

# New Proj Ideas

## ▼ (강) 돈줄: 사용자 참여형 착한 소비 인증 챌린지 플랫폼

### 1. 🎯 주제

사용자 참여형 착한 소비 인증 챌린지 플랫폼

사용자들이 선한 영향력을 가진 가게를 직접 제보하고,  
영수증 인증을 통해 착한 소비를 기록·분석·시각화하는  
데이터 기반의 참여형 착한 소비 플랫폼

### 2. 🧠 사용할 오픈소스 / 라이브러리

- OCR

- [Tesseract OCR](#) : 오픈소스 OCR 엔진
- 또는 [Kakao OCR API](#) (더 정확함, 쉽지만 API 호출 비용 있음)

- 자연어 처리

- [KoNLPy](#) (Python) → 품목 키워드 추출용
- 간단한 품목 분류 룰/사전 기반 매칭 가능

- 시각화

- [Chart.js](#), [Recharts](#) (React용)
- 월별 소비 히트맵, 카테고리별 점수 분포 등

### 3. 🔄 전체 프로세스

#### 1. 착한 가게 제보

- 사용자: 가게 정보 + 착한 사유 등록
- 관리자 or 커뮤니티 투표로 승인/거절

#### 2. 가게 등록 및 분류

- 카테고리: 소상공인 / 친환경 / 공익적 가치 등

- 등록된 가게는 지도/리스트에 노출

### 3. 영수증 업로드

- OCR 처리 → 상호명 / 품목 추출
- 등록된 착한 가게와 매칭되면 인증 완료

### 4. 점수 계산 + 피드백 메시지 + 기부

- 소비 카테고리 기반 점수 계산 (환경/지역/공익 점수)
- "오늘 환경점수 2점 + 지역점수 1점 획득" 메시지 제공
- 챌린지 누적 데이터 기록, 기부(네이버 콩처럼)

### 5. 대시보드 / 마이페이지

- 개인 소비 이력 시각화
- 누적 점수, 배지, 카테고리별 소비 트리 제공
- 챌린지 달성 현황 표시

## 4. 기술적 구현

기능	기술 스택 / 구현 방식
회원 관리	Spring Boot + JWT 로그인
OCR 처리	Python 서버 + Tesseract, React에서 이미지 업로드
착한 가게 제보/등록	DB + 관리자 페이지 구성
소비 분류	사전 기반 키워드 매칭 or 품목 분류 룰
점수 계산	카테고리 기반 점수 테이블 적용
피드백 메시지	소비 결과에 따라 동적 메시지 반환
마이페이지 / 시각화	Chart.js or Recharts, 개인별 소비 히스토리 표시

## 5. 예상 소요 시간

총 작업 기간: 3~4주 (팀 기준)

| 기본 기능 + 간단한 시각화 + 챌린지 구현까지

단계	기간	메모
기획/역할 분배	2~3일	기능 범위, 페이지 정의

DB/ERD 설계	1~2일	가게, 소비, 사용자, 점수 테이블
백엔드 API 구현	5~7일	인증, 업로드, 처리, 점수 계산
프론트 UI 구성	4~6일	제보/업로드/마이페이지
OCR 연동 + 테스트	3~5일	Python + API 연결 or Tesseract
시각화/피드백 로직	2~4일	점수표 기반 계산 로직
마무리 디버깅/통합	3~4일	발표용 정리 포함

## 6. 📌 기존 사례 및 차별점

### 기존 사례

서비스	설명	한계점
<u>우리동네 돈쫌내기</u>	착한 가게 지도 제공	참여 구조 없음, 영수증 인증 없음
네이버 영수증 리뷰	오프라인 리뷰 + 포인트	가게 제보/착한 소비 분류 없음
당근마켓 동네가게	지역 가게 노출	착한 소비 개념 없음, 인증 구조 없음

### 우리 서비스의 차별점 요약

항목	기존 서비스	우리 서비스	차별점
가게 등록	기관 or 고정 DB	<b>사용자 제보 기반 동적 생성</b>	커뮤니티 기반 확장
인증 방식	없음 or 리뷰 기반	<b>OCR 기반 영수증 인증</b>	실제 소비 확인
리워드 방식	포인트 or 없음	<b>점수 + 피드백 + 챌린지 시각화</b>	동기 부여, 지속 사용 유도
기술 요소	정적 지도, 리뷰	<b>OCR, 점수 계산, 시각화, 사용자 히스토리</b>	기술적 깊이 보완

#### ▼ 서희정

##### ▼ 강아지와 함께하는 일정

### 🌟 주제: 반려견과 함께하는 하루 일정 짜기 (LLM 모델 활용)

반려동물과의 외출을 즐겁게 만들기 위해, **사용자가 원하는 시간을 입력하면 AI 모델이 맞춤형 하루 일정을 생성**해주는 서비스야. 예를 들어, 반려견과 3시간 동안 즐겁게 놀 수 있는 일정을 LLM 모델을 활용해 자동으로 구성해주는 거야.

## ✓ 목표 기능

- 사용자가 입력한 조건에 따라 AI 모델이 일정 제안하기.
- 3시간의 시간을 다양한 활동으로 나눠서 제안하기.
- 사용자 맞춤형 추천 (예: 반려견의 크기, 나이, 활동 수준 등 고려).

## 🔧 구현 기술 스택

- **Frontend:** React (TailwindCSS로 스타일링)
- **Backend:** Flask (Python) 또는 FastAPI
- **AI 모델 (LLM):** OpenAI GPT-4 또는 Hugging Face의 `text-davinci-003`, `gpt-3.5-turbo`
- **Database:** MongoDB (NoSQL) 또는 PostgreSQL (SQL) - 사용자 정보와 반려동물 프로필 저장
- **Deployment:** Vercel (Frontend) + AWS, Render, or DigitalOcean (Backend)

## 📌 기능별 상세 구현 방법

### 🔍 1. 사용자 입력 받기 (Frontend)

사용자가 입력할 정보:

- 🕒 총 시간 (예: 3시간)
- 📅 활동을 나눌 방식 (예: 30분 x 6, 1시간 x 3 등)
- 🐕 반려동물 정보 (크기, 나이, 활동 수준 등)
- 🎯 선호하는 활동 종류 (산책, 공원 놀이, 실내 활동 등)

```
// React (Next.js 또는 Vite 기반)
```

```
import React, { useState } from 'react';
```

```
const ActivityPlanner = () => {  
  const [totalTime, setTotalTime] = useState(3); // 기본값 3시간  
  const [segments, setSegments] = useState(6); // 기본값 30분 x 6  
  const [dogInfo, setDogInfo] = useState("");
```

```

const [preferences, setPreferences] = useState("");

const handleSubmit = async (e) => {
  e.preventDefault();

  const response = await fetch('/api/plan', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ totalTime, segments, dogInfo, preferences })
  });

  const data = await response.json();
  console.log(data.plan); // 일정 결과 확인
};

return (
  <div className="p-6 max-w-md mx-auto bg-white rounded-xl shadow-md space-y-4">
    <h1 className="text-xl font-bold">반려견과 함께하는 하루 일정 짜기</h1>
    <form onSubmit={handleSubmit} className="space-y-4">
      <div>
        <label>총 시간 (시간 단위): </label>
        <input type="number" value={totalTime} onChange={e => setTotalTime(e.target.value)} className="border p-1"/>
      </div>
      <div>
        <label>활동 분할 개수: </label>
        <input type="number" value={segments} onChange={e => setSegments(e.target.value)} className="border p-1"/>
      </div>
      <div>
        <label>반려동물 정보: </label>
        <textarea value={dogInfo} onChange={e => setDogInfo(e.target.value)} className="border p-1 w-full"/>
      </div>
      <div>

```

```

        <label>선호하는 활동 종류: </label>
        <textarea value={preferences} onChange={e ⇒ setPref
erences(e.target.value)} className="border p-1 w-full"/>
    </div>
    <button type="submit" className="bg-blue-500 text-wh
ite p-2 rounded">일정 짜기</button>
</form>
</div>
);
};

export default ActivityPlanner;

```

## 🔍 2. 백엔드 설정 (Flask & OpenAI API 연결)

```

# app.py (Flask)
from flask import Flask, request, jsonify
import openai

app = Flask(__name__)

# OpenAI API Key 설정
openai.api_key = "YOUR_OPENAI_API_KEY"

@app.route('/api/plan', methods=['POST'])
def generate_plan():
    data = request.json
    total_time = data['totalTime']
    segments = data['segments']
    dog_info = data['dogInfo']
    preferences = data['preferences']

    prompt = f"""
나는 반려견과 함께하는 하루 일정을 짜주려고 해.
총 {total_time}시간 동안 {segments}개의 활동으로 나누어 주세요.
반려견 정보: {dog_info}
선호하는 활동: {preferences}

```

각 활동마다 구체적인 제안을 해주세요.

"""

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": prompt}],
    max_tokens=500
)
plan = response.choices[0].message['content']

return jsonify({"plan": plan})

if __name__ == '__main__':
    app.run(debug=True)
```

### 3. 프론트엔드 & 백엔드 연결하기

Frontend의 `fetch` 요청이 Flask API로 보내지고, 일정 결과를 받아서 표시하는 방식으로 동작함.

### 구현 결과 예시 (LLM 모델로 생성된 일정)




사용자 입력:

- 총 시간: **3시간**
- 활동 분할: **30분 x 6**
- 반려동물 정보: **중형견, 2살, 활발함**
- 선호 활동: **산책, 공놀이, 물놀이**

LLM 모델 출력 (예시):

1. 09:00 - 09:30: 공원 산책하기 (가벼운 산책으로 준비 운동)
2. 09:30 - 10:00: 공놀이 (공원 내 넓은 공간에서 공 던지기)
3. 10:00 - 10:30: 강아지 전용 수영장 가기
4. 10:30 - 11:00: 수영 후 건조 및 간식 시간
5. 11:00 - 11:30: 반려동물 카페 방문하여 휴식
6. 11:30 - 12:00: 짧은 산책 후 귀가

## 기술적 실현 가능성 평가

-  LLM 모델로 텍스트 기반 일정 추천은 충분히 가능.
-  Flask와 React를 연동하여 UI/UX 구현도 가능.
-  전체 일정 생성은 LLM 모델에 의존하므로 API 연결 속도만 최적화하면 됨.

## 다음 단계


- UI 개선 (일정 출력 결과를 깔끔하게 보여주기)
- 일정 추천 결과를 저장할 데이터베이스 연동 (사용자별 맞춤 데이터 유지)
- 활동 추천의 다양성 향상을 위해 모델 프롬프트 최적화

구현을 직접 해볼 수 있게 다음 단계로 도와줄까? 😊

### ▼ 이미지 생성 ai를 결합한 꿈 일기




내가 말한 '**사용자 개인화에 집중한 서비스 (페르소나 고정형 AI 생성)**' 를 구현하려면, 다음과 같은 단계로 개발을 진행하면 돼.

## 1. 사용자 페르소나 설정 (고정 주인공 설정)

 **목적:** 사용자가 지정한 외모나 캐릭터 스타일을 기반으로 일관성 있는 이미지 생성하기.

 **방법:**

### 1. 사용자 입력 데이터 수집

- 사용자에게 자신의 페르소나를 설정할 수 있게 UI 제공.
- 사용자가 다음 중 선택하도록 함:
  -  **사진 업로드** (본인의 얼굴이나 캐릭터의 이미지)
  -  **텍스트 기반 설명** (예: "긴 머리의 소녀, 푸른 눈, 몽환적인 스타일")
  -  **특정 스타일 선택** (예: 픽사 스타일, 애니메이션 스타일, 웹툰 스타일 등)

### 2. 사용자 페르소나 데이터 저장



- 사용자가 설정한 페르소나 정보를 데이터베이스에 저장 (DB Schema 예시)

```
CREATE TABLE Users (  
    user_id INT PRIMARY KEY,  
    username VARCHAR(50),  
    persona_description TEXT,  
    persona_image BLOB,  
    style_preference VARCHAR(50)  
);
```

- 이미지 기반 정보는 **파일 저장 시스템** 또는 **클라우드 스토리지** (예: **AWS S3, Firebase Storage**) 에 저장 가능.

## ✓ 2. 페르소나 고정형 이미지 생성 모델 구축

📌 **목적:** 사용자 고유의 페르소나를 기준으로 AI 이미지를 일관되게 생성하기.

📌 **방법:**

### 1. 모델 선택

- OpenAI의 DALL·E, Stable Diffusion 또는 **DreamBooth** (특정 캐릭터 학습에 특화된 모델) 사용.
- 예를 들어, **Stable Diffusion + DreamBooth** 조합으로 사용자 지정 스타일을 고정 가능.
  - Hugging Face 모델 예시: [stable-diffusion-v1-5](#)
  - DreamBooth 구현 예시: [dreambooth-hf](#)

### 2. 사용자 스타일 학습

- 사용자가 업로드한 이미지를 모델에 학습시켜서 **개인화된 스타일로 변환** 가능하게 만들.
- 예시 코드 (Python, [diffusers](#) 라이브러리 사용)

```
from diffusers import StableDiffusionPipeline  
import torch
```

```

model_id = "CompVis/stable-diffusion-v-1-5"
pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_
dtype=torch.float16)
pipe.to("cuda")

prompt = "A beautiful dream world with a character resembling
the user, in a fantasy style"
image = pipe(prompt).images[0]

image.save("user_dream_image.png")

```

### 3. 페르소나 유지

- 사용자가 생성할 때마다 **기존 페르소나 정보**를 prompt에 포함하여 모델에 입력.
- 예시 프롬프트:

"A dream scene featuring [User's Persona Description], in a [User's Style Preference] style."

## ✓ 3. 사용자 경험 (UI/UX) 설계

 **목적:** 사용자 친화적인 인터페이스로 서비스의 차별성을 극대화하기.

 **방법:**

#### 1. 페르소나 설정 화면 (초기 설정 또는 변경 가능)

- 이미지 업로드, 텍스트 설명 입력, 스타일 선택 기능 제공.
- 설정 후 미리보기 기능으로 사용자에게 결과를 보여줌.

#### 2. 페르소나 기반 이미지 생성 화면

- 프롬프트 입력과 함께 **페르소나 고정 옵션 활성화** 버튼 제공.
- 사용자별로 이전의 결과물도 확인할 수 있게 **갤러리 형태로** 제공.

## ✓ 4. 데이터베이스 설계 및 유지

📌 **목적:** 사용자가 언제든지 동일한 페르소나로 꿈을 생성할 수 있도록 데이터 유지.

📌 **방법:**

- 사용자별로 고정된 페르소나 데이터를 저장하기 위해 데이터베이스 활용 (SQL or NoSQL).
- 예를 들어, MongoDB를 사용한다면:

```
{
  "user_id": "12345",
  "username": "dreamer1",
  "persona": {
    "description": "긴 머리의 소녀, 푸른 눈, 몽환적인 스타일",
    "style_preference": "Anime",
    "image_url": "https://your-storage/user12345/persona.jpg"
  }
}
```

## ✅ 5. 웹/어플리케이션으로 구현하기 (두 달 안에 가능)

📌 **핵심 기술 스택**

- **Backend:** Flask (Python), FastAPI, or Django (Python)
- **Frontend:** React, TailwindCSS
- **AI 모델:** Stable Diffusion, DreamBooth
- **데이터베이스:** MongoDB (NoSQL), PostgreSQL (SQL)
- **스토리지:** Firebase Storage, AWS S3

📌 **프로젝트 일정 (예상)**

주차	작업 내용
1주차	기본 UI/UX 설계 및 Flask/FastAPI 설정
2주차	사용자 데이터베이스 구축 및 페르소나 설정 기능 구현
3주차	AI 모델 학습 파이프라인 구축 및 연결
4주차	사용자 이미지 생성 기능 구현 및 테스트

5주차	사용자 갤러리 및 꿈 기록 기능 구현
6주차	디버깅, 최적화 및 배포 (웹 or 앱)
7주차	사용자 피드백 반영 및 기능 개선
8주차	최종 마무리 및 발표 준비

## 추가 팁

- 만약 웹으로 구현한다면 **Flask + React 조합**으로 빠르게 구축 가능.
- 앱으로 만들고 싶다면 **Flutter** 또는 **React Native** 사용 추천.

구체적인 코드 구현 예시나 구조가 필요하면 알려줘! 😊

### ▼ 냉장고 사진으로 강아지 음식 만들어주는 ai 레시피 어플

💡 아이디어: 냉장고 사진 찍으면 만들 수 있는 강아지 간식 & 식사 레시피 추천 앱

## 아이디어 설명

### 1. 핵심 기능

- 사용자가 냉장고 사진을 찍어서 업로드하면, **이미지 인식 모델**이 재료를 파악하고,
- \*강아지의 상태(노견, 아픈 강아지, 활동량 많은 강아지 등)\*\*에 맞는 간식이나 밥 레시피를 추천해줌.
- 간단한 레시피, 건강에 좋은 재료 위주로 제공.

### 2. 특징

- 📷 **사진 인식 기반 추천** (냉장고 속 재료 분석)
- 🐕 **강아지의 상태에 따른 맞춤 추천** (노견, 활동량 많은 강아지, 다이어트 중, 아픈 강아지 등)
- 🍲 **간식과 밥을 구분하여 추천** (예: 아픈 강아지에게는 부드러운 식사 추천)
- 📁 **사용자 기록 저장 및 분석** (사용자가 만든 레시피 기록 유지)

## 구현 기술 스택

- **Frontend:** React Native (앱 개발), TailwindCSS (웹 버전 스타일링)

- **Backend:** FastAPI (Python)
  - **AI 모델 (이미지 인식):** YOLOv8, Google Vision API, OpenCV
  - **LLM 모델 (텍스트 기반 추천):** GPT-4, BERT 등
  - **Database:** MongoDB (NoSQL) 또는 PostgreSQL (SQL)
  - **Storage:** Firebase Storage or AWS S3 (사진 저장)
  - **Deployment:** AWS EC2, Render, or Vercel
- 

## 구현 방법 (기능별로 정리)

### 1. 이미지 인식 기능 구현 (재료 인식)

사용자가 찍은 냉장고 사진에서 재료를 인식하는 단계.

#### 이미지 인식 모델 설정

- **YOLOv8 (Ultralytics):** 객체 인식에 특화되어 있음.
- **Google Vision API:** 빠르고 정확하게 다양한 재료를 인식 가능.

### 예제 코드 (YOLOv8 사용)

```
from ultralytics import YOLO
from PIL import Image

# 모델 로드
model = YOLO("yolov8x.pt") # YOLOv8 모델 (기존 훈련된 모델 사용)

# 이미지 불러오기
image = Image.open("fridge.jpg")

# 이미지 예측
results = model(image)

# 인식된 객체 출력
for result in results:
    print(result.names)
```

## 2. 사용자 정보 입력 및 설정

사용자가 자신의 강아지 정보를 입력하고 맞춤 추천을 받을 수 있게 설정.

### 사용자 정보 예시

- 반려견 나이: (어린 강아지 / 노견)
- 활동 수준: (활발 / 보통 / 적음)
- 건강 상태: (다이어트 중 / 특정 질병 있음 / 일반)
- 선호하는 식사 종류: (간식 / 밥 / 특별식)

### 데이터베이스 스키마 (MongoDB 예시)

```
{
  "user_id": "user123",
  "dog_info": {
    "name": "Bobby",
    "age": 7,
    "activity_level": "적음",
    "health_condition": "노견",
    "preferences": "부드러운 음식"
  }
}
```

## 3. 레시피 추천 기능 (LLM 모델 사용)

사용자가 찍은 사진과 강아지 정보를 바탕으로 LLM 모델이 레시피를 추천해 줌.

### 모델 설정 (GPT-4 API 예시)

```
import openai

openai.api_key = "YOUR_API_KEY"

def get_recipe(ingredients, dog_info):
    prompt = f"""
    냉장고 속 재료: {' '.join(ingredients)}
    반려견 정보: {dog_info}
    """
```

위의 정보를 바탕으로 강아지를 위한 건강한 간식 레시피를 추천해줘.  
레시피는 간단하고 강아지에게 해로운 재료는 사용하지 않아야 해.  
""

```
response = openai.ChatCompletion.create(  
  model="gpt-4",  
  messages=[{"role": "user", "content": prompt}],  
  max_tokens=500  
)  
  
return response.choices[0].message['content']
```

#### 4. 결과 출력 & 사용자 피드백 수집 (UI 구현)

사용자에게 레시피를 보여주고 저장하거나 공유할 수 있게 제공.

```
// React Native 예시 (TailwindCSS 사용)  
import React, { useState } from 'react';  
import { View, Text, TextInput, Button, Image } from 'react-native';  
  
const RecipeApp = () => {  
  const [recipe, setRecipe] = useState("");  
  
  const handleGetRecipe = async () => {  
    const response = await fetch('/api/get_recipe', {  
      method: 'POST',  
      headers: { 'Content-Type': 'application/json' },  
      body: JSON.stringify({ ingredients: ['당근', '닭가슴살'], dogInfo: '노견, 부드러운 음식 선호' })  
    });  
    const data = await response.json();  
    setRecipe(data.recipe);  
  };  
  
  return (  
    <View className="p-4">  
      <Button title="레시피 가져오기" onPress={handleGetRecipe} />  
    </View>  
  );  
};
```

```

    {recipe && <Text className="mt-4 text-black">{recipe}</T
ext>}
  </View>
);
};

export default RecipeApp;

```

## 17 개발 일정 (8주 프로젝트)

주차	작업 내용
1주차	이미지 인식 모델 학습 및 데이터셋 구축
2주차	Flask / FastAPI 서버 구축 및 API 연결
3주차	사용자 정보 입력 시스템 및 데이터베이스 구축
4주차	LLM 모델 연결 및 레시피 추천 기능 구현
5주차	사용자 UI/UX 디자인 및 피드백 시스템 추가
6주차	버그 수정 및 최적화
7주차	배포 및 사용자 테스트
8주차	최종 마무리 및 발표 준비

### 💡 차별화 포인트

- 📷 사진 인식 기반 추천 (재료를 분석하여 자동 추천)
- 🐕 개별 강아지 맞춤형 레시피 제공 (나이, 건강 상태, 선호도 반영)
- 🇮🇹 사용자 기록 저장 및 분석 (사용자가 만든 레시피를 기록하여 피드백 제공)

다음 단계로 실제 코드를 작성하거나 앱으로 배포하는 방법까지 도와줄까? 😊

#### ▼ 강아지 산책 틴더 어플

💡 아이디어: GPS 기반 강아지 산책 평가 및 애견인 만남 앱

### 🔥 아이디어 설명

#### 1. 핵심 기능

- 📍 GPS를 이용해 산책 경로 기록 및 평가하기.



- 🐕 강아지의 정보(크기, 활동 수준 등)에 따라 경로 평가 가능.
- 🔔 근처 애견인이 있으면 알림 기능으로 즉석 만남 기회 제공.

## 2. 특징

- 경로 평가 기능을 활용해 “대형견에게 적합한 산책로”, “소형견에게 안전한 길” 같은 정보를 제공.
- 다른 애견인들과 즉석으로 만남을 주선하여 커뮤니티 기능 제공.
- 사용자 기록을 데이터베이스로 분석해 맞춤형 산책 추천 경로 제공.

## 📌 구현 기술 스택

- **Frontend:** React Native (앱 개발), TailwindCSS (웹 버전 스타일링)
- **Backend:** FastAPI (Python)
- **GPS 트래킹:** React Native Geolocation API, Google Maps API
- **Database:** PostgreSQL or Firebase (사용자 데이터 및 경로 평가 저장)
- **실시간 알림 기능:** WebSockets (Socket.io), Firebase Messaging
- **AI 모델 (경로 추천 분석):** GPT-4, Scikit-Learn (사용자 리뷰 분석)

## 📌 기능별 상세 구현 방법

### 🔍 1. GPS 기반 산책 경로 기록 (Frontend & Backend)

사용자가 앱을 켜고 GPS를 활성화하면 산책 경로를 기록함.

#### 📱 React Native (Frontend) 예제

```
import React, { useState, useEffect } from 'react';
import { View, Text, Button, PermissionsAndroid } from 'react-native';
import Geolocation from 'react-native-geolocation-service';

const WalkTracker = () => {
  const [location, setLocation] = useState(null);
  const [route, setRoute] = useState([]);

  useEffect(() => {
```

```

const requestLocationPermission = async () => {
  const granted = await PermissionsAndroid.request(
    PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
  );
  if (granted === PermissionsAndroid.RESULTS.GRANTED) {
    Geolocation.watchPosition(
      position => {
        const { latitude, longitude } = position.coords;
        setLocation({ latitude, longitude });
        setRoute(prevRoute => [...prevRoute, { latitude, longitude }]);
      },
      error => console.log(error),
      { enableHighAccuracy: true, distanceFilter: 10 }
    );
  }
};

requestLocationPermission();
}, []);


return (
  <View>
    <Text>현재 위치: {location ? `${location.latitude}, ${location.longitude}` : '위치 불러오는 중...'}</Text>
    <Button title="경로 저장" onPress={() => console.log(route)} />
  </View>
);
};

export default WalkTracker;

```

## 🔍 2. 경로 평가 기능 (강아지 정보별로 평가 가능하게 설정)

- 사용자가 산책 후 경로에 대해 평가를 남길 수 있도록 설계.
- 평가 항목:
  - 🌳 경로의 안전성 (대형견/소형견 기준)
  - 🐾 편안함 (예: 포장이 잘 되어 있는지, 잔디가 많은지 등)

-  거리 및 난이도 (짧음, 보통, 길음 / 평지, 오르막길 등)

### 3. 강아지 정보에 따른 맞춤 경로 추천 기능

- 사용자가 입력한 강아지 정보 (크기, 활동 수준, 건강 상태 등) 기반으로 경로를 추천.
- 사용자가 평가한 데이터 기반으로 추천 알고리즘 학습 가능 (Scikit-Learn, GPT-4 등).

### 예제 데이터베이스 스키마 (PostgreSQL)

```
CREATE TABLE Users (  
    user_id SERIAL PRIMARY KEY,  
    username VARCHAR(50),  
    dog_size VARCHAR(10), -- 소형, 중형, 대형  
    activity_level VARCHAR(20), -- 활발함, 보통, 적음  
    health_condition VARCHAR(50)  
);  
  
CREATE TABLE Routes (  
    route_id SERIAL PRIMARY KEY,  
    user_id INT REFERENCES Users(user_id),  
    route_data JSONB,  
    evaluation TEXT,  
    recommended_for VARCHAR(50)  
);
```

### 4. 근처 애견인 알림 기능 (실시간 만남 알림)

- 사용자가 산책 중일 때 **WebSockets (Socket.io)** 나 **Firebase Messaging** 으로 주변 사용자와 연결.
- 근처 반경 (예: 100m 이내)에 동일 앱을 사용하는 사람이 있다면 진동 및 알림 제공.

### 예제 코드 (Socket.io)

```
from fastapi import FastAPI, WebSocket  
from typing import List
```

```

app = FastAPI()
connected_users: List[WebSocket] = []

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    connected_users.append(websocket)

    try:
        while True:
            data = await websocket.receive_text()
            for user in connected_users:
                await user.send_text(f"근처 사용자 발견: {data}")
    except:
        connected_users.remove(websocket)

```

## 🔍 5. 사용자 피드백 저장 및 분석

- 사용자가 평가를 남기면 이를 데이터베이스에 저장하고, 이후 경로 추천 시 활용.
- 리뷰 분석을 위해 간단한 NLP 모델 (Scikit-Learn) 활용 가능.

## 17 개발 일정 (8주 프로젝트)

주차	작업 내용
1주차	GPS 트래킹 기능 구축, 사용자의 경로 기록 기능 구현
2주차	강아지 정보 입력 시스템 및 데이터베이스 구축
3주차	경로 평가 기능 및 사용자 피드백 시스템 구축
4주차	근처 사용자 알림 기능 구축 (Socket.io)
5주차	사용자 맞춤 경로 추천 알고리즘 개발
6주차	UI/UX 디자인 및 최적화
7주차	디버깅 및 배포 (앱스토어 or 웹 배포)
8주차	사용자 피드백 반영 및 발표 준비

## 💡 차별화 포인트

- 📍 GPS 기반 경로 평가 (특정 강아지 정보별로 평가)
- 🔔 실시간 애견인 만남 알림 기능 (커뮤니티 활성화)
- 🐾 사용자 맞춤형 경로 추천 시스템 (리뷰 기반 학습 모델)

이걸 실제로 구현하려면 백엔드와 프론트엔드를 어떻게 연결하는지, 그리고 사용자 인터페이스를 어떻게 설계할지 알려줄까? 😊

### ▼ 이설후

#### ▼ 여행기록

프로젝트 이름: **Travelog.AI** - 너만의 감성 여행 플래너 & 기록 도우미

## 🧭 프로젝트 개요

**Travelog.AI**는 사용자의 취향과 예산을 기반으로 여행 루트를 자동으로 생성해주고, 여행 후에는 사진과 간단한 기록만으로 감성적인 여행 회고록을 만들어주는 올인원 AI 여행 도우미입니다.

여행의 시작부터 끝까지 - **계획, 실행, 기록** - 모든 순간을 AI가 함께하며, 사용자는 단순히 여행을 '가는 것'이 아닌 '기억되는 경험'으로 남길 수 있습니다.

## ✨ 핵심 기능

구분	기능	설명
여행 전	여행 성향 분석	GPT가 사용자 취향(자연, 맛집, 휴식 등)을 파악하여 스타일 분류
여행 전	일정 & 루트 추천	GPT가 2박 3일 등 일정 구성 + 동선 고려하여 장소 제안
여행 전	지도 시각화	추천 경로를 Folium 기반 지도에 표시
여행 전	<b>예산 기반 루트 생성</b>	입력된 예산을 환율로 변환해 숙소/식사/교통/체험 항목별 소비 제안
여행 후	사진 업로드	여행 사진 여러 장 업로드 가능
여행 후	한 줄 기록 입력	각 사진별 한 줄 설명 (또는 자동 캡션 지원)
여행 후	감성 회고록 자동 생성	GPT가 스토리, 일기, 영상 자막 등 감성적인 문장 생성

기타	저장/공유 기능	회고록을 PDF로 저장하거나 소셜 공유 가능
----	----------	--------------------------

## 전체 사용자 흐름 (User Flow)

### 1. 여행 전 단계

- "나는 바다랑 맛집 좋아해!" + 예산 입력
- AI가 성향 분석 + 추천 루트 제시 + 예산 계획까지 자동 구성
- 지도와 표로 확인 → 확정 일정표 저장


### 2. 여행 후 단계

- 사진 업로드 + 한 줄 메모 입력
- AI가 감성적 문장으로 회고록 자동 생성
- PDF 파일로 저장하거나 공유 가능

## 예산 기반 루트 생성 - 세부 예시

입력: "2박 3일 일본 여행, 예산 700,000원"

₩ 실시간 환율: 1엔 = 9.1원 → 총 ¥76,923

 여행 예산 분배:

- 숙박: ¥30,000 (비즈니스 호텔)
- 식사: ¥15,000 (평균 ¥1,500/식)
- 교통: ¥5,000 (패스 포함)
- 입장/체험: ¥10,000
- 여유/쇼핑: ¥16,923

→ 그 예산 안에서 GPT가 최적의 루트와 활동 구성까지 제안

## 기술 스택

분야	기술
----	----

언어	Python 3.10+
프론트	Streamlit (MVP), 또는 React 확장 가능
백엔드	FastAPI (선택)
AI 엔진	OpenAI GPT API + LangChain
이미지 처리	Pillow, OpenCV
지도 시각화	OpenStreetMap + Folium
환율 API	exchangerate.host API (무료)
데이터 저장	SQLite / Firebase
회고록 출력	FPDF / ReportLab (PDF 저장용)

## 프로젝트의 차별점

기존 여행 서비스	Travelog.AI
단순 관광지 추천	사용자의 취향과 예산을 함께 고려한 일정 자동 생성
기록은 수동 작성	사진 + 메모만으로 AI가 감성 회고록 자동 생성
환율 반영 안 됨	실시간 환율로 예산 계획까지 구성
기록 & 계획 분리됨	여행 전후를 하나로 연결하는 완성형 경험 제공

## 타겟 사용자

- 여행을 처음 가보는 대학생, 사회초년생
- 감성적인 여행 기록을 남기고 싶은 SNS 유저
- 귀찮은 걸 싫어하는 사람 (일정 & 기록 자동화)

## 향후 확장 아이디어

- 항공권 실시간 최저가 연동 (크롤링 or API)
- GPT 프롬프트 자동 최적화 (여행 스타일별 템플릿)
- 동영상 생성 기능 (사진 + 회고록 → 영상 자동화)
- 사용자간 일정 공유 기능 / SNS 연동

## 요약

Travelog.AI는 여행의 시작과 끝을 연결하는 감성+실용 통합 플랫폼으로,  
사용자의 성향과 예산에 맞는 맞춤형 루트 생성과  
여행 후 감성적인 회고록 자동화를 통해  
'여행을 완성된 추억'으로 바꿔주는 AI 서비스입니다.

#### ▼ 강아지 산책일기

프로젝트 이름: **강아지가 바라본 세상**  
*반려견 시점 자동 그림 스토리북 생성기*

### 프로젝트 개요

"강아지가 바라본 세상"은 사용자가 산책 중 찍은 강아지 사진을 업로드하면, AI가 이를 시간 순서로 정리하고, 마치 강아지가 하루를 이야기하듯 따뜻한 그림일기를 자동으로 생성해주는 감성 중심의 웹 서비스입니다.

사진에 담긴 순간들을 AI가 해석해 동화처럼 풀어내며, 반려견과의 추억을 특별한 콘텐츠로 남길 수 있게 돕습니다.

### 핵심 기능

기능	설명
사진 업로드	사용자가 산책 중 촬영한 사진을 업로드
시간 순 정렬	EXIF 정보를 활용해 사진 순서 정리
키워드 추출	각 사진에 대한 간단한 설명 또는 자동 캡션 생성
스토리 생성	GPT가 강아지 시점에서 동화 형식 이야기 작성
일기 구성 출력	사진 + 설명 + AI가 쓴 이야기 통합 뷰 제공
다운로드 기능	그림일기 PDF로 저장 또는 공유 기능 제공

### 개발 방법 요약

#### 1. 사진 업로드 + 메타데이터 추출

- 사용자가 여러 장의 사진 업로드 → EXIF 시간 정보 추출



## 2. 텍스트 정보 생성 (선택)

- 이미지 캡션 수동 입력 또는 CLIP 기반 추출

## 3. GPT 프롬프트 구성

- 키워드 또는 캡션 기반으로 이야기 생성용 입력 구성

## 4. GPT로 스토리 생성

- LangChain 활용해 이야기 구성 흐름 자동화

## 5. 스토리북 형식으로 렌더링

- 사진 + 이야기 + 날짜를 Streamlit 등으로 UI 구현

## 6. PDF or 공유 기능 구현

- FPDF/ReportLab 사용해 파일 생성 가능

## 전체 프로세스 요약

1. 사용자 사진 업로드
2. EXIF 정보 기반 시간순 정렬
3. 각 사진에 대해 수동 or 자동 캡션 생성
4. 캡션 기반 GPT 프롬프트 구성
5. GPT로 동화 형식 이야기 생성
6. 이야기와 사진을 결합하여 그림일기 형식 출력
7. 사용자가 PDF로 저장하거나 공유

## 예상 개발 소요 시간 (MVP 기준)

단계	기간	설명
요구사항 정리 & 기획	1일	기능 정리 및 역할 분배
기본 UI 구현	1~2일	Streamlit 기반 업로드/출력 화면 구성
EXIF 정렬 & 캡션 입력 처리	1일	이미지 정렬 및 키워드 수집 처리
GPT 연동 및 이야기 생성	1일	LangChain 구성 및 스토리 출력 확인

일기 렌더링 & 저장 기능	1~2일	PDF 출력 및 공유 기능 포함
테스트 및 시연 정리	1일	샘플 데이터로 결과물 준비
총계	5~7일	3~4인 팀 기준, MVP 수준 개발

## 현재 상용화된 유사 서비스와의 차이점

서비스명	기능 요약	차별점
Lollipop AI Diary	아기 영상 기반 AI 일기 자동 생성	강아지 전용 + 동화 스타일 없음
Childbook.ai	이야기 → 삽화 생성 기반 AI 동화 제작	텍스트 중심, 일기형 UX 아님
Recraft (AI Illustration)	사용자가 텍스트 입력 → 삽화 생성	사용자 프롬프트 기반, 반려동물 중심 아님

### 결론:

- 본 프로젝트는 **\*\*\*사진 기반 + 반려동물 시점 + 동화 스타일 스토리 생성\*\*\***이라는 측면에서 **새롭고 감성적인 경험을 제공하며,**
- **일상 기록 + AI 창작 + 정서적 만족**을 동시에 충족하는 독창적 서비스입니다.

## 사용 기술 스택

분야	기술
언어	Python 3.10 이상
프론트엔드	Streamlit or React (MVP는 Streamlit 추천)
백엔드/API	FastAPI (선택)
AI 텍스트 생성	OpenAI GPT API + LangChain
이미지 처리	Pillow, OpenCV, EXIF
저장	Firebase / SQLite (간단한 저장용)
PDF 생성	FPDF, ReportLab (선택)

## 타겟 사용자

- 반려견과의 추억을 특별하게 기록하고 싶은 보호자
- 반려동물 일상을 감성적으로 표현하고 싶은 일반 사용자

- 감성 기반 SNS 콘텐츠를 만들고 싶은 MZ세대
- 


## 향후 확장 아이디어

- Stable Diffusion을 통한 동화 스타일 이미지 자동 생성
  - 강아지 이름/성격에 따라 문체 변화 적용
  - TTS 연동: 그림일기를 음성으로 읽어주는 기능
  - 사용자별 "한 달 그림책 자동 완성 기능"
- 

## 요약

강아지의 시선으로 바라본 세상을 이야기로 풀어주는,  
나만을 위한 감성 그림일기 서비스

<https://blog.naver.com/ckdgus5290/222200219365>

 AI 기술을 따뜻하게, 그리고 재미있게 활용하는 진짜 실용+감성 프로젝트!

### ▼ 김시연

- 이미지 생성 모델을 이용한 서비스  
이상형에 부합하는 연예인 찾기  
본인만의 부합하는 이상형을 묘사하고, 이상형에 맞는 그림을 이미지화하고,  
그 이미지에 맞는 연예인을 찾아주는 서비스?