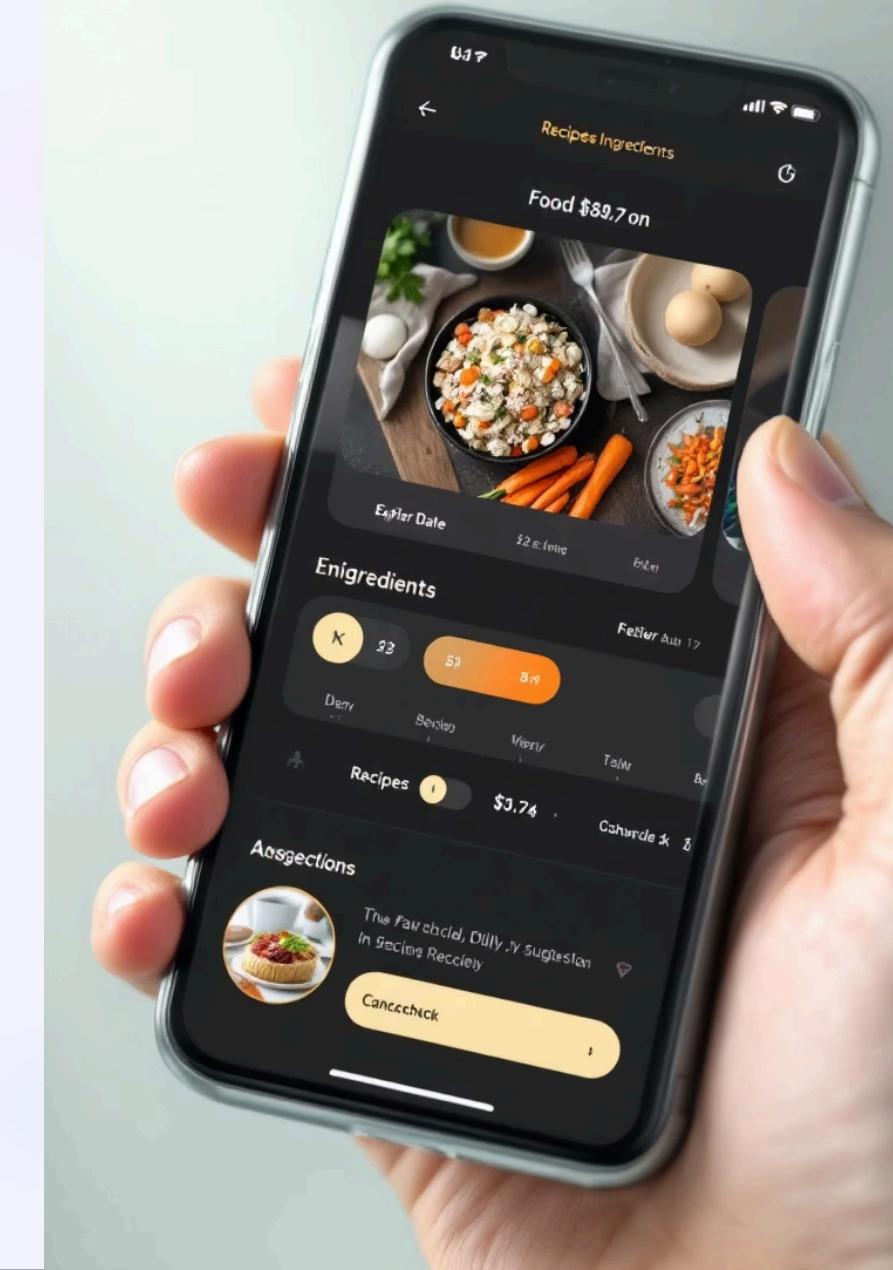


프레시알리미(Fresh-Alimi)

냉장고 속 식재료 유통기한 관리 및 레시피 추천 앱

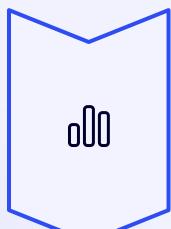
안녕하세요 오늘 저는 식재료 관리를 편하게 해줄 '프레시알리미' 앱의 설계서를 소개해 드리고자 합니다. 현대 사회에서 바쁜 일상을 살아가는 많은 사람들이 냉장고 속 식재료의 유통기한 관리에 어려움을 겪고 있습니다. 앱은 이러한 문제점을 해결하고 식재료 낭비를 줄이는 동시에 맞춤형 레시피를 제공함으로써 스마트한 식생활을 도와드립니다.

2025년 6월, 2021145058 이재용



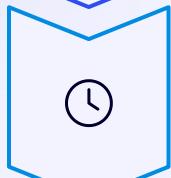


개발 배경 및 필요성



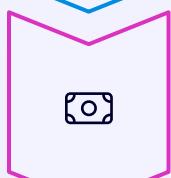
식재료 낭비 증가

한국의 1인당 연간 음식물 쓰레기 발생량은 약 130kg으로 OECD 국가 중 상위권



유통기한 관리의 어려움

바쁜 현대인들은 냉장고 속 식재료의 유통기한을 제대로 파악하지 못함



가계 경제 부담

4인 가족 기준 연간 식재료 낭비로 인한 경제적 손실은 평균 52만원 이상



환경 문제

버려지는 식재료로 인한 탄소 배출량 증가와 환경 오염 심화

이러한 문제들을 해결하기 위해 직관적이고 사용하기 쉬운 솔루션이 필요하다고 판단했습니다.
특히 한국 소비자들의 식재료 구매 및 보관 습관을 고려한 맞춤형 앱 개발이 시급했습니다.

기존 솔루션 분석

수기 관리 방식

냉장고 앞 화이트보드나 메모지를 이용한 전통적 관리 방식

- 일일이 수기로 작성해야 하는 번거로움
- 메모 분실 또는 관리 소홀 가능성
- 실시간 알림 기능 부재

기존 앱 솔루션

Pantry Check, Grocy 등 해외 중심의 식재료 관리 앱

- 한국어 지원 및 한국 식품 DB 부족
- 수동 입력에 의존하는 불편함
- 단순 관리 기능에 집중, 활용 제안 부족

사용자 요구사항

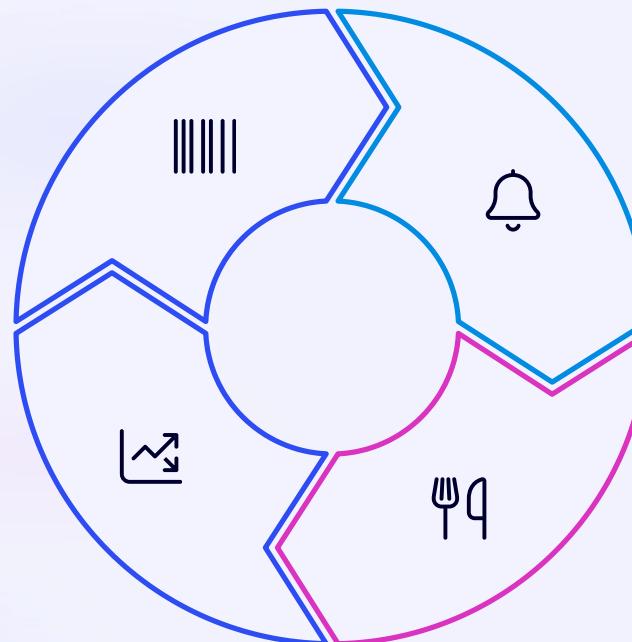
설문조사 결과 도출된 핵심 요구사항

- 간편한 등록 프로세스(87% 응답)
- 자동 알림 기능(92% 응답)
- 실질적인 활용 방안 제시(76% 응답)

기존 솔루션들은 공통적으로 사용자 편의성과 자동화 측면에서 한계를 보이고 있으며, 특히 한국 소비자의 식생활 패턴과 식품 구매 환경에 최적화되지 않은 문제점이 있습니다.

프레시알리미 핵심 기능

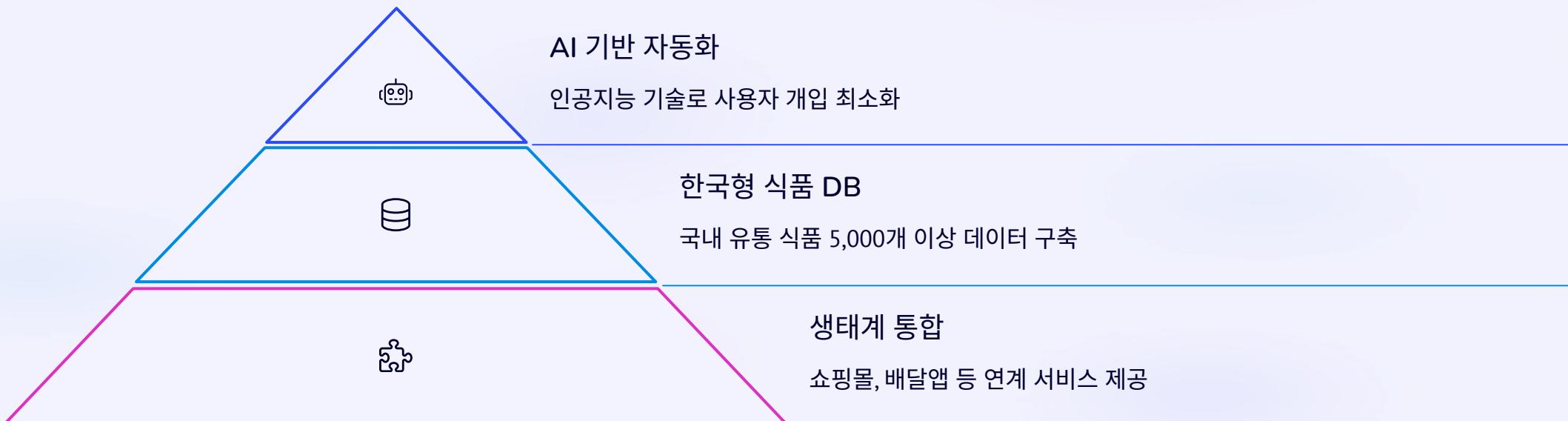
- 스마트 바코드 스캔**
식품 바코드 스캔으로 상품명, 제조일, 유통기한 자동 인식 및 저장
- 소비 패턴 분석**
사용자의 식재료 소비 패턴을 분석하여
최적의 구매량 제안



- 맞춤형 알림 시스템**
유통기한 임박 식재료에 대한 단계별 알림 설정 가능
- 지능형 레시피 추천**
냉장고 속 재료와 유통기한을 고려한 최적의 레시피 추천

프레시알리미는 단순한 유통기한 관리를 넘어 사용자의 식생활 전반을 개선하는 통합 솔루션을 제공합니다. 특히 한국 소비자들이 자주 구매하는 5,000개 이상의 식품 데이터베이스를 구축하여 정확한 정보 제공에 중점을 두고 있습니다.

차별화 전략



프레시알리미는 기존 앱들과 달리 한국인의 식생활과 구매 패턴에 최적화된 서비스를 제공합니다. 특히 자동화 수준을 높여 사용자가 최소한의 노력으로 식재료를 관리할 수 있도록 설계되었습니다. 국내 주요 마트와 편의점에서 판매되는 식품들의 바코드 정보를 데이터베이스화하여 인식률을 95% 이상으로 높였습니다.

또한 단순 관리를 넘어 식재료를 효율적으로 활용할 수 있는 다양한 레시피와 요리 팁을 제공하며, 향후 식품 쇼핑몰 및 배달 서비스와의 연계를 통해 통합 식생활 플랫폼으로 발전할 계획입니다.

앱 인터페이스 디자인



프레시알리는 직관적이고 사용하기 쉬운 인터페이스를 제공합니다. 메인 화면에서는 유통기한이 임박한 식재료를 한눈에 확인할 수 있으며, 하단 내비게이션을 통해 주요 기능에 빠르게 접근할 수 있습니다.

바코드 스캔 화면은 자동 초점 기능을 통해 빠르게 식품을 인식하며, 냉장고 관리 화면에서는 식재료를 카테고리별로 구분하여 체계적으로 관리할 수 있습니다. 또한 레시피 추천 화면에서는 보유한 식재료로 만들 수 있는 다양한 요리와 상세한 조리법을 제공합니다.

인터페이스 주요 기능 상세



유통기한 알림 시스템

식품별 맞춤 알림 설정 기능을 통해 유통기한 3일 전, 1일 전, 당일 등 다양한 시점에 푸시 알림을 받을 수 있습니다. 중요도에 따라 알림 강도 조절도 가능합니다.



냉장고 관리 시스템

식재료를 냉장실/냉동실별, 카테고리별로 분류하여 체계적으로 관리합니다. 보관 위치 지정 및 식재료 사용량 추적 기능으로 효율적인 재고 관리가 가능합니다.



식품 소비 분석 대시보드

월별/주별 식품 소비 패턴과 폐기율을 시각화하여 제공합니다. 식품 구매 최적화 제안과 낭비 감소를 위한 맞춤 조언을 받을 수 있습니다.



스마트 바코드 스캔

AI 기반 바코드 인식 기술로 0.5초 내 제품 정보를 자동 스캔합니다. 바코드가 없는 식품은 사진 촬영만으로도 식품명과 유통기한 정보를 자동 입력할 수 있습니다.



맞춤형 레시피 추천

보유한 식재료를 기반으로 만들 수 있는 레시피를 AI가 자동 추천 합니다. 개인 식습관, 알레르기 정보, 조리 난이도 등을 고려한 맞춤형 추천 시스템을 제공합니다.



가족 공유 기능

가족 구성원들과 냉장고 정보를 실시간으로 공유할 수 있습니다. 식품 구매 목록 작성 및 공동 편집 기능으로 중복 구매를 방지하고 효율적인 식단 계획이 가능합니다.

User Interface



Business Logic



Data Storage



Database
Server

기술 스택

프론트엔드

- Flutter - 크로스 플랫폼 개발
- Dart - 주요 프로그래밍 언어
- Provider - 상태 관리
- GetX - 라우팅 및 종속성 주입
- 안드로이드 스튜디오

백엔드

- Firebase - 실시간 데이터베이스
- Node.js - 서버 사이드 로직
- MongoDB - 레시피 데이터 저장
- Google Cloud Functions - 서비스 기능

핵심 기술

- Google ML Kit - 바코드 인식
- TensorFlow Lite - 식품 이미지 인식
- Firebase Cloud Messaging - 푸시 알림
- Recommendation API - 레시피 추천

프레시알리미는 빠른 개발 주기와 안정적인 성능을 위해 Flutter 프레임워크를 채택했습니다. 이를 통해 iOS 플랫폼에서 일관된 사용자 경험을 제공할 수 있습니다. 백엔드는 확장성과 실시간 데이터 처리를 위해 Firebase와 Node.js를 조합하여 구축했으며, 대용량 레시피 데이터는 MongoDB를 통해 효율적으로 관리합니다.

주요 코드 구현

이 섹션에서는 프로젝트에서 주요한 기능을 구현하는 코드를 살펴보겠습니다. 각 항목은 해당 기능의 주요 특징과 구현 방법에 대한 간략한 설명을 포함합니다.

- 회원 가입 및 로그인**: 사용자에게 회원 가입과 로그인 기능을 제공합니다. 이 기능은 보통 사용자 데이터베이스와 인증 서버를 통해 구현됩니다.
- 제품 목록 표시**: 제품 목록을 표시하는 API를 구현합니다. 이는 제품 데이터베이스와 연동되는 코드로, 제품의 이름, 가격, 재고 수 등을 포함합니다.
- 제품 상세 정보**: 선택한 제품의 상세 정보를 표시하는 API를 구현합니다. 이는 제품 데이터베이스와 연동되는 코드로, 제품의 설명, 이미지, 리뷰 등을 포함합니다.
- 장바구니 관리**: 사용자가 장바구니에 제품을 추가하거나 삭제하는 기능을 구현합니다. 이는 장바구니 데이터베이스와 연동되는 코드로, 제품의 수량과 총 가격을 업데이트합니다.
- 결제 처리**: 사용자가 결제를 완료하는 기능을 구현합니다. 이는 결제 서비스와 연동되는 코드로, 결제 정보를 확인하고 결제 상태를 업데이트합니다.
- 리뷰 및 평가**: 사용자가 제품에 대한 리뷰 및 평점을 남기는 기능을 구현합니다. 이는 사용자 평가 데이터베이스와 연동되는 코드로, 평가 내용과 점수를 저장합니다.
- 검색 기능**: 사용자가 제품을 검색할 수 있는 기능을 구현합니다. 이는 검색 엔진과 연동되는 코드로, 검색어에 맞는 결과를 반환합니다.
- 로그 기록**: 사용자의 행동과 시스템 상태를 기록하는 기능을 구현합니다. 이는 로그 파일과 연동되는 코드로, 문제 발생 시 문제 원인을 추적하는 데 도움입니다.

Firebase 실시간 데이터베이스 연동

```
import Firebase

func addFoodItem(_ item: [String: Any]) {
    guard let userId = Auth.auth().currentUser?.uid else { return }
    let ref = Database.database().reference()
    let itemRef = ref.child("users").child(userId).child("food_items").childByAutoId()

    var newItem = item
    newItem["id"] = itemRef.key

    itemRef.setValue(newItem)
}
```

현재 로그인된 사용자의 Realtime Database 경로(/users/사용자ID/food_items/랜덤ID)에

식품 데이터를 저장하고, 자동 생성된 ID를 항목에 함께 저장합니다.

푸시 알림 구현

```
import FirebaseMessaging
import UserNotifications

class ExpiryNotificationService: NSObject, MessagingDelegate {
    func messaging(_ messaging: Messaging, didReceive remoteMessage: MessagingRemoteMessage) {
        let data = remoteMessage.appData
        if let title = data["title"] as? String,
           let msg = data["message"] as? String,
           let id = data["foodId"] as? String {
            let content = UNMutableNotificationContent()
            content.title = title
            content.body = msg
            UNUserNotificationCenter.current().add(
                UNNotificationRequest(identifier: id, content: content, trigger: nil),
                completionHandler: nil
            )
        }
    }
}
```

FCM(Firebase Cloud Messaging)에서 받은 푸시 메시지를 받아

제목과 내용을 추출해 즉시 iOS 알림으로 표시하는 기능입니다.

식품 추천 알고리즘

```
struct FoodItem {
    let name: String
    let expiryDate: Date
}

struct Recipe {
    let id: String
    let ingredients: [String]
    var matchScore: Int
}

class RecipeRecommender {
    func recommend(foodItems: [FoodItem], snapshot: DataSnapshot) -> [Recipe] {
        let limit = Calendar.current.date(byAdding: .day, value: 3, to: Date())!
        let keywords = foodItems.filter { $0.expiryDate <= limit }
            .map { $0.name.lowercased() }

        var results: [Recipe] = []

        for case let snap as DataSnapshot in snapshot.children {
            guard let data = snap.value as? [String: Any],
                  let ing = data["ingredients"] as? [String] else { continue }

            let match = ing.filter { keywords.contains($0.lowercased()) }.count
            if match >= 2 {
                results.append(Recipe(id: snap.key, ingredients: ing, matchScore: match))
            }
        }
        return results.sorted { $0.matchScore > $1.matchScore }
    }
}
```

유통기한이 3일 이내로 임박한 식품명을 기준으로, 해당 재료를 2개 이상 포함한 레시피만 추려서 추천합니다.

일치한 재료 수를 기준으로 내림차순 정렬하여 사용자에게 가장 잘 맞는 레시피를 우선 제공합니다.

식품 소비 통계 분석

```
class FoodAnalytics {
    let db = Firestore.firestore()
    let userId: String

    init(userId: String) {
        self.userId = userId
    }

    // 최근 1개월 폐기율 계산 (간단히 작성)
    func calcWasteRate() async throws -> Double {
        let oneMonthAgo = Calendar.current.date(byAdding: .month, value: -1, to: Date())!
        let historyRef = db.collection("users").document(userId).collection("food_history")

        async let disposedCount = historyRef
            .whereField("disposalDate", isGreaterThanOrEqualTo: oneMonthAgo)
            .getDocuments()

        async let consumedCount = historyRef
            .whereField("consumedDate", isGreaterThanOrEqualTo: oneMonthAgo)
            .getDocuments()

        let (disposed, consumed) = try await (disposedCount, consumedCount)

        let total = disposed.count + consumed.count
        return total > 0 ? Double(disposed.count) / Double(total) * 100 : 0.0
    }

    // 평가를 가장 자주 폐기된 카테고리
    func topWasteCategories() async throws -> [String] {
        let docs = try await db.collection("users").document(userId)
            .collection("food_history")
            .whereField("status", isEqualTo: "disposed")
            .getDocuments()

        var count: [String: Int] = [:]
        for doc in docs.documents {
            let cat = doc.get("category") as? String ?? ""
            count[cat, default: 0] += 1
        }

        return count.sorted { $0.value > $1.value }.map { $0.key }
    }
}
```

- **목적:** 최근 한 달 동안의 식품 폐기율(%)을 계산
- **방식:**
- 한 달 전 날짜 기준으로 소비 및 폐기항목 조회
- 폐기 + 소비 총 수를 기준으로 폐기 비율 계산

이러한 코드를 통해 사용자는 식품 관리를 효율적으로 할 수 있으며, 음식물 쓰레기 감소와 가계 지출 절약 효과를 기대할 수 있습니다.

가족 공유 기능 개요(코드는 ppt 4장화)

사용자가 가족 구성원과 식품 정보를 공유할 수 있는 기능을 구현합니다. 이를 통해 여러 사용자가 동일한 냉장고 내용을 확인하고 관리할 수 있습니다.

기능 목적

- 여러 사용자가 **동일한 냉장고 식품 정보를 공유 가능**
- 가족 구성원을 초대하고, 초대 수락 시 자동으로 동기화

가족 그룹 생성

```
func createFamilyGroup(groupName: String) async throws -> String {
    let groupRef = db.collection("family_groups").document()
    let data: [String: Any] = [
        "name": groupName,
        "createdBy": userId,
        "createdAt": Timestamp(),
        "members": [userId]
    ]
    try await groupRef.setData(data)
    try await db.collection("users").document(userId).updateData([
        "familyGroupId": groupRef.documentID
    ])
    return groupRef.documentID
}
```

사용자 본인이 그룹 생성 후, `users` 컬렉션에 그룹 ID를 저장

가족 구성원 초대

```
func inviteFamilyMember(groupId: String, email: String) async throws {
    let inviteCode = generateInviteCode()
    let data: [String: Any] = [
        "groupId": groupId,
        "invitedBy": userId,
        "email": email,
        "code": inviteCode,
        "createdAt": Timestamp(),
        "status": "pending"
    ]
    try await db.collection("invitations").addDocument(data: data)
}
```

생성된 초대 코드를 이메일로 전송 (실제 앱에선 FCM, SMTP 연동 가능)

초대 수락

```
func acceptInvitation(inviteCode: String) async throws -> String {
    let snapshot = try await db.collection("invitations")
        .whereField("code", isEqualTo: inviteCode)
        .whereField("status", isEqualTo: "pending")
        .getDocuments()

    guard let doc = snapshot.documents.first else {
        throw NSError(domain: "InvalidInvite", code: 404)
    }

    let groupId = doc.get("groupId") as! String
    try await db.collection("family_groups").document(groupId)
        .updateData(["members": FieldValue.arrayUnion([userId])])
    try await db.collection("users").document(userId)
        .updateData(["familyGroupId": groupId])
    try await doc.reference.updateData(["status": "accepted"])

    return groupId
}
```

초대 코드가 유효하면, 사용자도 가족 그룹에 자동 추가

가족 식품 정보 동기화

```
func syncFamilyFoodItems() async throws -> [[String: Any]] {
    let groupId = try await getGroupId()
    let members = try await getGroupMembers(groupId: groupId)

    var allFoodItems: [[String: Any]] = []

    for memberId in members {
        let docs = try await db.collection("users").document(memberId)
            .collection("food_items").getDocuments()
        for doc in docs.documents {
            var item = doc.data()
            item["ownerId"] = memberId
            allFoodItems.append(item)
        }
    }

    return allFoodItems
}
```

가족 구성원 모두의 식품 정보를 모아서 하나의 냉장고처럼 보여줍니다.

```
import MLKit
import UIKit

func scanBarcode(from image: UIImage) {
    let visionImage = VisionImage(image: image)
    visionImage.orientation = .up // 이미지 방향 설정

    let scanner = BarcodeScanner.barcodeScanner()

    scanner.process(visionImage) { barcodes, error in
        guard error == nil, let barcodes = barcodes else { return }

        for barcode in barcodes {
            if let value = barcode.rawValue {
                fetchProductInfo(value)
            }
        }
    }
}
```

사용자가 찍은 이미지를 MLKit으로 분석해 바코드를 인식하고,

추출된 값을 통해 제품 정보를 조회하는 구조입니다.

바코드 인식 기능 설명 (슬라이드용 부연설명)

바코드 인식 기능은 스마트폰의 **카메라와 연동**되어야만 사용할 수 있는 기능입니다.
이를 구현하기 위해선 몇 가지 필수 설정이 필요합니다.

먼저, **MLKit의 BarcodeScanning 라이브러리를** 프로젝트에 설치해야 하며,
Info.plist 파일에 카메라 접근 권한 요청 문구를 추가해야 합니다.

그리고 사용자가 카메라로 찍은 이미지를 받아,
이를 MLKit의 **VisionImage** 객체로 변환한 뒤 분석합니다.
분석된 바코드 값(**rawValue**)을 통해 제품 정보를 조회하는 방식입니다.
반드시 실기기(iPhone)에서 카메라 권한을 허용한 상태에서만 동작합니다.

이러한 구조를 통해 사용자가 냉장고 속 식품의 바코드를 스캔하면,
앱은 해당 바코드를 자동 인식하여 제품 정보를 빠르게 불러올 수 있게 됩니다.



기대 효과 및 사회적 가치

30%

음식물 쓰레기 감소

가구당 식재료 낭비 감소율 목표치

₩520K

연간 비용 절감

4인 가족 기준 식비 절약 예상액

45%

요리 빈도 증가

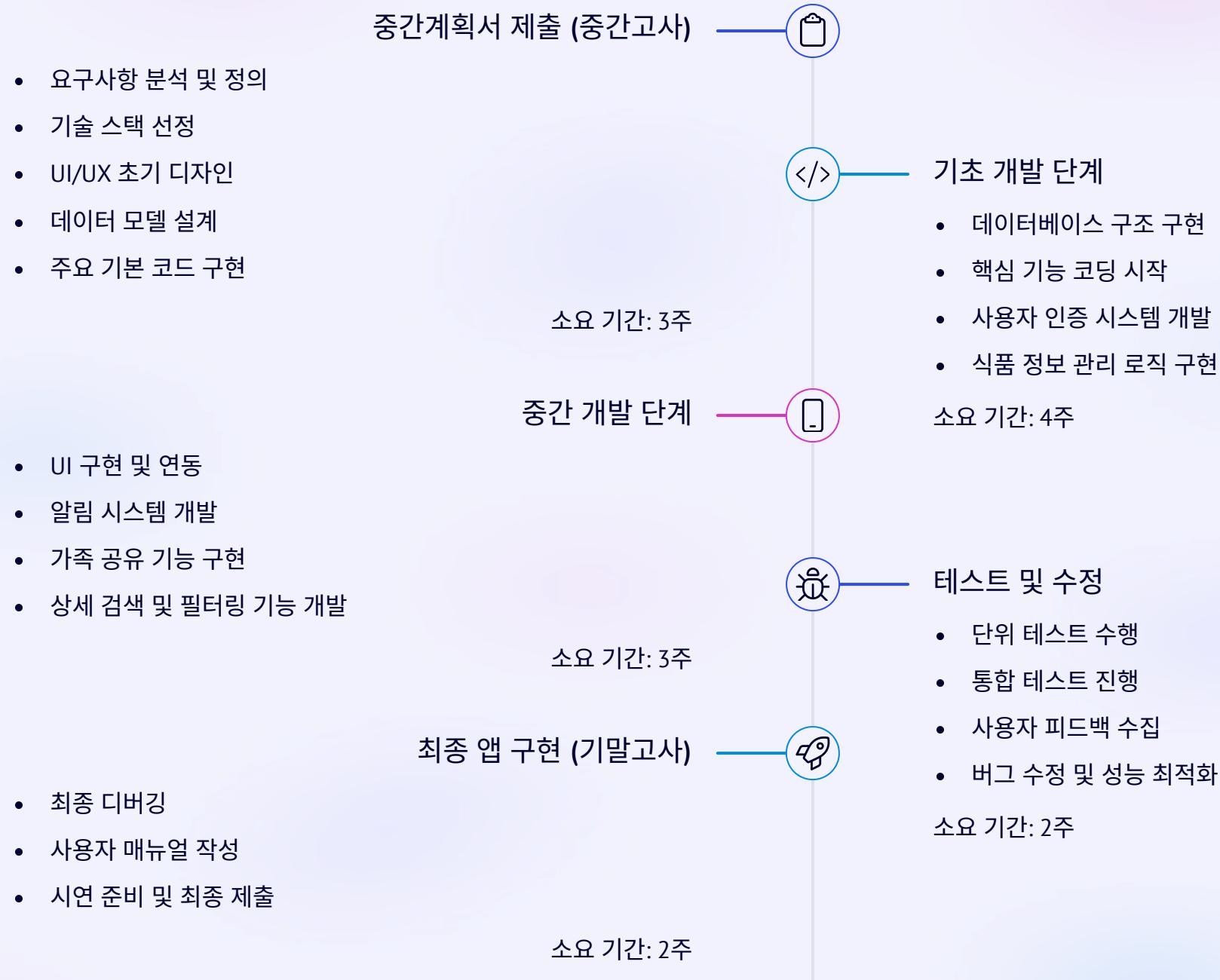
집에서 직접 요리하는 빈도 증가율

프레시알리미는 개인의 식생활 관리 개선을 넘어 사회적으로도 의미 있는 가치를 창출합니다. 음식물 쓰레기 감소를 통한 환경 보호, 가계 경제 부담 완화, 그리고 건강한 식습관 형성에 기여할 것으로 기대됩니다. 특히 1인 가구와 맞벌이 가정에서 효율적인 식재료 관리를 통해 균형 잡힌 식단을 유지하는 데 도움이 될 것입니다.

또한 지역 농산물 소비 촉진 기능을 추가하여 지역 경제 활성화에도 기여할 계획이며, 사용자들의 데이터를 분석해 한국인의 식생활 패턴 연구에 도움이 될 수 있는 익명화된 데이터를 제공할 예정입니다.

프레시알리미 개발 과정정

중간계획서 제출부터 최종 앱 구현까지의 단계별 일정입니다.



총 개발 기간: 14주 (한 학기)



향후 발전 방향

- 1 1단계: 런칭(2025)
기본 기능 구현 및 바코드 스캔, 유통기한 관리, 레시피 추천 기능 출시
- 2 2단계: 고도화(2026)
AI 맞춤형 레시피 추천 및 온라인 쇼핑몰 연계 구매 서비스 도입
- 3 3단계: 확장(2027)
스마트 냉장고 연동 시스템과 사용자 레시피 공유 플랫폼 구축
- 4 4단계: 생태계 구축(2028)
식품 정기 배송 서비스와 영양사 상담 연계 종합 식생활 플랫폼으로 확장

프레시알리미는 단계적 기능 확장을 통해 IoT 기기 연동 스마트 주방 생태계의 중심 앱으로 자리매김하며, 지속적인 데이터 축적과 AI 알고리즘 개선으로 정확하고 맞춤화된 서비스를 제공할 것입니다.

메인 코드 완성본 캡쳐화면 (화면은 안드로이드 버전) - (IOS 앱개발 최종 미완성)

The screenshot shows the Android Studio interface with a Flutter project named "my_app". The code editor displays the main.dart file, which contains the following code:

```
// main.dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
  runApp(const FreshAlimiApp());
}

class FreshAlimiApp extends StatelessWidget {
  const FreshAlimiApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: '프레쉬 알리미',
      theme: ThemeData(primarySwatch: Colors.deepPurple),
      home: const MainTabController(),
    ); // MaterialApp
  }
}

class FoodItem {
  final String name;
  final DateTime expiryDate;
  final String storageType;
  final String category;
  int usageCount;

  FoodItem({
    required this.name,
    required this.expiryDate,
    required this.storageType,
    required this.category,
    this.usageCount = 0,
  });

  int daysLeft() {
    return expiryDate.difference(DateTime.now()).inDays;
  }

  Map<String, dynamic> toJson() => {
    'name': name,
    'expiry': expiryDate.toIso8601String(),
    'storage': storageType,
    'category': category,
    'usage': usageCount,
  };
}

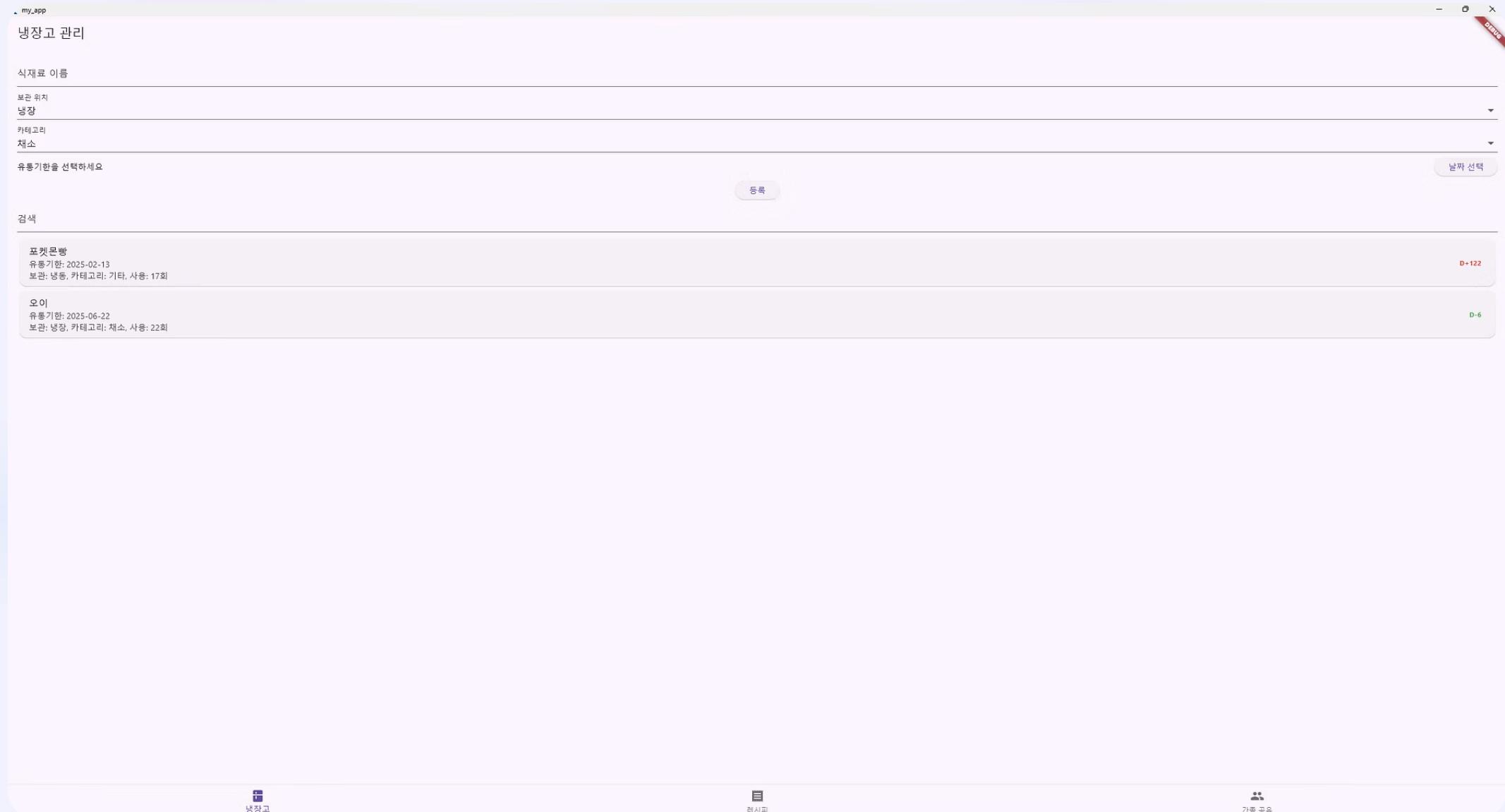
factory FoodItem.fromJson(Map<String, dynamic> json) => FoodItem(
  name: json['name'],
  expiryDate: DateTime.parse(json['expiry']),
  storageType: json['storage'],
  category: json['category'],
  usageCount: json['usage'],
);

class MainTabController extends StatefulWidget {
  const MainTabController({super.key});

  @override
  State<MainTabController> createState() => _MainTabControllerState();
}
```

The project structure on the left shows various build outputs (apk, flutter-apk, native-assets, etc.) and platform-specific files (ios, lib, android). The bottom status bar indicates the code is at line 30, column 18, with 2 spaces, UTF-8 encoding, and CRLF line endings.

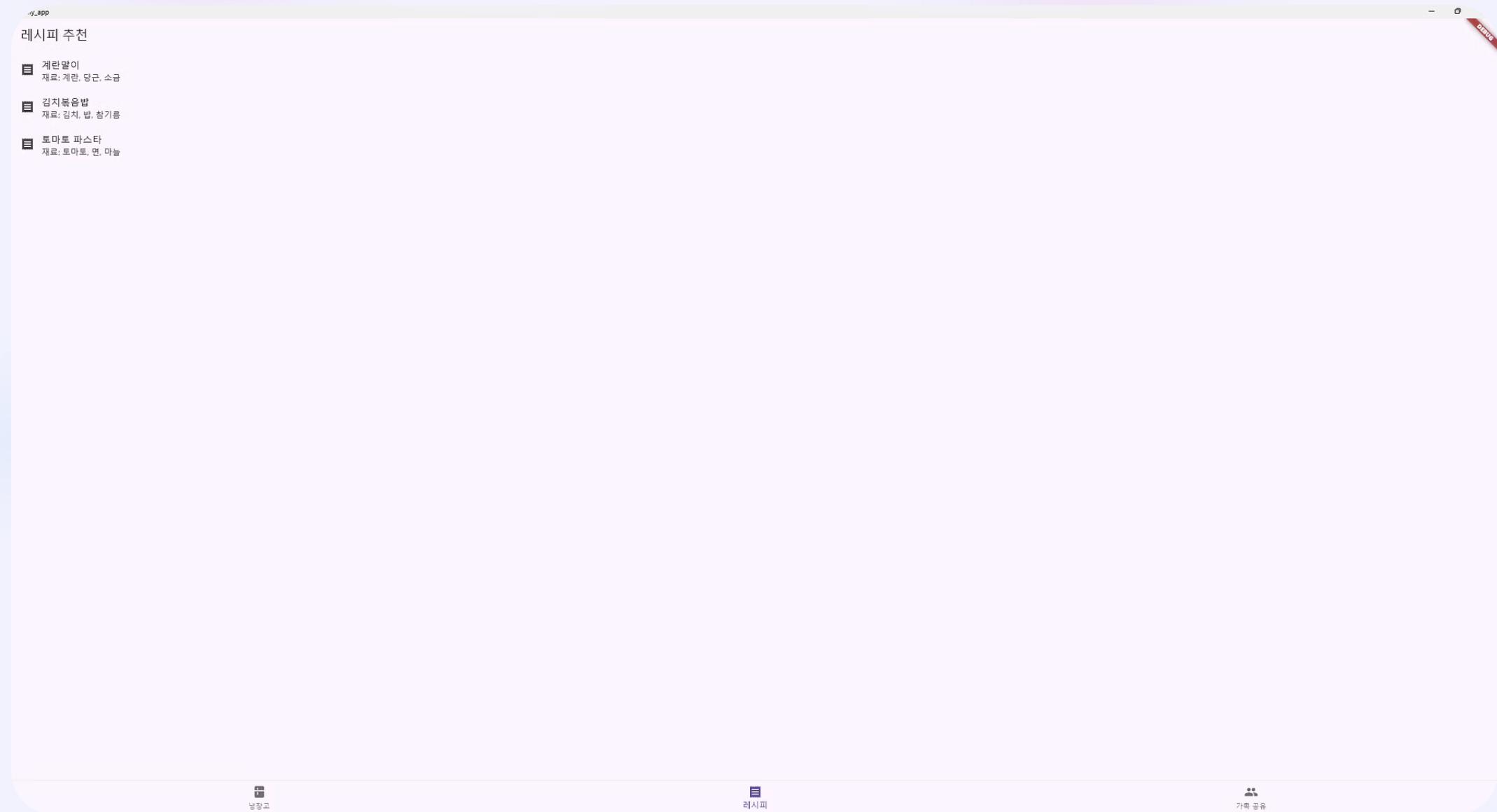
메인 화면 : 냉장고 관리



식재료 이름을 입력하고 냉장/냉동 분류하고 여러가지 카테고리 중 해당 카테고리를 체크하고 날짜를 선택한 다음에 등록 버튼을 누르면 하단에 보이시는 것처럼 제품이 추가되며, 날짜가 몇일 남았는지 보여주며 날짜가 지났으면 빨강색, 날짜가 얼마 안남았으면 (3일 이하) 노랑색, 날짜가 4일 이상 넉넉히 있으면 초록색으로 또는 것을 확인 할 수 있습니다.

그리고 화면을 나갔다 들어와도 그대로 유지되어 메모리 업데이트에도 신경을 썼습니다. 그리고 해당 검색창에 검색하면 음식도 단번에 찾을수 있습니다. 등록하는 음식 개수 제한은 없습니다.

레시피 추천 화면



매일 매일 업데이트 되는 레시피 추천화면입니다.

남은 식재료 위주로 업데이트 됩니다.

가족 공유 기능



가족 공유 기능입니다. 가족이나 친구 지인이 해당 앱을 깔면 공유기능을 사용할 수 있습니다.

등록된 식재료를 확인할 수 있으며, 개수도 확인 할 수 있습니다.

분류를 깔끔하게 할 수 있는 장점이 있습니다.

감사합니다.