프로젝트 계획서

팀원	이동제, 윤현서, 최현빈, 홍성호, 홍창희

제출일	2024.10.08	**팀장**	홍성호
과목	소프트웨어공학	**담당교수**	김상철

1. 프로젝트 개요

1.1 프로젝트 개요

프로젝트 명: 식단관리를 위한 AI를 이용한 성분 분석 어플리케이션

목표: 사용자가 섭취하는 음식의 칼로리 및 영양소 정보를 효율적으로 기록하고, 목표에 맞춰 식단을 관리할 수 있도록 지원하는 어플리케이션을 개발한다. AI 기반 음식 이미지 인식을 통해 음식의 칼로리와 영양 정보를 자동으로 분석하고, 데이터베이스에 기록하여 사용자 맞춤형 피드백을 제공한다.

필요성: 현재 많은 사람들이 다이어트, 건강 관리, 근육 증가 등의 목표를 위해 섭취한 음식의 칼로리와 영양 성분을 정확하게 파악할 필요가 있지만, 이를 수동으로 기록하는 것은 번거롭고 정확성도 떨어질 수 있다. 이러한 문제를 해결하기 위해 본 어플리케이션은 사용자가 편리하게 식단을 영양 성분 및 칼로리로 관리할 수 있도록 돕는다.

1.2 프로젝트의 산출물

AI 기반 식단 관리 어플리케이션: Android 및 iOS 플랫폼에서 사용할 수 있는 어플리케이션.

데이터베이스 설계 문서: 음식의 칼로리 및 영양소 정보를 저장하고 관리하는 데이터베이스 구조.

사용자 매뉴얼: 사용자가 어플리케이션을 쉽게 사용할 수 있도록 돕는 매뉴얼 문서.

테스트 보고서: 성능 시험 및 각종 기능 테스트 결과를 문서화한 보고서.

1.3 정의 약어

AI: Artificial Intelligence (인공지능)

DB: Database (데이터베이스)

API: Application Programming Interface(애플리케이션 프로그래밍 인터페이스)

2. 자원 및 일정 예측

2.1 자원

- 인력

프론트엔드 개발자 (1명): 사용자가 어플리케이션에서 직접 이용하는 화면과 인터페이스를 개발한다.

백엔드 개발자 (1명): 서버와 데이터베이스를 관리하고, AI 이미지 인식 기능을 통합한다.

AI 엔지니어 (1명): 음식 이미지 분석을 위한 AI 모델을 개발하고, 이를 어플리케이션에 적용한다.

데이터 분석가 (1명): 음식 영양 정보 데이터를 분석하고 AI 학습을 위한 데이터를 준비한다.

UI/UX 디자이너 (1명): 어플리케이션의 사용자 경험을 최적화하고, 사용자 친화적인 디자인을 구현한다.

- 비용

a. 개발 도구: 클라우드 서버 비용 (AWS, 약 100만 원), 데이터베이스 비용, AI API 사용료 (Google Vision API 또는 자체 개발 API). 데이터셋은 Food-101 이용.

b. 테스트 장비: 다양한 스마트폰 (iOS/Android)에서의 성능 테스트를 위한 기기 대여 비용.

c. 인건비(Man Month 계산):

1) 프로젝트 기간: 6개월

2) 필요한 인력: 5명

3) 인력 1명의 월 인건비: 200만원

프로젝트 전체 인건비를 계산하기 위해서는, MM을 기준으로 다음과 같이 계산한다.

Man Month = 프로젝트 기간 x 개발자 수

Man Month = 6개월 x 5명 = 30Man Months

이제 인건비는 각 개발자의 월 인건비와 곱해서 구할 수 있다.

총 인건비 = 30Man Months x 200만원 = 6천만원

따라서 이 프로젝트에서 소요되는 총 인건비는 6천만원이 된다.

d. 기타: 마케팅, 배포 및 유지 관리에 필요한 비용.

2.2 일정

1주차: 프로젝트 요구사항 수집, 분석 및 문서화

2~4주차: 아키텍쳐 설계, 데이터베이스 설계 및 UI/UX 디자인

4~6주차: 음식 이미지 분석 및 데이터 전처리 AI 모델 구축, FE-BE, AI-DB 통합설계

7~11주차: 프론트엔드 기능 구현 및 백엔드 개발

11~13주차: AI모델 구현, 통합 및 연동

13~17주차: 기능 테스트 및 시스템 통합 테스트

17~21주차: 사용자 수용 테스트

21~24주차: 성능 및 보안 최적화

주차	**작업 내용**
1주차	프로젝트 요구사항 수집, 분석 및 문서화
2~4주차	아키텍쳐 설계, 데이터베이스 설계 및 UI/UX 디자인
4~6주차	음식 이미지 분석 및 데이터 전처리 AI 모델 구축, FE-BE, AI-DB 통합설계
7~11주차	프론트엔드 기능 구현 및 백엔드 개발
11~13주차	AI모델 구현, 통합 및 연동
13~17주차	기능 테스트 및 시스템 통합 테스트
17~21주차	사용자 수용 테스트
21~24주차	성능 및 보안 최적화

3. 조직 구성 및 인력 배치

3.1 조직 구성

프로젝트 매니저: 전반적인 프로젝트 일정 관리 및 팀원 간의 조율을 담당.

프론트엔드 개발자: 사용자 인터페이스와 사용자 경험을 최적화하여 어플리케이션의 UI를 구축.

백엔드 개발자: 서버, 데이터베이스 설계 및 관리, AI 분석 결과를 서버에 연동.

AI 엔지니어: 음식 사진 인식 및 분석을 위한 AI 모델 설계 및 구현.

데이터 분석가: 영양 정보 데이터 수집 및 AI 학습용 데이터 준비.

UI/UX 디자이너: 사용자 친화적인 화면 설계 및 사용성 개선.

3.2 직무 기술

프론트엔드 개발자: React Native를 이용하여 다양한 기기에서 동작할 수 있는 모바일 UI를 개발.

백엔드 개발자: Node.js와 Express를 사용해 서버 구축 및 데이터 연동. MongoDB를 사용하여 사용자 데이터를 저장 및 관리.

AI 엔지니어: TensorFlow 및 OpenCV를 사용하여 음식 이미지 인식 모델을 개발하고 성능을 최적화.

데이터 분석가: 영양소 데이터 분석 및 AI 모델 학습을 위한 데이터 준비.

UI/UX 디자이너: Figma를 사용해 어플리케이션 화면 설계 및 사용자 편의성 개선 작업.

4. WBS (Work Breakdown Structure)

(1단계) 기획 단계:기획 및 요구사항 정의 (1주차)

- 프로젝트 요구사항 수집
- 요구사항분석 및 문서화
- 기술적 요구사항 정의

(2단계) 설계 단계: 데이터베이스 및 UI/UX 설계 (2~6주차)

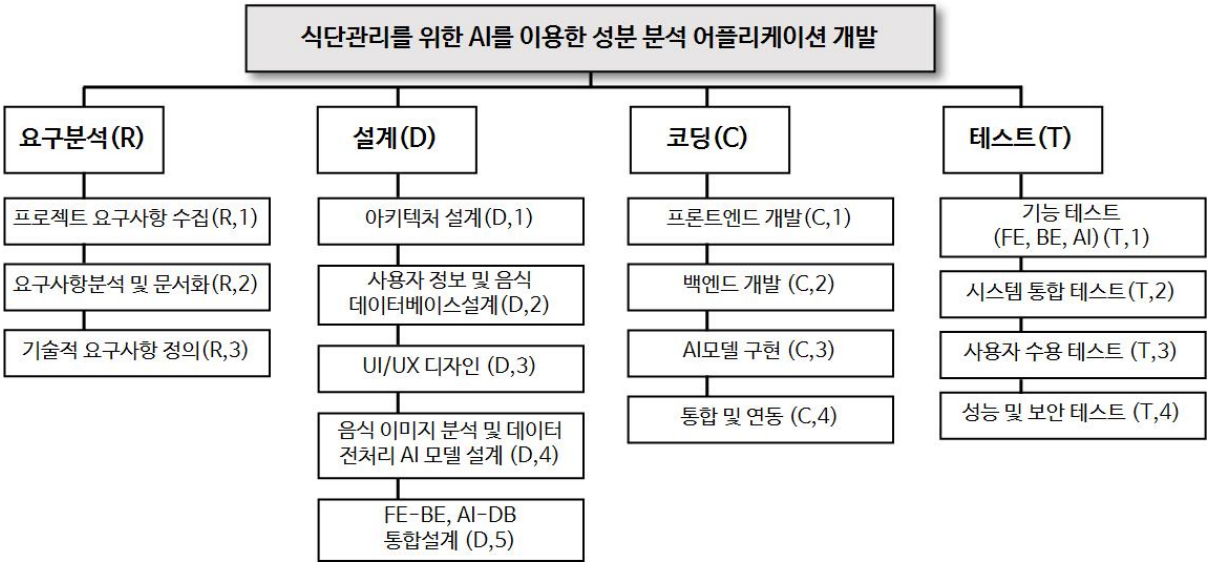
- 아키텍처 설계
- 사용자 정보 및 음식 데이터베이스 설계
- UI/UX 디자인
- 음식 이미지 분석 및 데이터 전처리 AI 모델 설계
- FE-BE, AI-DB 통합설계

(3단계) 개발(코딩) 단계: 프론트엔드 및 백엔드, AI 모델 개발 (7~13주차)

- 프론트엔드 개발
- 백엔드 개발
- AI모델 구현
- 통합 및 연동

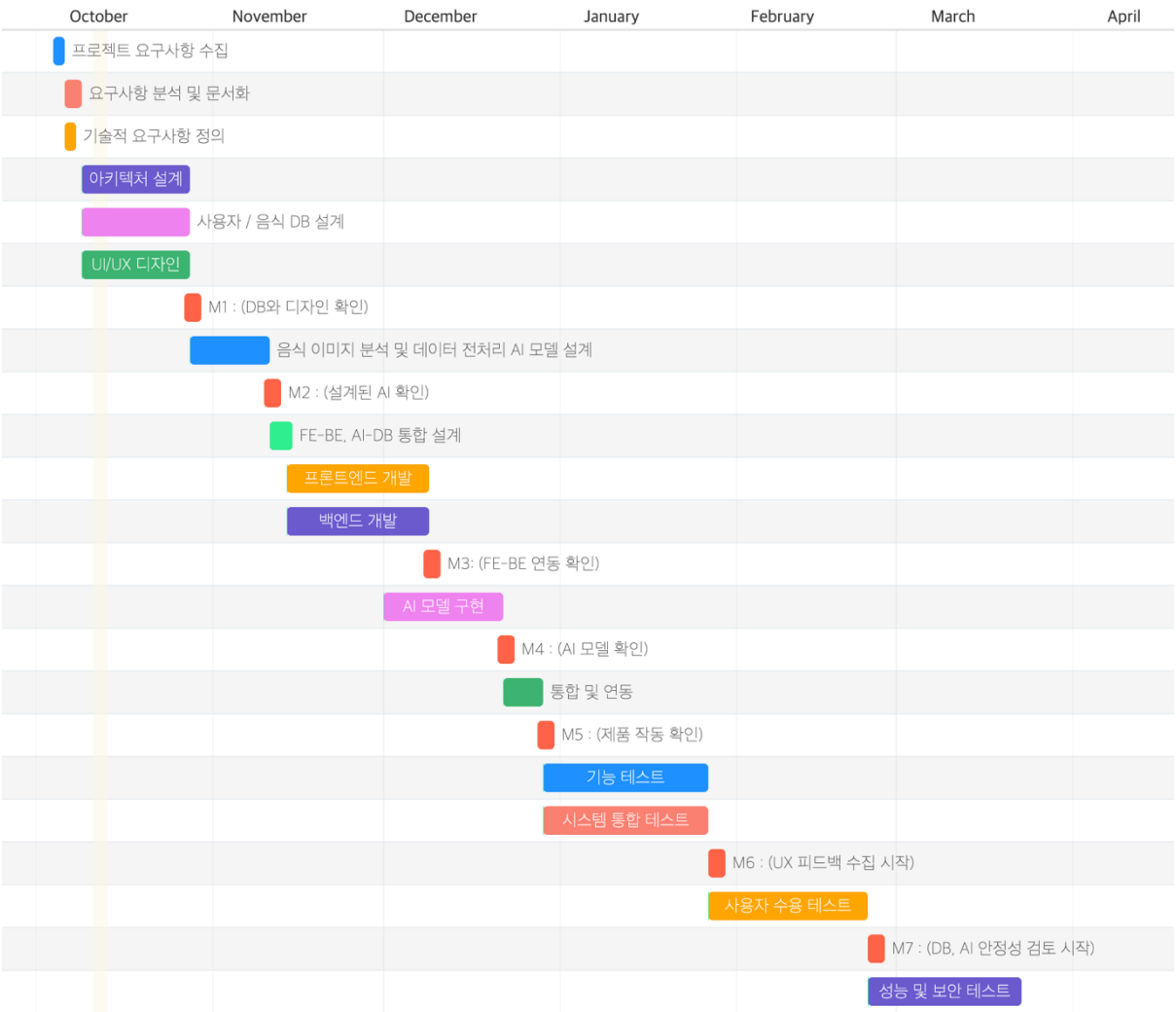
(4단계) 테스트 단계: 성능 시험 및 최적화 (13~24주차)

- 기능 테스트(FE, BE, AI)
- 시스템 통합 테스트
- 사용자 수용 테스트
- 성능 및 보안 테스트



4-2 각 단계별 일정

	A	B	C	D	E	F	G
1	ID	Activities	Dependency	Start Date	Scheduled End I	Completion	Type
2	Task 0	프로젝트 요구사항 수집		2024. 10. 4	2024. 10. 5		Main Task
3	Task 1	요구사항 분석 및 문서화		2024. 10. 6	2024. 10. 8		Main Task
4	Task 2	기술적 요구사항 정의		2024. 10. 6	2024. 10. 7		Main Task
5	Task 3	아키텍처 설계		2024. 10. 9	2024. 10. 27		Main Task
6	Task 4	사용자 / 음식 DB 설계		2024. 10. 9	2024. 10. 27		Main Task
7	Task 5	UI/UX 디자인		2024. 10. 9	2024. 10. 27		Main Task
8	Task 6	M1 : (DB와 디자인 확인)		2024. 10. 27	2024. 10. 29	0	Milestone
9	Task 6	음식 이미지 분석 및 데이터 전처리 AI		2024. 10. 28	2024. 11. 10		Main Task
10	Task 17	M2 : (설계된 AI 확인)		2024. 11. 10	2024. 11. 12	0	Milestone
11	Task 7	FE-BE, AI-DB 통합 설계		2024. 11. 11	2024. 11. 14	1	Main Task
12	Task 8	프론트엔드 개발		2024. 11. 14	2024. 12. 8		Main Task
13	Task 9	백엔드 개발		2024. 11. 14	2024. 12. 8		Main Task
14	Task 18	M3: (FE-BE 연동 확인)		2024. 12. 8	2024. 12. 10	0	Milestone
15	Task 10	AI 모델 구현		2024. 12. 1	2024. 12. 21		Main Task
16	Task 19	M4 : (AI 모델 확인)		2024. 12. 21	2024. 12. 23	0	Milestone
17	Task 11	통합 및 연동		2024. 12. 22	2024. 12. 28		Main Task
18	Task 20	M5 : (제품 작동 확인)		2024. 12. 28	2024. 12. 30	0	Milestone
19	Task 12	기능 테스트		2024. 12. 29	2025. 1. 26		Main Task
20	Task 13	시스템 통합 테스트		2024. 12. 29	2025. 1. 26		Main Task
21	Task 21	M6 : (UX 피드백 수집 시작)		2025. 1. 27	2025. 1. 29		Milestone
22	Task 14	사용자 수용 테스트		2025. 1. 27	2025. 2. 23		Main Task
23	Task 22	M7 : (DB, AI 안정성 검토 시작)		2025. 2. 24	2025. 2. 26		Milestone
24	Task 15	성능 및 보안 테스트		2025. 2. 24	2025. 3. 22		Main Task



M1	**DB 와 디자인 확인**	**(10.27)**
M2	**설계된 AI 확인**	**(11.10)**
M3	**FE-BE 연동 확인**	**(12.8)**
M4	**AI 모델 확인**	**(12.21)**
M5	**제품 작동 확인**	**(12.28)**
M6	**UX 피드백 수집 시작**	**(1.27)**
M7	**DB, AI 안정성 검토 시작**	**(2.24)**

5. 기술 관리 방법

5.1 변경 관리

요구사항 변경은 프로젝트 매니저 승인 후 문서화한다.

Git으로 버전관리를 하는데, 모든 소스 코드 및 문서 변경 사항은 GitHUb를 통해 버전관리 되고 팀원들과 공유된다. GitFlow 전략을 사용하여 branch(버전)는 main과 develop으로 나누어 관리하며, 새로운 기능이나 수정사항은 각각 별도의 feature, fix 브랜치에서 작업한 후 병합한다.

5.2 위험 관리

주요 위험 요소: AI 모델 성능 저하, 예상보다 긴 개발 시간

대응 방안: 일정에 버퍼를 두고 AI 모델을 조정하는 단계 마련, 주마다 일정 조정 및 성능 문제 발견 시 즉각적인 해결방안 마련.

5.3 비용 및 진도 관리

매주 진행 상황을 보고하며, 예산 초과 시 즉시 대응 방안을 논의한다.

5.4 문제점 해결 방안

주요 문제 발생 시 즉각적으로 팀 회의를 소집하여 해결 방안을 논의한다.

6. 표준 및 개발 절차

6.1 개발 방법론

폭포수 방법론을 사용하여 개발을 진행하되, 각 단계의 완료 후 피드백을 반영하여 다음 단계에 유연하게 접근. 프로젝트의 진행 상황에 따라 필요한 수정 및 개선을 지속적으로 반영.

8. 개발 환경

프론트엔드: React Native를 이용해 크로스 플랫폼 모바일 어플리케이션 개발.

백엔드: Node.js, Express를 사용하여 API 서버 구축.

데이터베이스: MongoDB를 사용해 음식 영양 정보 및 사용자 데이터를 관리.

AI 개발 환경: TensorFlow와 OpenCV를 사용하여 음식 인식 모델 개발.

배포 환경: AWS EC2, S3를 사용해 서버 및 데이터베이스 호스팅.

9. 성능 시험 방법

9.1 성능 항목

1. 메모리 사용량

- 목표: 애플리케이션 실행 중 메모리 사용량을 일정 수준 이하로 유지 (예: 100MB 이하).

- 측정 방법:

- 서버나 디바이스에서 애플리케이션이 실행될 때의 메모리 소비량을 지속적으로 모니터링.

- Android의 경우 Android Studio의 메모리 프로파일러, iOS의 경우 Xcode Instruments로 메모리 사용량을 측정.

- 서버 쪽에서는 Linux 기반 시스템의 top 명령어나 htop을 사용해 메모리 사용률을 확인.

2. CPU 사용률

- 목표: AI 모델 분석이나 데이터 처리 시 CPU 사용률을 70% 이하로 유지.

- 측정 방법:

- 서버 또는 클라이언트 측에서 애플리케이션이 구동될 때 CPU 점유율을 측정.

- 서버 쪽에서는 top, htop, 또는 클라우드 환경의 모니터링 도구(AWS CloudWatch, GCP Stackdriver)를 사용해 CPU 사용률을 모니터링.

3. 네트워크 대역폭 사용량

- 목표: 네트워크 대역폭을 최소화하여 1MB 이하로 이미지 데이터를 전송.

- 측정 방법:

- 이미지 업로드와 AI 결과 전송 시 패킷의 크기를 측정하여 네트워크 대역폭 사용량을 확인.

- Wireshark 같은 네트워크 트래픽 분석 도구를 사용해 전송 데이터의 크기를 분석하거나, 클라우드 서비스의 네트워크 트래픽 로그를 확인.

4. 데이터베이스 응답 시간

- 목표: 데이터베이스의 데이터 조회 및 기록 시 응답 시간을 100ms 이하로 유지.

- 측정 방법:

- 쿼리 실행 시간 측정 도구를 사용하여 각 데이터베이스 쿼리의 실행 시간을 측정.

- MongoDB의 경우 explain() 명령어를 사용하여 쿼리 성능을 평가할 수 있으며, SQL 기반 데이터베이스는 각종 로그 및 성능 분석 도구를 사용할 수 있음.

5. AI 모델 정확도

- 목표: 음식 이미지 인식 모델의 정확도를 90% 이상 유지.

- 측정 방법:

- 테스트 데이터셋을 이용해 AI 모델이 음식을 정확하게 인식하는지를 측정.

- Precision, Recall, F1-Score 같은 성능 지표를 사용해 모델의 성능을 평가.

- 각 음식 카테고리 별로 인식 성공률을 평가하고, 잘못된 인식에 대한 오류 분석 진행.

6. 응용 프로그램 안정성

- 목표: 애플리케이션이 24시간 이상 연속 실행될 때도 안정적으로 작동 (Crash-Free Rate 99.9%).

- 측정 방법:

- Crashlytics나 Sentry와 같은 로그 수집 및 오류 보고 도구를 사용해 애플리케이션의 충돌 빈도를 모니터링.

- 장시간 실행 테스트를 통해 메모리 누수 또는 다른 문제로 인한 비정상 종료를 확인.

7. 로드 밸런싱 성능

- 목표: 동시 접속자가 많아질 때 로드 밸런서를 통해 각 서버의 부하를 균등하게 분배.

- 측정 방법:

- 클라우드 환경에서 로드 밸런싱이 제대로 동작하는지, 서버 간 부하가 균등하게 나뉘어지는지 확인.

- AWS ELB(Elastic Load Balancer)나 GCP의 Load Balancer 로그를 통해 각 서버의 트래픽 분포를 확인.

10. 문서화

모든 개발 및 변경 사항은 GitHub를 통해 관리되며, 각 단계의 산출물은 문서화되어 팀원 간에 공유된다.