애플리케이션과 서버 애플리케이션을 호스팅하는 서버 환경으로, 안드로이드 애플리케이션은 EC2에서 실행 중인 서버 애플리케이션

선과 통신하여 데이터를 주고받는다.

4.3 시스템 구현

4.3.1 로그인, 회원가입 기능

로그인 및 회원가입은 안드로이드 애플리케이션에서 서버로 안전한 HTTPS 통신을 통해 구현되었다. 사용자는 안드로이드 애플리케이션의 로그인 화면에서 아이디와 비밀번호를 입력하여 로그인하고, 회원가입 화면에서는 필요한 정보를 입력하여 새로운 계정을 생성할 수 있다. 서버에서는 데이터베이스에 저장된 정보와 일치하는지 확인하고, 새로운 회원 정보를 안전하게 저장한다.

4.3.2 식재료 입력

식재료 입력은 수동 입력과 바코드 입력 두 가지 방법으로 이루어진다. 사용자는 안드로이드 애플리케이션에서 제공하는 식재료 입력 화면에서 직접 이름, 유통기한, 수량, 구매 날짜, 종류를 입력하여 새로운 식재료를 추가할 수 있다. 또한 바코드를 스캔하면 바코드 api를 통해 해당 제품의 정보를 온라인 데이터베이스에서 가져와서 자동으로 입력된다.

4.3.3 보유 식재료 출력

보유 식재료는 유통기한과 수량을 고려하여 우선순위대로 화면에 정렬되어 표시된다. 사용자는 빠르게 소비가 필요한 식재료를 우선적으로 확인할 수 있으며, 각 식재료에 대한 수정 및 삭제 기능을 제공한다.

4.3.4 레시피 검색 및 출력

식재료 기반 레시피 검색은 사용자가 선택한 식재료를 기반으로 최대 10개의 레시피가 출력됩니다. 이를 통해 사용자는 보유한 식재료를 활용한 다양한 레시피를 찾을 수 있다. 각 레시피는 상세 정보를 포함하여 사용자에게 제공되며, 필요한 재료와 조리 방법을 확인할 수 있다.

5. 실험

5.1 시스템 구성

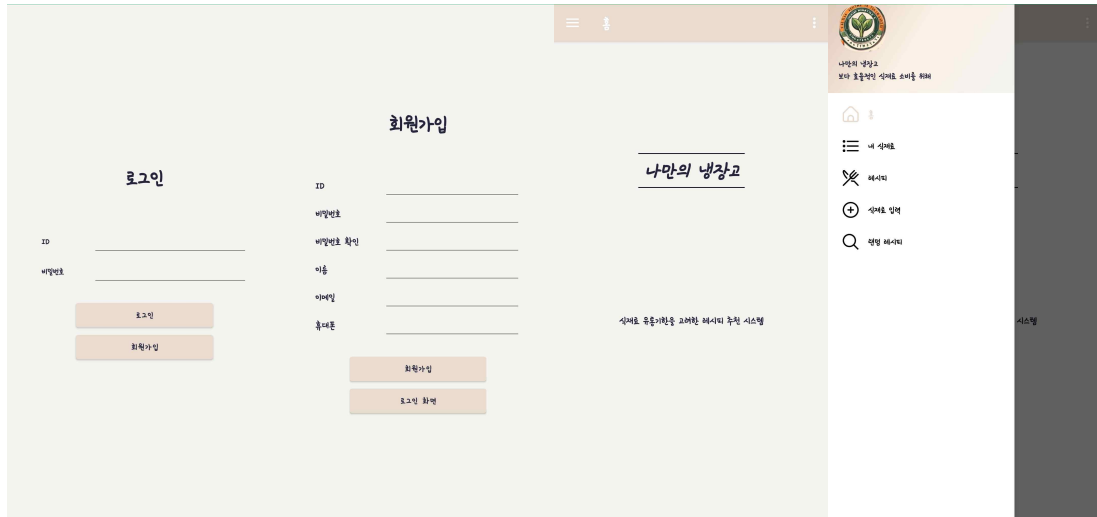


그림 5 로그인 화면 그림 6 회원가입 화면 그림 7 홈 화면 그림 8 네비게이션 바

5.1.1 로그인 및 회원가입 기능

테스트 목표: 사용자가 정상적으로 로그인하고 회원가입을 할 수 있는지 확인

결과: 안드로이드 애플리케이션에서 제공한 로그인 및 회원가입 기능은 성공적으로 동작하였다. 사용자가 입력한 정보가 안전하게 서버로 전송되고, 서버에서는 올바른 응답을 반환하였다.

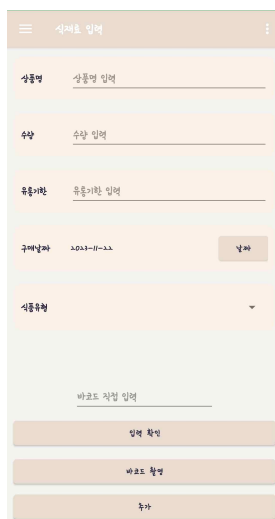


그림 9 식재료 입력
화면

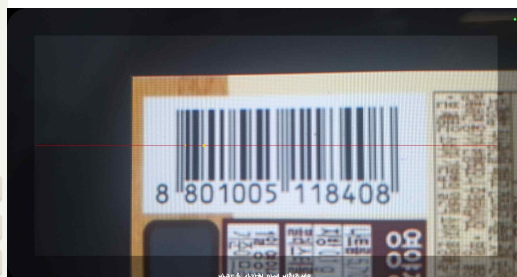
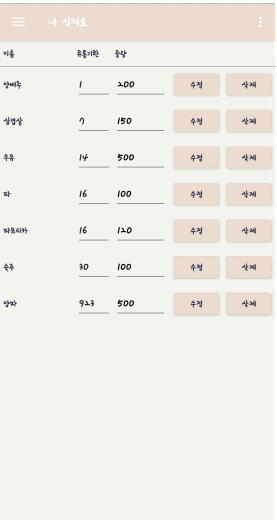


그림 10 바코드 촬영 화면

5.1.2 식재료 입력 기능

테스트 목표: 사용자가 수동 및 바코드를 통한 식재료 입력이 정상적으로 이루어지는지 확인

결과: 사용자가 안드로이드 애플리케이션에서 제공한 식재료 입력 기능을 사용하여 새로운 식재료를 성공적으로 추가할 수 있었다. 바코드를 통한 자동 입력 또한 정확하게 동작하였다.



이름	수량	가격	수정	삭제
양파	1	200	수정	삭제
감	1	150	수정	삭제
우유	14	500	수정	삭제
바나나	16	100	수정	삭제
파인애플	16	120	수정	삭제
귤	30	100	수정	삭제
양파	23	500	수정	삭제

그림 11 보유 식재료

5.1.3 보유 식재료 출력 및 수정/삭제

테스트 목표: 사용자가 보유한 식재료가 정상적으로 출력되고, 수정 및 삭제가 올바르게 이루어지는지 확인

결과: 사용자가 안드로이드 애플리케이션에서 제공한 보유 식재료 관리 기능을 통해 식재료의 우선순위대로 정렬되어 출력되었다. 또한, 사용자가 선택한 식재료의 정보를 수정하거나 삭제하는 기능 또한 정확하게 동작하였다.



그림 12 랜덤 레시피 출력 그림 13 레시피 재료, 순서 그림 14 레시피 검색 화면 그림 15 레시피 검색 결과

5.1.4 랜덤 레시피 출력 및 레시피 검색

테스트 목표: 서버에 저장된 레시피 중 랜덤으로 10개가 출력되고, 사용자가 검색한 식재료가 올바르게 출력되는지 확인

결과: 사용자가 랜덤 레시피 출력 버튼을 클릭할 때마다 서버에 저장된 레시피 중 10개가 랜덤으로 화면에 출력되었다. 또한 사용자가 검색을 하면 사용자가 검색한 레시피를 출력한다

5.2 성능 테스트

5.2.1 서버 통신 성능

테스트 목표: 서버와의 통신이 안정적이며 빠른지 확인

결과: 안드로이드 애플리케이션과 서버 간의 통신은 안정적으로 이루어졌다. Thread와 콜백 함수를 통해 비동기 문제를 해결하여 사용자에게 원활한 화면 전환이 가능하였다.



그림 16 식재료 선택 그림 17 레시피 추천

5.2.2 레시피 추천 성능

테스트 목표: 사용자가 선택한 식재료를 기반으로 레시피를 빠르게 추천하는지 확인

결과: 레시피 추천 기능은 사용자가 선택한 식재료를 기반으로 빠르게 결과를 반환하였다. Levenshtein Distance 알고리즘을 활용하여 단어의 유사성을 고려하여 정확한 검색이 이루어졌다.

6. 결론

이 연구에서 개발한 레시피 추천 시스템은 사용자의 식재료 유통기한과 수량을 고려하여 안드로이드 애플리케이션을 통해 효율적이고 안정적인 레시피를 제공하며, 실험 결과를 통해 이를 확인했다. 이로써 우리는 요리에 필요한 재료를 효율적으로 활용하고 다양한 레시피를 탐색할 수 있는 사용자 친화적인 플랫폼을 구축했다.

향후에는 더 많은 레시피 정보 수집 및 데이터베이스 확장을 통해 레시피의 다양성을 높이고, 빠르고 정교한 레시피 추천 알고리즘을 구현하여 사용자에게 더욱 특화된 추천 서비스를 제공할 것이다. 또한, 사용자 취향을 더욱 정확히 파악하고 반영하기 위한 개인화된 맞춤형 레시피 추천 기능을 강화하여 사용자들이 보다 편리하게 요리를 즐길 수 있도록 발전시켜 나갈 것이다.

참고문헌

- [1] 민성희(Seonghee Min),and 오유수(Yoosoo Oh). "레시피 데이터 기반의 식재료
궁합 분석을 이용한 레시피 추천 시스템 구현." 멀티미디어학회논문지 24.8
(2021): 1114-1121.
- [2] 홍지현(Jiheon Hong),and 이희정(Heejung Lee). "텍스트 분석 기법과 식재료 계
층구조를 활용한 음식 레시피 추천 방법." 대한산업공학회지 45.4 (2019):
302-312.
- [3] 김현지(Kim Hyeonji),and 오유수(Oh Yoosoo). "머신러닝 프로세스의 이미지 및
문자열 데이터 입력에 대한 전처리 방법 추천 시스템 설계." 한국HCI학회 학술
대회 2022.2 (2022): 147-151.
- [4] Yumml. (2023). <https://www.yummly.com/>
- [5] Too Good To Go. (2023). <https://www.toogoodtogo.com/>

부록

[1] 가중치 부여 코드

```
private double calculateWeight(IngredientItem item) {  
    int expiration = item.getExpirationDate(); // 남은 유통기한  
    int quantity = item.getQuantity();         // 남은 수량  
    String buyDate = item.getBuyDate();        // 구매 날짜  
    String type = item.getType();              // 종류  
    int amount = setAmount(type);  
    int daysDifference = calculateDaysDifference(buyDate); // 오늘과 구매 날짜 차이  
    double expirationWeight = calculateExpirationWeight(Math.max(0, expiration +  
daysDifference));  
    return expirationWeight * (quantity / amount);  
}  
  
private double calculateExpirationWeight(int expirationDate) {  
    return 1.0 / Math.pow((1 + expirationDate), 2);  
}
```

[2] 서버 통신을 위한 php 파일

```
$result = $conn->query($query);  
// 결과를 JSON 형태로 변환  
if ($result !== false) {  
    if ($result->num_rows > 0) {  
        $output = array();  
        while($row = $result->fetch_assoc()) {  
            $output[] = $row;  
        }  
        echo json_encode($output);  
    }  
}
```

```

    } else {
        echo json_encode(array("message" => "0 results"));
    }
} else {
    echo json_encode(array("error" => "Query execution error: " . $conn->error));
}

```

[3] Levenshtein Distance

```

public static int levenshteinDistance(String s1, String s2) {
    int[][] dp = new int[s1.length() + 1][s2.length() + 1];

    for (int i = 0; i <= s1.length(); i++) {
        dp[i][0] = i;
    }
    for (int j = 0; j <= s2.length(); j++) {
        dp[0][j] = j;
    }
    for (int i = 1; i <= s1.length(); i++) {
        for (int j = 1; j <= s2.length(); j++) {
            int cost = (s1.charAt(i - 1) == s2.charAt(j - 1)) ? 0 : 1;
            dp[i][j] = Math.min(
                Math.min(dp[i - 1][j] + 1, dp[i][j - 1] + 1),
                dp[i - 1][j - 1] + cost
            );
        }
    }
    return dp[s1.length()][s2.length()];
}

```


[4] 종성 확인

```
public static boolean hasJongseongN(String str, String ingredientName) {  
    int startIndex = str.indexOf(ingredientName);  
    if (startIndex >= 1 && startIndex < str.length()) {  
        char beforeChar = str.charAt(startIndex - 1);  
        if (beforeChar >= 0xAC00) {  
            char jong = (char)((beforeChar - 0xAC00) % 28);  
            return (int)jong == 4;  
        } else if (beforeChar == ' ') {  
            int spaceIndex = startIndex - 2;  
            while (spaceIndex >= 0 && str.charAt(spaceIndex) == ' ') {  
                spaceIndex--;  
            }  
            if (spaceIndex >= 0) {  
                beforeChar = str.charAt(spaceIndex);  
                if (beforeChar >= 0xAC00) {  
                    char jong = (char)((beforeChar - 0xAC00) % 28);  
                    return (int)jong == 4;  
                }  
            }  
        }  
    }  
    return false;  
}
```

[5] 서버 통신

```
protected void getData(OnDataReceivedListener listener, String query, String  
fileName) {  
    new Thread(new Runnable() {
```

```

@Override
public void run() {
    try {
        // URL 및 연결 설정

        // HTTP 요청 전송
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            // 성공적인 응답 처리

            // 데이터 수신 및 전달
            if (listener != null) {
                // 데이터 처리
                listener.onDataReceived(dataField, resultField);
            }
        } else {
            // 에러 응답 처리
        }

        // 연결 종료
        connection.disconnect();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}).start();
}

protected interface OnDataReceivedListener {
    void onDataReceived(String data, int result);
}

```

[6] 레시피 추천

```
public void CalculateFrequency(List<IngredientRecommendItem> selectedItems,
View root) {
    // 동시성 문제 해결
    final AtomicInteger totalTasks = new AtomicInteger(selectedItems.size());
    // 검색 결과 저장 (레시피id, 빈도수)
    Map<Integer, Integer> recipeIdFrequency = new HashMap<>();
    for (IngredientRecommendItem item : selectedItems) {
        String ingredientName = item.getName();
        recipeDB.getRecipeIdsByIngredientName(ingredientName, new
RecipeDB.OnRecipeIdsByIngredientNameListener() {
            @Override
            public void onRecipeIdsByIngredientName(List<Integer> recipeIds) {
                for (int recipeId : recipeIds) {
                    if (recipeIdFrequency.containsKey(recipeId)) {
                        recipeIdFrequency.put(recipeId, recipeIdFrequency.get(recipeId) + 1);
                    } else {
                        recipeIdFrequency.put(recipeId, 1);
                    }
                }
            }
        })
        if (totalTasks.decrementAndGet() == 0) {
            if (!recipeIdFrequency.isEmpty()) {
                recipeIds = getTopRecipeIds(recipeIdFrequency);
                for (int id : recipeIds) {
                    getRecipeTitleById(id);
                }
            } else {
                // 처리할 레시피 ID가 없는 경우
            }
        }
    }
}
```

```

        }
    }
    });
}
}

public List<Integer> getTopRecipeIds(Map<Integer, Integer> recipeIdFrequency)
{
    List<List<Integer>> frequencyGroups = new ArrayList<>();
    List<Integer> topIds = new ArrayList<>();

    // 새로운 맵 생성
    Map<Integer, Integer> modifiedFrequency = new
    HashMap<>(recipeIdFrequency);

    // 빈도수 별로 그룹 나누기
    for (int frequency = Collections.max(modifiedFrequency.values()); frequency >=
1; frequency--) {
        List<Integer> group = new ArrayList<>();
        for (Map.Entry<Integer, Integer> entry : modifiedFrequency.entrySet()) {
            if (entry.getValue() == frequency) {
                group.add(entry.getKey());
            }
        }
        frequencyGroups.add(group);
    }

    // 각 그룹의 멤버들을 랜덤으로 섞기
    Random random = new Random();
    for (List<Integer> group : frequencyGroups) {

```

```

        Collections.shuffle(group, random);
    }

    // 빈도수가 가장 큰 그룹부터 추가
    for (List<Integer> group : frequencyGroups) {
        topIds.addAll(group);
        if (topIds.size() >= 10) {
            break;
        }
    }

    // 레시피 상위 10개 출력
    return topIds.subList(0, Math.min(10, topIds.size()));
}

```