

15팀 GRAD 캡스톤 그로스 프로젝트 2차 보고서

팀장: 이재인(2076325)
팀원: 나지원(2176120)

1. 팀 정보

팀 정보			
프로젝트 명	휴대폰 사용 제어에 어려움을 겪는 사용자들을 위한 AI 분석 기반 커뮤니티 게임형 관리 시스템		
팀 명	15 팀 GRAD	팀 구성원	이재인 (207625), 나지원 (2176120)

2. 과제 요약

2.1 문제 정의 및 제안 내용
<p>1. 문제 정의</p> <p>최근 많은 사람들이 스마트폰의 과도한 사용을 제어하기 위해 다양한 앱사용 제어 앱을 이용하여 자신의 앱 사용량을 관리하는 경우가 많다. 실제로 스마트폰이 보급되기 시작한 20여년 전과 현재를 비교해보면 과거엔 상상조차 못할 정도로 스마트폰에 대한 의존도가 급격하게 올라간 것을 확인할 수 있다. 이처럼 일상생활에 영향을 끼칠 정도로 휴대폰에 의존하는 경향은 증가하고 있는데, 이는 조사 결과에서도 찾아볼 수 있다. <휴대폰 중독 척도의 개발 및 타당화>라는 논문에 따르면, 중국의 대학생들을 조사한 결과, 절반이 넘는 학생들이 휴대폰에만 하루에 6시간 이상의 시간을 할애하고 있다고 한다.</p> <p>위와 같은 상황을 바탕으로 Pain Point를 살펴보면 다음과 같다.</p> <ol style="list-style-type: none">1) 휴대폰 사용 조절을 위해 앱 사용 조절을 돕는 앱을 사용하더라도, 강제적인 제제로 사용자의 반발감과 부정적인 감정을 유발하여 앱 사용 제어 효과가 약해지는 경우가 있음.2) 앱 사용 제어에 직접적인 도움을 주는 것이 아닌, 사용자의 앱 사용 로그를 기록하는 정도에 그치는 앱이 많음.3) 이미 시장에 존재하는 휴대폰 사용 제어 앱의 자율성이 부족하여 사용자마다 원하는 방식으로 앱 사용을 조절하기에 한계가 존재함.4) 개인의 노력으로 위와 같은 문제점을 해결하려 하는 경우가 많아 휴대폰 사용 제어에 대한 목표 및 동기가 약화될 가능성이 존재함. <p>2. 제안 내용</p> <p>현대 사회의 사회적인 문제점으로 떠오르고 있는 휴대폰 사용 조절의 어려움을 정리하여 작성한 Pain Point를 바탕으로 해결책을 작성하였다.</p>

- 1) 강제성을 약화하기 위해 자기주도적으로 달성하고자 하는 목표를 설정하여 변화를 추구함.
- 2) AI를 활용하여 사용자의 앱 사용 습관을 분석하고, 분석한 로그를 바탕으로 효과적인 제어를 위해 추가 개선 요소를 사용자에게 추천함.
- 3) 사용자의 앱 사용 목적에 따라 조절하고자 하는 앱의 조절 강도를 원하는 만큼 설정함.
- 4) 사용자와 유사한 목표를 가진 유저들과 소통할 수 있는 커뮤니티 기능을 통해 동기 부여를 강화함.

다음과 같은 해결책을 바탕으로 앱 사용을 단순히 금지하는 등의 통제, 권한 제한과 같은 수동적인 제한보다는, 더 효과적인 사용량 제어와 사용자의 능동적인 행동 변화를 이끌어내기 위하여 스스로 달성하고자 하는 목표를 설정하고, AI를 통해 자신의 앱 사용 습관을 돌아보며, 목표를 달성하는 중에도 다른 사용자와 끊임없이 교류하며 동기부여를 얻을 수 있는 앱을 만들하고자 한다. 또한 실제로 현재 게임의 형식을 빌려 사용자의 능력 계발과 긍정적인 습관 변화를 이끌어내는 데 도움을 준 <Habitica>와 <듀오링고> 어플을 주요 레퍼런스로 삼아 기획을 진행하여, 목표 달성 과정에 게임적 요소를 넣어 강제성을 띠지 않도록 하고 효과적인 사용량 기록 및 휴대폰 사용 제어에 동기부여가 될 수 있는 앱을 구상했다.

2.2 기존 연구와 비교와 기대 효과 및 의의

1. 기존 연구와 비교

설계한 프로젝트를 바탕으로 시중에 출시되어 있는 유사한 기능을 가진 앱과 비교하면 다음과 같다.

1. **Habitica** (게임형식의 업무 관리 어플)
‘RPG’ 게임과 유사하게 유저가 원하는 직업을 선택하고, 자신이 원하는 목표를 설정하여 해당 목표를 달성했을 시 경험치를 얻는 형식의 업무 관리 어플이다.

기존 어플의 특징과 불편한 점	프로젝트의 특징과 차별되는 점
설정한 목표의 달성 여부를 사용자가 스스로 체크한다.	자신이 원하는 목표만 설정하면 자동으로 휴대폰 사용 로그를 기록하고 달성 여부를 체크한다.
목표 달성 여부를 체크하는 것에 있어 객관성이 약화되며, 직접 체크해야 한다는 점에서 불편함이 발생한다.	목표 달성 과정을 확실하게 체크할 수 있고, 측정치를 기준으로 하여 목표 달성 여부를 자동으로 판단하기에 객관성과 편의성이 강화된다.

2. **Forest: Focus for Productivity** (사용자의 폰 사용 시간을 바탕으로 한 그래픽 상호작용 어플)
사용자의 휴대폰 사용 시간의 감소 정도와 비례하여 어플 내의 ‘나무’를 키워나가는 기록 형식의 휴대폰 사용 시간 관리 어플이다.

기존 어플의 특징과 불편한 점	프로젝트의 특징과 차별되는 점
------------------	------------------

‘휴대폰 사용 여부’에 따라 스톱워치가 작동하여 스크린 타임을 기록한다.	개별 사용자에게 따라 원하는 앱의 사용 여부를 설정하고, 앱 별로 사용 기록을 분석한다.
휴대폰의 사용 여부만 판단할 수 있으며, 앱 별로 따로 관리할 수 없어 세세한 제어가 불가능하다.	단순히 사용 시간을 기록하는 것이 아닌 맞춤형 목표 설정이 가능하며, 활용성이 높은 추가 기능을 제공한다.

2. 기대 효과 및 의의

이 어플은 사용자가 스스로 설정한 목표를 달성하도록 돕고, 목표 달성 여부에 따라 포인트를 제공하는 보상 시스템을 통해 재미와 성취감을 부여한다. 또한 자신의 사용 습관을 객관적으로 확인하고 개선할 수 있도록 맞춤형 피드백과 추천을 제공하며, 이를 통해 능동적인 사용 조절이 가능하다. 더 나아가 커뮤니티와 랭킹 기능을 통해 같은 목표를 가진 사용자들 간의 소통과 경쟁을 계속하며 유대감과 동기부여가 강화된다. 이러한 방식은 단순한 사용량 측정을 넘어 자율성과 즐거움을 기반으로 한 건강한 스마트폰 사용 문화를 형성하는 데 의의가 있다.

2.3 주요 기능 리스트

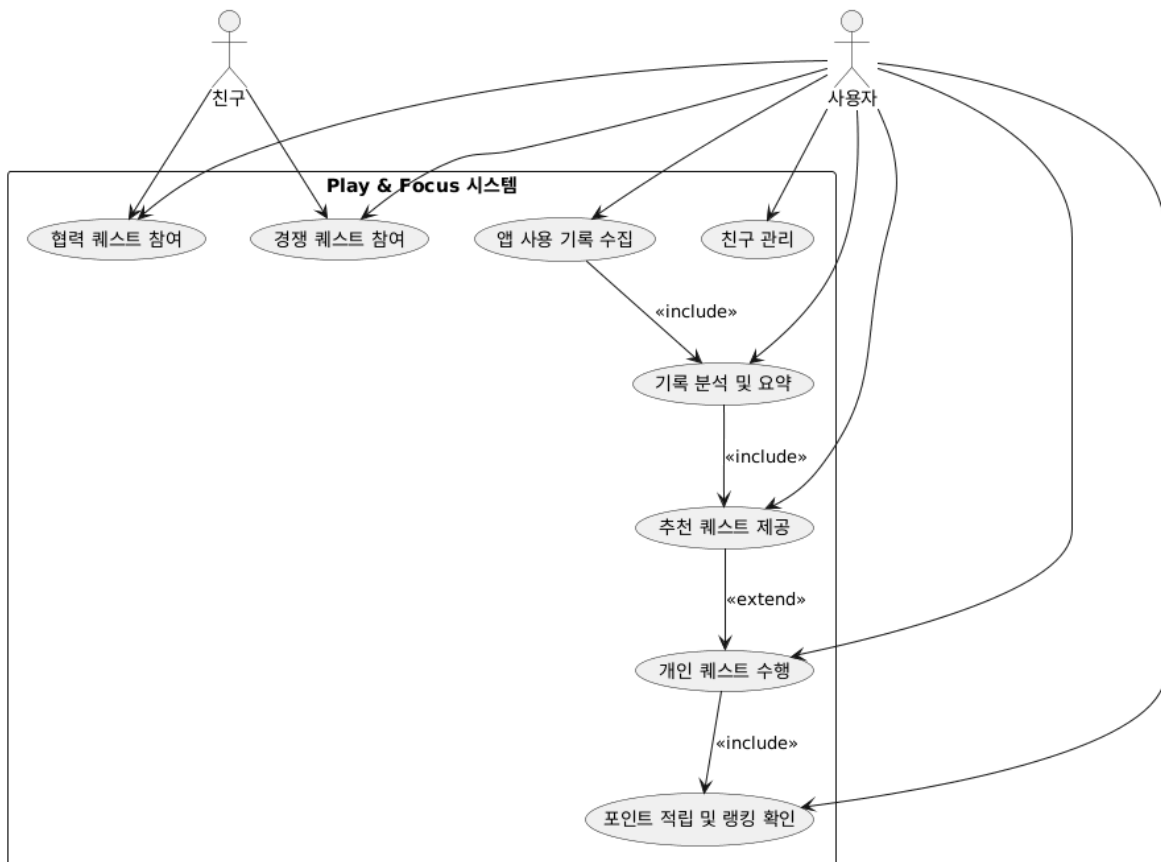
사용자의 앱 사용 관리 프로젝트의 주요 기능을 정리하면 다음과 같다.

주요 기능	기능 세부 서술
사용자 맞춤 목표 설정	원하는 앱을 직접 선택하여 제어하고자 하는 사용 시간을 설정한다. 설정해둔 시간 동안 실시간으로 목표 달성 여부 확인 가능하며, 자신의 목표 달성 여부를 통해 스스로 앱 사용 제어의 강도를 조절할 수 있다.
분석 & 요약 피드백	AI가 하루 동안 기록된 사용자의 앱 사용 로그를 기반으로 패턴을 분석하고, 한 줄 요약 메시지를 제공한다. 또한 어플 사용 시간 및 분석을 카테고리 별로 분류하여 제공한다.
포인트 & 랭킹 시스템	사용자가 설정한 목표를 달성할 시 포인트를 지급한다. 누적 포인트 기반으로 다른 사용자와의 순위를 제공함으로써 경쟁 시스템을 제공한다.
퀘스트 기반 동기부여	사용자의 주 관심 관리 앱의 카테고리에 따라 매일 맞춤형의 새로운 퀘스트 (개인, 합동, 경쟁)를 제안한다. 해당 퀘스트를 완료할 시 보상을 제공하고, 실패할 시 관련 피드백 제공으로 꾸준한 동기부여를 제공한다.
커뮤니티 기능	커뮤니티 기능을 통해 다른 유저들과 퀘스트 결과, 성과등을 공유할 수 있다. 서로 응원 메시지 주고받으며 유사한 목표를 달성하는 공동체적 경험을 제공한다.

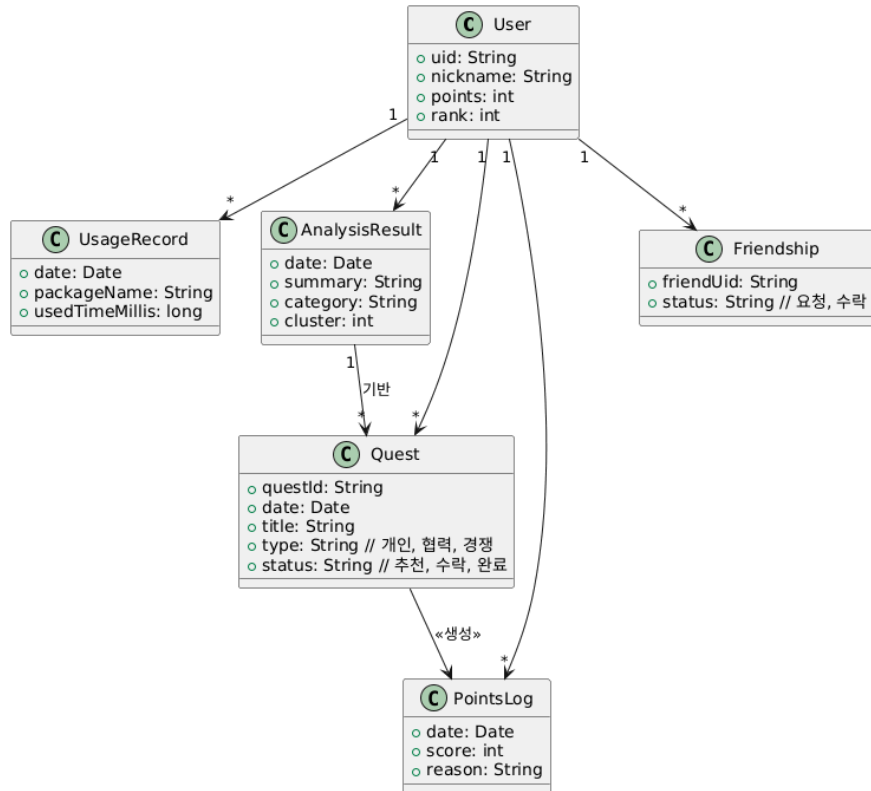
3. 과제 설계

3.1 요구사항 정의

1. 유스케이스 다이어그램



2. 클래스 다이어그램과 클래스 및 모듈 명세서



클래스명	주요 속성	설명	관련 유스케이스
User	uid, nickname, points, rank	시스템 사용자 계정 정보와 현재 포인트·랭킹을 관리	로그인, 포인트 적립, 랭킹 확인, 친구 관리
UsageRecord	date, packageName, usedTimeMillis	앱 사용시간 기록	앱 사용 로그 수집
AnalysisResult	date, summary, category, cluster	분석 결과(요약·카테고리·군집 정보)	기록 분석 및 요약
Quest	questId, date, title, type, status	개인/협력/경쟁 퀘스트 정보와 상태 관리	추천 퀘스트 제공, 퀘스트 수행
PointsLog	date, score, reason	포인트 적립 내역 기록	포인트 적립, 랭킹 확인
Friendship	friendUid, status	친구 관계 관리(요청, 수락)	친구 관리, 협력/경쟁 퀘스트 참여

2.1 Android App 모듈

- **Usage Collector:** 앱 포그라운드 사용시간을 수집하여 UsageRecord 생성 및 업로드
- **Auth & Profile:** 사용자 계정(User) 로그인 및 관리
- **Dashboard:** 분석 결과 및 퀘스트 현황을 조회하는 UI 제공
- **Quest UI:** 추천 퀘스트 수락 및 수행 현황 표시
- **Friends UI:** 친구 관리, 협력/경쟁 퀘스트 참여 기능 제공
- **Notifier:** 퀘스트, 포인트 적립, 친구 활동 등 알림 제공

2.2 분석 서버 모듈

- **ETL 모듈:** UsageRecord를 기반으로 AnalysisResult 생성
- **Analysis 모듈:** TF-IDF + KMeans 알고리즘을 이용한 로그 군집화 및 카테고리 분류
- **Quest Recommender:** 분석 결과에 기반한 추천 퀘스트 생성
- **Points Engine:** 퀘스트 완료 시 포인트 적립 및 PointsLog 기록
- **Ranking Builder:** User 포인트 집계 및 일간 랭킹 산출

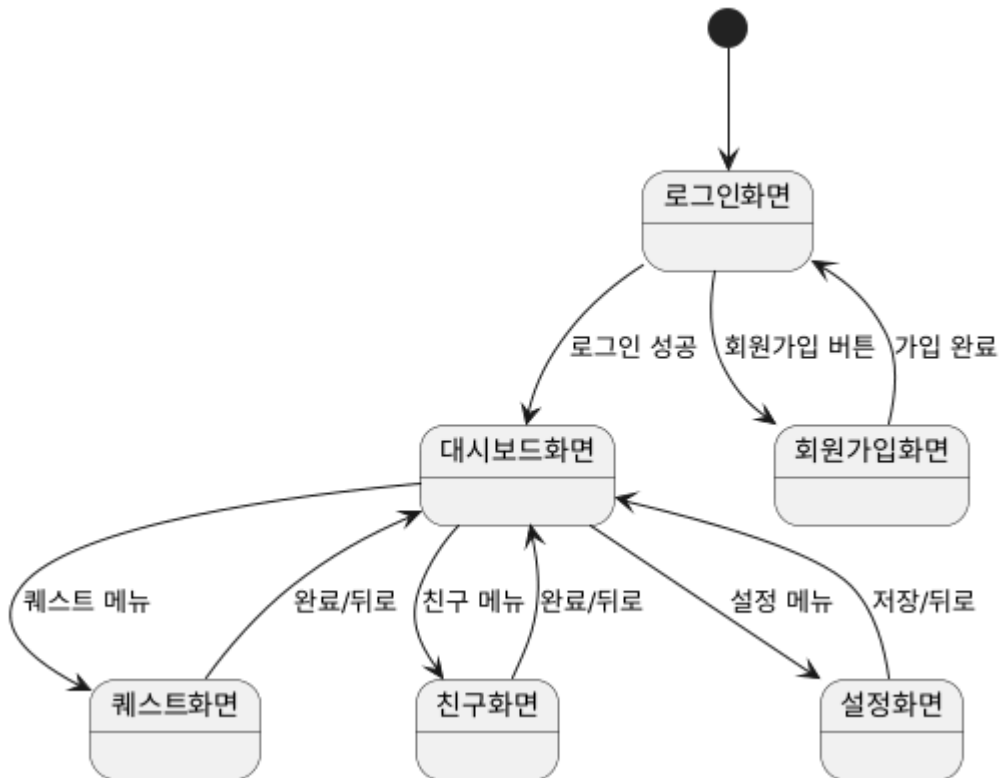
2.3 Firebase RTDB 모듈

- 저장소 역할: User, UsageRecord, AnalysisResult, Quest, PointsLog, Friendship 관리
- 특징: 실시간 데이터 동기화 지원, 보안 규칙 기반 접근 제어

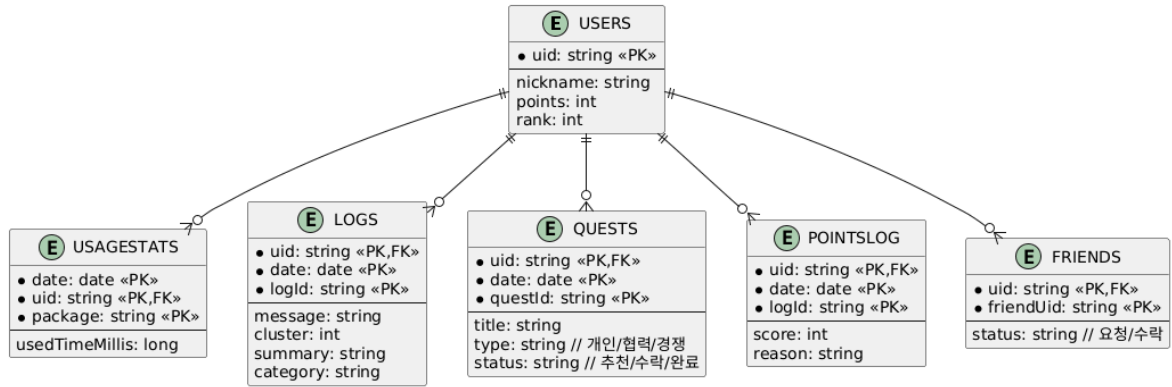
3. UI 분석 및 설계모델

화면	주요 요소	기능 설명	관련 유스케이스
회원가입/로그인 화면	입력창, 버튼	신규 사용자 계정 생성, 기존 사용자	UC1 회원가입/로그인

		로그인, 비밀번호 재설정	
대시보드 화면	일일 요약, 사용시간 그래프, 포인트/랭킹	앱 사용 기록 요약 확인, 분석 결과 표시, 포인트/랭킹 조회	UC2 로그 수집, UC6 요약 조회, UC5 랭킹 확인
퀘스트 화면	추천 퀘스트 카드, 수락/완료 버튼	개인 퀘스트 수락 및 완료 처리	UC3 추천 퀘스트 제공, UC4 퀘스트 수행
친구 화면	친구 목록, 초대/수락 버튼	친구 관리, 협력/경쟁 퀘스트 참여	UC6 친구 관리, UC7 협력 퀘스트, UC8 경쟁 퀘스트



4. E-R 다이어그램 및 DB 설계 구조

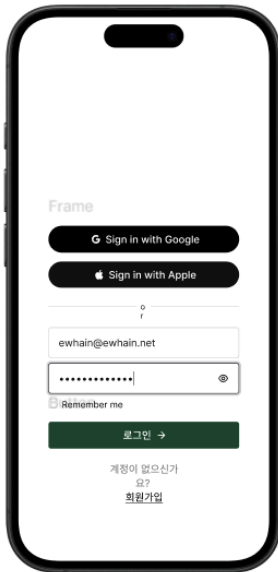


테이블명	주요 속성	설명
USERS	uid(PK), nickname, points, rank	사용자 계정 및 포인트/랭킹 관리
USAGESTATS	date(PK), uid(PK,FK), package(PK), usedTimeMillis	일별 앱 사용 기록 (패키지 단위)
LOGS	uid(PK,FK), date(PK), logId(PK), message, cluster, summary, category	앱 기록 분석 결과 (요약/카테고리/군집 포함)
QUESTS	uid(PK,FK), date(PK), questId(PK), title, type, status	사용자에게 제공된 퀘스트와 상태(추천/수락/완료)
POINTSLOG	uid(PK,FK), date(PK), logId(PK), score, reason	포인트 적립 내역 기록
FRIENDS	uid(PK,FK), friendUid(PK), status	친구 관계(요청/수락 상태)

5. 개발 프로젝트 주요 화면 구성도

1) 로그인/회원가입, 메인 화면

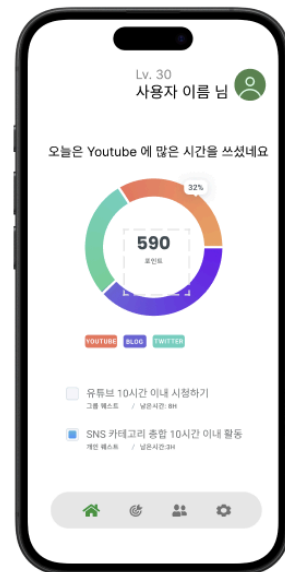
로그인 / 회원가입 초기 화면



회원가입 상세 화면



메인 화면



2) 퀘스트 및 랭킹, 친구 관련, 사용량 분석 화면

퀘스트 및 랭킹 화면



친구 신청 및 친구 목록 화면



친구 상세(현재 도전 중인 퀘스트)화면

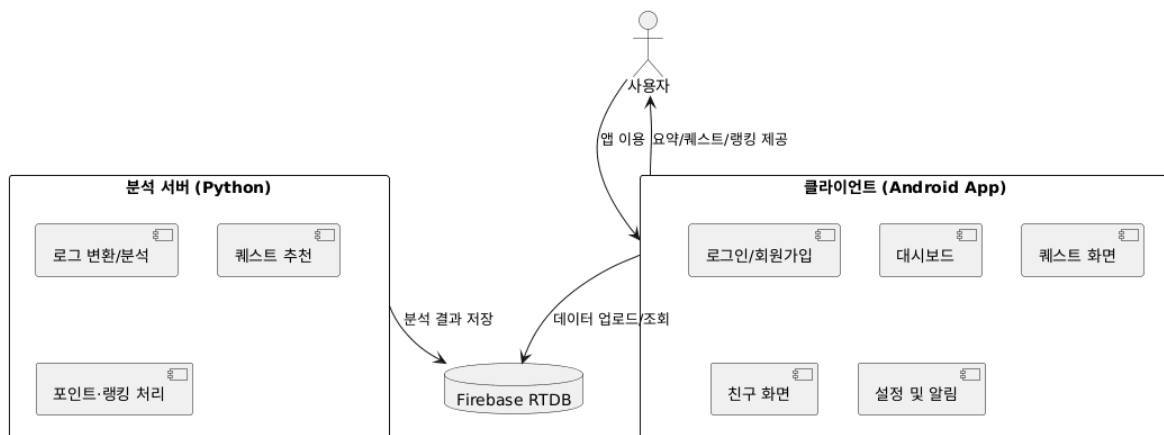


오늘의 어플 사용량 및 분석 화면



3.2 전체 시스템 구성

1. SW 구조 기술



2. 주요 구성 요소 및 역할

- 사용자: 앱을 설치하고 로그인하여 대시보드, 퀘스트, 친구 기능을 이용.
- 클라이언트(Android App): 앱 사용 기록 수집, 로그인/회원 관리, UI 제공, 알림 전달.
- 분석 서버(Python): 수집된 기록을 ETL 처리, TF-IDF/KMeans 및 GPT 기반 분석 수행, 퀘스트 추천 및 포인트 계산.
- Firebase RTDB: 모든 사용자·로그·퀘스트·포인트 데이터를 저장하고, 클라이언트와 서버 간 실시간 동기화 담당.

3. 외부/오픈소스 모듈

- Firebase SDK (Android, Python): RTDB와 실시간 연동.
- OpenAI GPT API: 로그 요약 및 카테고리 분류.
- Scikit-learn (TF-IDF, KMeans): 군집 분석.
- Android UsageStatsManager: 앱 사용 기록 수집.

3.3 주요 엔진 및 기능 설계

다음은 프로젝트에서 사용되는 엔진과 ~~현 상황~~ ~~진척도~~ 설계 및 구현 내용을 상세히 정리하였다.

1. 주요 엔진 및 활용 기술

주요 엔진	프로젝트 내 활용도 서술
OpenAI API	기록된 사용자의 앱 사용 로그를 읽어와 앱 사용 패턴을 분석, 사용 습관에 따른 추가 피드백을 출력함.
UsageStats API	해당 API를 통해 안드로이드 환경에서 사용된 앱의 활동 로그를 전부 불러옴.
Firebase Realtime DB	UsageStats API를 통해 불러온 앱의 활동 로그를 저장함.
K-means /HDBSCAN	데이터베이스에 기록된 앱 활동 로그를 기반으로 앱의 종류에 따라 카테고리를 분류하여 앱의 카테고리 명을 데이터베이스에 추가하여 저장함.

2. 주요 기능별 구현 내용 기술

주요 기능	구현 내용 서술
사용자 맞춤 목표 설정 (퀘스트 생성)	<p>1. 사용자가 직접 퀘스트를 생성 사용자는 'QuestActivity'를 통해 제어하고자 하는 앱과 목표 시간, 달성 조건(이상/이하 사용)을 설정한다. 해당 데이터는 'QuestManager'를 통해 Firebase의 /quests/{UID} 경로에 실시간으로 저장된다.</p> <p>2. 데이터베이스 기반, 카테고리 선택에 따른 퀘스트 생성 퀘스트 설정 화면 내 사용자의 카테고리 선택에 따라, 사용자별 Firebase의 /quests/{UID} 경로에 저장되었던 기존 퀘스트 내역을 기반으로 동일한 카테고리 내에 포함되어 있는 앱 관련 퀘스트를 추천한다. 선택한 카테고리를 기반으로 설정된 추천 퀘스트를 수락/거절하여 신규 퀘스트를 생성한다.</p>
분석 & 요약 피드백	<p>1. 사용자의 앱 사용 기록 수집 안드로이드의 UsageWorker가 백그라운드에서 사용자 개입 없이 앱 사용 기록을 수집하여 Firebase에 실시간으로 업로드한다.</p> <p>2. 수집된 기록 AI 분석 분석 서버는 하루 동안 수집된 로그를 취합하여 OpenAI GPT 모델에게 전달한다. GPT는 해당 로그를 기반으로 사용자의 행동 패턴을 분석하여 '오늘의 요약 메시지'와 '카테고리별 사용량' 데이터를 생성하고, 이를 Firebase의 /dashboard 경로에 기록한다.</p> <p>3. 요약 피드백 제공 메인 화면(MainActivity)은 Firebase를 통해 분석 결과가 업데이트되는 즉시, AI가 작성한 요약 메시지와 직관적인 파이 차트 형태를 통해 사용자에게 피드백을 제공한다.</p>
포인트 & 랭킹 시스템	<p>1. 퀘스트 판정 및 포인트 제공 퀘스트 성공 판정 시, 서버는 목표 달성 난이도에 비례하여 보상 포인트를 계산한다. 이 포인트는 사용자의 누적 포인트(/users/{UID}/points)에 합산되며, 상세 내역은 포인트 로그(/pointsLog)에 기록되어 관리된다.</p> <p>2. 사용자 별 랭킹 집계 서버는 변동된 포인트 데이터(DB 내 /pointsLog에 저장된 사용자별 포인트량)를 기반으로 전체 사용자의 순위를 오름차순으로 정렬하여 랭킹 데이터(/ranking)를 갱신하고, 이를 랭킹 화면의 UI 상으로 사용자에게 정보를 제공한다.</p>

3. 개발 진척도

해당 주차	진척도
1~2주차	핵심 화면 흐름도 제작, 불필요한 코드 정리, 팀 내 재정비 완료
3~5주차	'퀘스트' 설정 UI, 추가 Firebase 연동, 포인트/파드백 로직 개발 진행 중
6~7주차	GPT와 앱 사용 분류 개선, 유저의 커뮤니티 기능 기획 및 설계 예정

3.4 주요 기능의 구현

다음은 **3.3** 주요 엔진 및 기능 설계 에서 설명한 기능의 흐름을 중심으로 기술하였다.

1. 데이터 수집 단계 (Background)

사용자가 별도의 조작을 하지 않아도 앱은 백그라운드에서 필요한 데이터를 지속적으로 수집하여 분석의 기반을 마련한다.

자동 로그 수집: 안드로이드의 **UsageWorker**가 백그라운드에서 주기적으로 동작하며 **UsageStats API**를 통해 사용자의 앱 실행 기록을 수집한다.

실시간 데이터 동기화: 수집된 로우 데이터(Raw Data)는 사용자 개입 없이 **Firestore Realtime DB**의 **/usageStats** 경로에 실시간으로 업로드되어 분석 서버가 언제든지 접근할 수 상태로 유지된다.

2. 목표 설정 단계 (Active Interaction)

사용자는 수집된 데이터를 바탕으로 자신의 습관을 개선하기 위한 퀘스트(목표)를 능동적으로 설정한다.

사용자 맞춤 퀘스트 생성: 사용자는 **QuestActivity**를 통해 제어하고 싶은 앱을 선택하고, 목표 시간 및 성공 조건(이상/이하 사용)을 직접 설정한다. 생성된 퀘스트 정보는 **QuestManager**를 통해 **Firestore**의 **/quests/{UID}** 경로에 저장된다.

카테고리 기반 퀘스트 추천: 사용자가 특정 카테고리를 선택하면, 시스템은 **DB**에 축적된 기존 퀘스트 내역과 **K-means** 알고리즘 등을 통해 분류된 앱 카테고리 정보를 기반으로 동일 카테고리 내의 연관 앱 퀘스트를 추천한다. 사용자는 이를 수락하거나 거절하여 손쉽게 목표를 설정할 수 있다.

3. 분석 및 피드백 단계 (AI Processing & Feedback)

하루의 사용이 끝나거나 일정 주기가 되면, 서버는 축적된 데이터를 **AI** 엔진을 통해 분석하고 사용자에게 유의미한 피드백으로 가공하여 전달한다.

AI 행동 패턴 분석: 분석 서버는 하루 동안 취합된 로그 데이터를 **OpenAI GPT** 모델에 전달한다. **GPT** 엔진과 자체 군집화 알고리즘(**K-means** 등)은 사용자의 행동 패턴을 다각도로 분석하여 정량적 데이터(카테고리별 사용량)와 오늘의 요약 메시지를 생성한다. 시각적 피드백 제공: 분석 결과는 **Firestore**의 **/dashboard** 경로에 업데이트된다.

MainActivity는 이 변경 사항을 즉시 감지하여, 사용자에게 **AI**가 생성한 하루 요약 메시지를 텍스트로, 카테고리별 사용 비중을 파이 차트 형태로 시각화하여 제공한다.

4. 보상 및 경쟁 단계 (Gamification)

사용자의 목표 달성 여부를 판정하고 이를 보상과 경쟁으로 연결하여 지속적인 앱 사용 동기를 부여한다.

퀘스트 성공 판정 및 보상: 서버는 사용자의 실제 앱 사용 기록과 설정된 퀘스트 목표를 비교하여 성공 여부를 판정한다. 목표 달성 시 난이도에 비례한 포인트를 계산하여 `/users/{UID}/points`에 합산하고, 상세 내역을 `/pointsLog`에 기록한다.

실시간 랭킹 갱신: 서버는 변동된 사용자별 포인트 데이터를 기반으로 전체 사용자의 순위를 산정하여 `/ranking` 데이터를 갱신한다. 사용자는 랭킹 UI를 통해 자신의 현재 위치를 확인하고 타 사용자와 경쟁을 할 수 있다.

3.5 기타

1. OpenAI GPT (요약/분류)

- Firebase에 저장된 하루치 앱 사용 로그를 입력으로 받아,
 - 사용자의 앱 사용 행동을 한 줄 요약으로 생성
 - 앱 사용 목적을 분류하여 **category** 필드로 저장 (예: 공부 / 엔터테인먼트 등)
- GPT는 앱 카테고리를 판단하는 것이 아니라, 앱 사용 기록의 의미를 정리하는 역할을 수행한다.

2. TF-IDF + KMeans 클러스터링 (군집화 분석)

- 앱 이름 문자열 데이터를 TF-IDF로 벡터화하여 KMeans로 군집화.
- 비슷한 유형의 앱 사용 패턴(영상, SNS, 생산성, 멀티태스킹 등)을 그룹으로 묶고, 해당 결과를 퀘스트 추천 로직에 반영한다.
- 군집 결과는 **cluster** 필드로 Firebase에 저장된다.

AI 동작 흐름

1. Android에서 `UsageStatsManager`로 앱 사용 로그 수집
2. Firebase Realtime DB `/usageStats/{date}/{uid}/`에 저장
3. Python 분석 서버에서 로그 조회 후:
 - TF-IDF + KMeans로 군집화 → **cluster** 결정

- OpenAI GPT에 로그 전달 → 요약 및 카테고리 생성

4. 결과를 `/logs/{uid}/{date}/{logID}` 에 저장

5. 앱에서 사용자에게 요약/퀘스트/포인트 제공